

XMC4300

Microcontroller Series
for Industrial Applications

XMC4000 Family

ARM[®] Cortex[®]-M4
32-bit processor core

Reference Manual

V1.1 2016-07

Microcontrollers

Edition 2016-07

**Published by
Infineon Technologies AG
81726 Munich, Germany**

**© 2016 Infineon Technologies AG
All Rights Reserved.**

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (www.infineon.com).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

XMC4300

Microcontroller Series
for Industrial Applications

XMC4000 Family

ARM[®] Cortex[®]-M4
32-bit processor core

Reference Manual

V1.1 2016-07

XMC4300 Reference Manual

Revision History: V1.1 2016-07

Previous Versions:

V1.0 2016-02

Page	Subjects (major changes since last revision)
8-35	Updated Flash/PMU chapter <ul style="list-style-type: none"> Corrected inconsistent definition of PREF_PCON.DBYP bit.
11-62	Updated SCU chapter <ul style="list-style-type: none"> Removed register CANCLKCR Added reset values for RSTSTAT and PRSTAT2 registers
19-135	Updated VADC chapter <ul style="list-style-type: none"> Add ECAT connections SYNC0 and SYNC1 to table

Trademarks

C166™, TriCore™, XMC™ and DAVE™ are trademarks of Infineon Technologies AG.
ARM®, ARM Powered®, Cortex®, Thumb® and AMBA® are registered trademarks of ARM, Limited.

CoreSight™, ETM™, Embedded Trace Macrocell™ and Embedded Trace Buffer™ are trademarks of ARM, Limited.

Synopsys™ is a trademark of Synopsys, Inc.

We Listen to Your Comments

Is there any information in this document that you feel is wrong, unclear or missing?
Your feedback will help us to continuously improve the quality of this document.
Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com



Table of Contents

1	Introduction	1-1
1.1	Architectural Overview	1-1
1.1.1	Block Diagram	1-3
1.2	System Core Components	1-3
1.3	On-Chip System Resources	1-5
1.3.1	Memories	1-5
1.4	Bus System	1-6
1.4.1	Bus Matrix	1-6
1.4.2	Bus Interfaces	1-6
1.5	Communication	1-7
1.6	Analog Frontend	1-8
1.7	Industrial Control	1-9
1.8	On-Chip Debug Support	1-9
2	Central Processing Unit (CPU)	2-1
2.1	Overview	2-1
2.1.1	Features	2-2
2.1.2	Block Diagram	2-2
2.2	Programmers Model	2-4
2.2.1	Processor Mode and Privilege Levels for Software Execution	2-4
2.2.2	Stacks	2-4
2.2.3	Core Registers	2-6
2.2.4	Exceptions and Interrupts	2-17
2.2.5	Data Types	2-17
2.2.6	The Cortex Microcontroller Software Interface Standard	2-17
2.2.7	CMSIS functions	2-18
2.3	Memory Model	2-20
2.3.1	Memory Regions, Types and Attributes	2-20
2.3.2	Memory System Ordering of Memory Accesses	2-21
2.3.3	Behavior of Memory Accesses	2-22
2.3.4	Software Ordering of Memory Accesses	2-23
2.3.5	Memory Endianness	2-24
2.3.6	Synchronization Primitives	2-24
2.3.7	Programming Hints for the Synchronization Primitives	2-26
2.4	Instruction Set	2-26
2.5	Exception Model	2-26
2.5.1	Exception States	2-26
2.5.2	Exception Types	2-27
2.5.3	Exception Handlers	2-29
2.5.4	Vector Table	2-30
2.5.5	Exception Priorities	2-31

Table of Contents

2.5.6	Interrupt Priority Grouping	2-31
2.5.7	Exception Entry and Return	2-32
2.6	Fault Handling	2-36
2.6.1	Fault Types	2-37
2.6.2	Fault Escalation and Hard Faults	2-38
2.6.3	Fault Status Registers and Fault Address Registers	2-39
2.6.4	Lockup	2-39
2.7	Power Management	2-40
2.7.1	Entering Sleep Mode	2-40
2.7.2	Wakeup from Sleep Mode	2-41
2.7.3	The External Event Input	2-41
2.7.4	Power Management Programming Hints	2-42
2.8	Private Peripherals	2-42
2.8.1	About the Private Peripherals	2-42
2.8.2	System control block	2-42
2.8.2.1	System control block design hints and tips	2-43
2.8.3	System timer, SysTick	2-43
2.8.3.1	SysTick design hints and tips	2-43
2.8.4	Nested Vectored Interrupt Controller (NVIC)	2-43
2.8.4.1	Level-sensitive and pulse interrupts	2-44
2.8.4.2	NVIC design hints and tips	2-45
2.8.4.3	Using CMSIS functions to access NVIC	2-45
2.8.5	Memory Protection Unit (MPU)	2-46
2.8.5.1	MPU Access Permission Attributes	2-48
2.8.5.2	MPU Mismatch	2-50
2.8.5.3	Updating an MPU Region	2-50
2.8.5.4	MPU Design Hints and Tips	2-53
2.8.6	Floating Point Unit (FPU)	2-53
2.8.6.1	Enabling the FPU	2-54
2.9	PPB Registers	2-54
2.9.1	SCS Registers	2-58
2.9.2	SysTick Registers	2-84
2.9.3	NVIC Registers	2-87
2.9.4	MPU Registers	2-93
2.9.5	FPU Registers	2-101
3	Bus System	3-1
3.1	Bus Interfaces	3-1
3.2	Bus Matrix	3-1
4	Service Request Processing	4-1
4.1	Overview	4-1
4.1.1	Features	4-1
4.1.2	Block Diagram	4-2

Table of Contents

4.2	Service Request Distribution	4-3
4.3	Interrupt Service Requests	4-4
4.4	DMA Line Router (DLR)	4-7
4.4.1	Functional Description	4-7
4.4.2	DMA Service Request Source Selection	4-9
4.5	Event Request Unit (ERU)	4-14
4.5.1	Event Request Select Unit (ERS)	4-15
4.5.2	Event Trigger Logic (ETLx)	4-16
4.5.3	Cross Connect Matrix	4-17
4.5.4	Output Gating Unit (OGUy)	4-18
4.6	Service Request Generation	4-21
4.7	Debug Behavior	4-22
4.8	Power, Reset and Clock	4-22
4.9	Initialization and System Dependencies	4-22
4.10	Registers	4-23
4.10.1	DLR Registers	4-24
4.10.2	ERU Registers	4-27
4.11	Interconnects	4-32
4.11.1	ERU0 Connections	4-33
4.11.2	ERU1 Connections	4-36
5	General Purpose DMA (GPDMA)	5-1
5.1	Overview	5-1
5.1.1	Features	5-1
5.1.2	Block Diagram	5-3
5.2	Functional Description	5-4
5.2.1	Terminology	5-4
5.2.2	Variable Definitions	5-7
5.2.3	Flow Controller and Transfer Type	5-8
5.2.4	Handshaking Interface	5-9
5.2.4.1	Hardware Handshaking	5-10
5.2.4.2	Software Handshaking	5-10
5.2.4.3	Handshaking with GPDMA as Flow Controller	5-11
5.2.4.4	Handshaking with Peripheral as Flow Controller	5-13
5.2.5	FIFO Usage	5-14
5.2.6	Bus and Channel Locking	5-15
5.2.7	Scatter/Gather	5-17
5.2.8	Abnormal Transfer Termination	5-20
5.3	Basic Transfers	5-21
5.3.1	Block transfer with GPDMA as the flow controller	5-22
5.3.2	Effect of maximum AMBA burst length on a block transfer	5-23
5.4	Multi Block Transfers	5-27
5.4.1	Block Chaining Using Linked Lists	5-27

Table of Contents

5.4.2	Auto-Reloading of Channel Registers	5-32
5.4.3	Contiguous Address Between Blocks	5-32
5.4.4	Suspension of Transfers Between Blocks	5-32
5.4.5	Ending Multi-Block Transfers	5-33
5.4.6	Programing Examples	5-34
5.4.6.1	Single-block Transfer	5-34
5.4.6.2	Multi-Block Transfer with Source Address Auto-Reloaded and Contiguous Destination Address 5-35	
5.4.6.3	Multi-Block Transfer with Source and Destination Address Auto- Reloaded 5-39	
5.4.6.4	Multi-Block Transfer with Source Address Auto-Reloaded and Linked List Destination Address 5-43	
5.4.6.5	Multi-Block DMA Transfer with Linked List for Source and Contiguous Destination Address 5-48	
5.4.6.6	Multi-Block Transfer with Linked List for Source and Destination .	5-51
5.5	Service Request Generation	5-57
5.6	Power, Reset and Clock	5-58
5.7	Initialization and System Dependencies	5-58
5.8	Registers	5-59
5.8.1	Configuration and Channel Enable Registers	5-63
5.8.2	Channel Registers	5-64
5.8.3	Interrupt Registers	5-96
5.8.4	Software Handshaking Registers	5-107
5.8.5	Miscellaneous GPDMA Registers	5-113
6	Flexible CRC Engine (FCE)	6-1
6.1	Overview	6-1
6.1.1	Features	6-1
6.1.2	Application Mapping	6-2
6.1.3	Block Diagram	6-2
6.2	Functional Description	6-3
6.2.1	Basic Operation	6-5
6.2.2	Automatic Signature Check	6-5
6.2.3	Register protection and monitoring methods	6-6
6.3	Service Request Generation	6-8
6.4	Debug Behavior	6-9
6.5	Power, Reset and Clock	6-9
6.6	Initialization and System Dependencies	6-9
6.7	Registers	6-11
6.7.1	System Registers description	6-12
6.7.2	CRC Kernel Control/Status Registers	6-13
6.8	Interconnects	6-24
6.9	Properties of CRC code	6-24

7	Memory Organization	7-1
7.1	Overview	7-1
7.1.1	Features	7-1
7.1.2	Cortex-M4 Address Space	7-1
7.2	Memory Regions	7-3
7.3	Memory Map	7-3
7.4	Service Request Generation	7-7
7.5	Debug Behavior	7-9
7.6	Power, Reset and Clock	7-9
7.7	Initialization and System Dependencies	7-9
7.8	Registers	7-10
8	Flash and Program Memory Unit (PMU)	8-1
8.1	Overview	8-1
8.1.1	Block Diagram	8-1
8.2	Boot ROM (BROM)	8-2
8.2.1	BROM Addressing	8-2
8.3	Prefetch Unit	8-2
8.3.1	Overview	8-2
8.3.2	Operation	8-3
8.3.2.1	Instruction Buffer	8-3
8.3.2.2	Data Buffer	8-3
8.3.2.3	PMU Interface	8-4
8.4	Program Flash (PFLASH)	8-5
8.4.1	Overview	8-5
8.4.1.1	Features	8-5
8.4.2	Definition of Terms	8-6
8.4.3	Flash Structure	8-7
8.4.4	Flash Read Access	8-8
8.4.5	Flash Write and Erase Operations	8-9
8.4.6	Modes of Operation	8-9
8.4.7	Command Sequences	8-9
8.4.7.1	Command Sequence Definitions	8-10
8.4.7.2	Flash Page Programming Example	8-15
8.4.8	Flash Protection	8-17
8.4.8.1	Configuring Flash Protection in the UCB	8-17
8.4.8.2	Flash Read Protection	8-18
8.4.8.3	Flash Write and OTP Protection	8-20
8.4.8.4	System Wide Effects of Flash Protection	8-22
8.4.9	Data Integrity and Safety	8-22
8.4.9.1	Error-Correcting Code (ECC)	8-22
8.4.9.2	Margin Checks	8-23
8.5	Service Request Generation	8-23

Table of Contents

8.5.1	Interrupt Control	8-24
8.5.2	Trap Control	8-24
8.5.3	Handling Errors During Operation	8-25
8.5.3.1	SQER “Sequence Error”	8-25
8.5.3.2	PFOPER “Operation Error”	8-26
8.5.3.3	PROER “Protection Error”	8-26
8.5.3.4	VER “Verification Error”	8-27
8.5.3.5	PFSBER/DFSBER “Single-Bit Error”	8-28
8.5.3.6	Handling Flash Errors During Startup	8-29
8.6	Power, Reset and Clock	8-29
8.6.1	Power Supply	8-29
8.6.2	Power Reduction	8-29
8.6.3	Reset Control	8-31
8.6.3.1	Resets During Flash Operation	8-31
8.6.4	Clock	8-33
8.7	Registers	8-33
8.7.1	PMU Registers	8-33
8.7.1.1	PMU ID Register	8-34
8.7.2	Prefetch Registers	8-35
8.7.2.1	Prefetch Configuration Register	8-35
8.7.3	Flash Registers	8-37
8.7.3.1	Flash Status Definition	8-38
8.7.3.2	Flash Configuration Control	8-43
8.7.3.3	Flash Identification Register	8-48
8.7.3.4	Margin Check Control Register	8-49
8.7.3.5	Protection Configuration Indication	8-49
9	Window Watchdog Timer (WDT)	9-1
9.1	Overview	9-1
9.1.1	Features	9-1
9.1.2	Block Diagram	9-2
9.2	Time-Out Mode	9-3
9.3	Pre-warning Mode	9-4
9.4	Bad Service Operation	9-5
9.5	Service Request Processing	9-7
9.6	Debug Behavior	9-7
9.7	Power, Reset and Clock	9-7
9.8	Initialization and Control Sequence	9-7
9.8.1	Initialization & Start of Operation	9-8
9.8.2	Reconfiguration & Restart of Operation	9-8
9.8.3	Software Stop & Resume Operation	9-9
9.8.4	Enter Sleep/Deep Sleep & Resume Operation	9-9
9.8.5	Prewarning Alarm Handling	9-9

Table of Contents

9.9	Registers	9-11
9.9.1	Registers Description	9-11
9.10	Interconnects	9-16
10	Real Time Clock (RTC)	10-1
10.1	Overview	10-1
10.1.1	Features	10-1
10.1.2	Block Diagram	10-1
10.2	RTC Operation	10-2
10.3	Register Access Operations	10-3
10.4	Service Request Processing	10-4
10.4.1	Periodic Service Request	10-4
10.4.2	Timer Alarm Service Request	10-4
10.5	Wake-up From Hibernation Trigger	10-4
10.5.1	Periodic Wake-up Trigger Generation	10-4
10.5.2	Timer Alarm Wake-up Trigger Generation	10-4
10.6	Debug behavior	10-5
10.7	Power, Reset and Clock	10-5
10.8	Initialization and Control Sequence	10-5
10.8.1	Initialization & Start of Operation	10-5
10.8.2	Re-configuration & Re-start of Operation	10-6
10.8.3	Configure and Enable Periodic Event	10-6
10.8.4	Configure and Enable Timer Event	10-6
10.9	Registers	10-8
10.9.1	Registers Description	10-8
10.10	Interconnects	10-20
11	System Control Unit (SCU)	11-1
11.1	Overview	11-1
11.1.1	Features	11-1
11.1.2	Block Diagram	11-2
11.2	Miscellaneous Control Functions	11-5
11.2.1	Startup Software Support	11-5
11.2.2	Service Requests	11-6
11.2.2.1	Service Request Sources	11-6
11.2.3	Memory Parity Protection	11-7
11.2.3.1	Parity Error Handling	11-7
11.2.4	Trap Generation	11-9
11.2.4.1	Trap Sources	11-9
11.2.5	Die Temperature Measurement	11-10
11.2.5.1	Temperature Measurement	11-10
11.2.5.2	Offset Adjustment	11-11
11.2.5.3	Gain Adjustment	11-12
11.2.6	Retention Memory	11-13

Table of Contents

11.2.7	Out of Range Comparator Control	11-14
11.3	Power Management	11-14
11.3.1	Functional Description	11-14
11.3.2	System States	11-15
11.3.3	Hibernate Domain Operating Modes	11-17
11.3.4	Embedded Voltage Regulator (EVR)	11-19
11.3.5	Power-on Reset	11-19
11.3.6	Supply Watchdog (SWD)	11-20
11.3.7	Power Validation	11-20
11.3.8	Supply Voltage Brown-out Detection	11-20
11.3.9	Hibernate Domain Power Management	11-21
11.3.10	Flash Power Control	11-21
11.4	Hibernate Control	11-21
11.4.1	Hibernate Mode	11-21
11.4.2	Hibernate Domain Pin Functions	11-22
11.4.3	System Level Integration	11-23
11.5	Reset Control	11-26
11.5.1	Supported Reset types	11-26
11.5.2	Peripheral Reset Control	11-27
11.5.3	EtherCAT Reset	11-28
11.5.4	Reset Status	11-28
11.6	Clock Control	11-29
11.6.1	Block Diagram	11-29
11.6.2	Clock Sources	11-30
11.6.3	Clock System Overview	11-31
11.6.3.1	Clock System Architecture	11-33
11.6.4	High Precision Oscillator Circuit (OSC_HP)	11-37
11.6.5	Backup Clock Source	11-38
11.6.6	Main PLL	11-39
11.6.6.1	Features	11-39
11.6.6.2	System PLL Functional Description	11-39
11.6.6.3	Configuration and Operation of the Prescaler Mode	11-43
11.6.6.4	Bypass Mode	11-45
11.6.6.5	System Oscillator Watchdog (OSC_WDG)	11-45
11.6.6.6	VCO Power Down Mode	11-46
11.6.6.7	PLL Power Down Mode	11-46
11.6.6.8	Application Hints	11-46
11.6.7	Internally Generated System Clock Calibration	11-47
11.6.7.1	Factory Calibration	11-47
11.6.7.2	Automatic Calibration	11-47
11.6.7.3	Alternative Internal Clock Calibration	11-47
11.6.8	USB PLL	11-48
11.6.9	Ultra Low Power Oscillator	11-50

Table of Contents

11.6.9.1	OSC_ULP Oscillator Watchdog (ULPWDG)	11-50
11.6.10	Internal Slow Clock Source	11-50
11.6.11	Clock Gating Control	11-50
11.7	Debug Behavior	11-51
11.8	Power, Reset and Clock	11-51
11.9	Initialization and System Dependencies	11-52
11.9.1	Power-Up	11-53
11.9.2	Power-on Reset Release	11-54
11.9.3	System Reset Release	11-55
11.9.4	Clock System Setup	11-57
11.9.5	Configuration of Special System Functions	11-59
11.9.6	Configuration of Miscellaneous Functions	11-60
11.10	Registers	11-62
11.10.1	GCU Registers	11-67
11.10.2	PCU Registers	11-121
11.10.3	HCU Registers	11-126
11.10.4	RCU Registers	11-134
11.10.5	CCU Registers	11-149
11.11	Interconnections	11-186
12	LED and Touch-Sense (LEDTS)	12-1
12.1	Overview	12-1
12.1.1	Features	12-1
12.1.2	Block Diagram	12-2
12.2	Functional Overview	12-4
12.3	LED Drive Mode	12-7
12.3.1	LED Pin Assignment and Current Capability	12-9
12.4	Touch-Sense Mode	12-10
12.4.1	Finger Sensing	12-13
12.5	Operating both LED Drive and Touch-Sense Modes	12-14
12.6	Service Request Processing	12-14
12.7	Debug Behavior	12-15
12.8	Power, Reset and Clock	12-15
12.9	Initialisation and System Dependencies	12-15
12.9.1	Function Enabling	12-15
12.9.2	Interpretation of Bit Field FNCOL	12-16
12.9.3	LEDTS Timing Calculations	12-17
12.9.4	Time-Multiplexed LED and Touch-Sense Functions on Pin	12-18
12.9.5	LEDTS Pin Control	12-18
12.9.6	Software Hints	12-20
12.9.7	Hardware Design Hints	12-21
12.10	Registers	12-22
12.10.1	Registers Description	12-23

Table of Contents

12.11	Interconnects	12-36
13	SD/MMC Interface (SDMMC)	13-1
13.1	Overview	13-1
13.1.1	Features	13-1
13.1.2	Block Diagram	13-2
13.2	Functional Description	13-4
13.3	Card Detection	13-6
13.4	Data Transfer Modes	13-6
13.5	Read/ Write Operation	13-7
13.5.1	Write Operation	13-7
13.5.2	Read Operation	13-7
13.5.3	Abort Transaction	13-7
13.6	Special Command Types	13-9
13.7	Error Detection	13-10
13.8	Service Request Generation	13-10
13.9	Debug Behavior	13-10
13.10	Power, Reset and Clocks	13-10
13.10.1	Tap delay control	13-11
13.11	Initialisation and System Dependencies	13-12
13.11.1	Setup SDMMC Data Transfer	13-12
13.11.2	Read Operation	13-14
13.11.3	Write Operation	13-14
13.11.4	Abort Transaction	13-15
13.11.5	Write Protection and Card Detection Software Control	13-16
13.12	Registers	13-17
13.12.1	Registers Description	13-21
13.13	Interconnects	13-90
14	Ethernet MAC (ETH)	14-1
14.1	Overview	14-1
14.1.1	ETH Core Features	14-2
14.1.2	DMA Block Features	14-3
14.1.3	Transaction Layer (MTL) Features	14-3
14.1.4	Monitoring, Test, and Debugging Support Features	14-5
14.1.5	Block Diagram	14-5
14.2	Functional Description	14-5
14.2.1	ETH Core	14-6
14.2.1.1	Transmission	14-6
14.2.1.2	MAC Transmit Interface Protocol	14-10
14.2.1.3	Reception	14-10
14.2.2	MAC Transaction Layer (MTL)	14-18
14.2.2.1	Transmit Path	14-18
14.2.2.2	Receive Path	14-24

Table of Contents

14.2.3	DMA Controller	14-26
14.2.3.1	Initialization	14-27
14.2.3.2	Transmission	14-30
14.2.3.3	Reception	14-35
14.2.3.4	Interrupts	14-39
14.2.4	DMA Descriptors	14-41
14.2.4.1	Descriptor Formats	14-41
14.2.5	MAC Management Counters	14-74
14.2.6	Power Management Block	14-74
14.2.6.1	PMT Block Description	14-75
14.2.6.2	Remote Wake-Up Frame Detection	14-77
14.2.6.3	Magic Packet Detection	14-77
14.2.6.4	System Considerations During Power-Down	14-78
14.2.7	PHY Interconnect	14-79
14.2.7.1	PHY Interconnect selection	14-79
14.2.8	Station Management Interface	14-79
14.2.8.1	Station Management Functions	14-80
14.2.8.2	Station Management Write Operation	14-81
14.2.8.3	Station Management Read Operation	14-81
14.2.9	Media Independent interface	14-82
14.2.10	Reduced Media Independent Interface	14-83
14.2.10.1	RMII Block Diagram	14-84
14.2.10.2	RMII Block Overview	14-84
14.2.10.3	Transmit Bit Ordering	14-85
14.2.10.4	RMII Transmit Timing Diagrams	14-85
14.2.11	IEEE 1588-2002 Overview	14-88
14.2.11.1	Reference Timing Source	14-90
14.2.11.2	Transmit Path Functions	14-90
14.2.11.3	Receive Path Functions	14-90
14.2.11.4	Time Stamp Error Margin	14-91
14.2.11.5	Frequency Range of Reference Timing Clock	14-91
14.2.11.6	Advanced Time Stamp Feature Support	14-92
14.2.11.7	Peer-to-Peer PTP (Pdelay) Transparent Clock (P2P TC) Message Support	14-92
14.2.11.8	Clock Types	14-94
14.2.11.9	PTP Processing and Control	14-95
14.2.11.10	Reference Timing Source (for Advance Timestamp Feature)	14-99
14.2.11.11	Transmit Path Functions	14-100
14.2.11.12	Receive Path Functions	14-100
14.2.12	System Time Register Module	14-101
14.2.13	Application BUS Interface	14-103
14.3	Service Request Generation	14-105
14.3.1	DMA Service Requests	14-106

Table of Contents

14.3.2	Power Management Service Requests	14-106
14.3.3	System Time Module	14-106
14.3.4	MAC Management Counter Service Requests	14-106
14.4	Debug	14-107
14.5	Power Reset and Clock	14-107
14.6	ETH Registers	14-108
14.6.1	Register Description	14-108
14.6.2	Registers Overview	14-109
14.6.2.1	Registers Description	14-125
14.7	Interconnects	14-332
14.7.1	ETH Pins	14-332
15	EtherCAT Slave Controller (ECAT)	15-1
15.1	Overview	15-1
15.1.1	Features	15-1
15.1.2	Block Diagram	15-2
15.1.3	EtherCAT Slave Controller Function Blocks	15-2
15.1.4	Further Reading on EtherCAT	15-3
15.2	EtherCAT Protocol	15-3
15.3	Ethernet Physical Layer	15-3
15.3.1	MII Interface	15-4
15.3.1.1	LINK_MII Signal	15-4
15.3.2	PHY Management Interface	15-5
15.3.2.1	PHY Management Interface Signals	15-5
15.3.2.2	PHY Address Configuration	15-6
15.3.2.3	PHY Reset Signal	15-6
15.3.2.4	PHY Clock	15-6
15.4	FMMU	15-6
15.5	SyncManager	15-6
15.6	EtherCAT State Machine	15-8
15.7	EtherCAT Slave Controller Address Space Overview	15-8
15.8	ESI EEPROM	15-9
15.8.1	EEPROM Emulation	15-10
15.9	Sync/Latch Signals	15-12
15.10	LED Signals (Indicators)	15-13
15.10.1	RUN LED	15-13
15.10.2	ERR LED	15-13
15.10.3	STATE RUN LED	15-14
15.10.4	LINK ACT LED	15-14
15.11	Service Request Generation	15-15
15.12	Debug Behavior	15-15
15.13	Power, Reset and Clock	15-16
15.13.1	Power	15-16

Table of Contents

15.13.2	Module Reset	15-16
15.13.3	EtherCAT Reset	15-16
15.13.4	Module Clock	15-16
15.14	Initialization and System Dependencies	15-16
15.14.1	Process and User RAM Parity handling	15-17
15.15	Registers	15-19
15.15.1	ECAT kernel registers	15-26
15.15.2	Station Address	15-32
15.15.3	Write Protection	15-33
15.15.4	Data Link Layer	15-37
15.15.5	Application Layer	15-46
15.15.6	PDI / ESC Configuration	15-51
15.15.7	Interrupts	15-56
15.15.8	Error Counters	15-68
15.15.9	Watchdogs	15-71
15.15.10	SII EEPROM Interface	15-75
15.15.11	MII Management Interface	15-81
15.15.12	FMMU	15-87
15.15.13	SyncManager	15-94
15.15.14	DC – Receive Times	15-102
15.15.15	Time Loop Control Unit	15-104
15.15.16	SyncManager Event Times	15-131
15.15.17	ESC specific registers (0x0E00:0x0E01)	15-134
15.16	Interconnects	15-135
16	Universal Serial Bus (USB)	16-1
16.1	Overview	16-1
16.1.1	Features	16-1
16.1.2	Block Diagram	16-2
16.2	Functional Description	16-3
16.2.1	OTG Dual-Role Device (DRD)	16-3
16.2.2	USB Host	16-3
16.2.3	USB Device	16-4
16.2.4	FIFO Architecture	16-5
16.2.4.1	Host FIFO Architecture	16-5
16.2.4.2	Device FIFO Architecture	16-6
16.3	Programming Overview	16-7
16.3.1	Programming Options on DMA	16-7
16.3.1.1	DMA Mode	16-7
16.3.1.2	Slave Mode	16-7
16.3.2	Core Initialization	16-11
16.4	Host Programming Overview	16-12
16.4.1	Host Initialization	16-12

Table of Contents

16.4.2	Host Connection	16-13
16.4.3	Host Disconnection	16-13
16.4.4	Channel Initialization in Buffer DMA or Slave Mode	16-14
16.4.5	Halting a Channel	16-15
16.4.6	Selecting the Queue Depth	16-16
16.4.7	Handling Special Conditions	16-16
16.4.7.1	Handling Babble Conditions	16-16
16.4.7.2	Handling Disconnects	16-17
16.4.8	Host HFIR Functionality	16-17
16.4.8.1	HFIR Behaviour when HFIR.HFIRIdCtrl = 0 _B	16-17
16.4.8.2	HFIR Behaviour when HFIR.HFIRIdCtrl = 1 _B	16-18
16.4.9	Host Programming for Various USB Transactions	16-18
16.5	Host Programming in Slave mode	16-20
16.5.1	Writing the Transmit FIFO in Slave Mode	16-21
16.5.2	Reading the Receive FIFO in Slave Mode	16-22
16.5.3	Control Transactions in Slave Mode	16-23
16.5.4	Bulk and Control IN Transactions in Slave Mode	16-23
16.5.4.1	Normal Bulk and Control IN Operations	16-23
16.5.4.2	Handling Interrupts	16-24
16.5.5	Bulk and Control OUT/SETUP Transactions in Slave Mode	16-25
16.5.5.1	Normal Bulk and Control OUT/SETUP Operations	16-25
16.5.5.2	Handling Interrupts	16-27
16.5.6	Interrupt IN Transactions in Slave Mode	16-28
16.5.6.1	Normal Interrupt IN Operation	16-28
16.5.6.2	Handling Interrupts	16-29
16.5.7	Interrupt OUT Transactions in Slave Mode	16-31
16.5.7.1	Normal Interrupt OUT Operation	16-31
16.5.7.2	Handling Interrupts	16-31
16.5.8	Isochronous IN Transactions in Slave Mode	16-34
16.5.8.1	Normal Isochronous IN Operation	16-34
16.5.8.2	Handling Interrupts	16-35
16.5.9	Isochronous OUT Transactions in Slave Mode	16-36
16.5.9.1	Normal Isochronous OUT Operation	16-36
16.5.9.2	Handling Interrupts	16-37
16.6	Host Programming in Buffer DMA Mode	16-38
16.6.1	Control Transactions in Buffer DMA Mode	16-38
16.6.2	Bulk and Control IN Transactions in Buffer DMA Mode	16-38
16.6.2.1	Normal Bulk and Control IN Operations	16-39
16.6.2.2	Handling Interrupts	16-39
16.6.3	Bulk and Control OUT/SETUP Transactions in Buffer DMA Mode	16-40
16.6.3.1	Overview	16-40
16.6.3.2	Normal Bulk and Control OUT/SETUP Operations	16-40
16.6.3.3	NAK and NYET Handling With Internal DMA	16-41

Table of Contents

16.6.3.4	Handling Interrupts	16-43
16.6.4	Interrupt IN Transactions in Buffer DMA Mode	16-45
16.6.4.1	Normal Interrupt IN Operation	16-45
16.6.4.2	Handling Interrupts	16-46
16.6.5	Interrupt OUT Transactions in Buffer DMA Mode	16-47
16.6.5.1	Normal Interrupt OUT Operation	16-47
16.6.5.2	Handling Interrupts	16-49
16.6.6	Isochronous IN Transactions in Buffer DMA Mode	16-50
16.6.6.1	Normal Isochronous IN Operation	16-50
16.6.6.2	Handling Interrupts	16-51
16.6.7	Isochronous OUT Transactions in Buffer DMA Mode	16-52
16.6.7.1	Normal Isochronous OUT Operation	16-52
16.6.7.2	Handling Interrupts	16-53
16.7	Host Programming in Scatter-Gather DMA Mode	16-55
16.7.1	Programming Requirements	16-55
16.7.2	SPRAM Requirements	16-55
16.7.2.1	Descriptor Memory Structures	16-55
16.7.2.2	IN Memory Structure	16-59
16.7.2.3	OUT Memory Structure	16-62
16.7.3	Channel Initialization in Scatter-Gather DMA Mode	16-64
16.7.4	Asynchronous Transfers	16-65
16.7.4.1	Asynchronous Transfer Descriptor	16-66
16.7.5	Periodic Transfers	16-66
16.7.5.1	Isochronous Transactions	16-67
16.7.5.2	Interrupt Transactions	16-70
16.8	Device Programming Overview	16-72
16.8.1	Device Initialization	16-72
16.8.2	Device Connection	16-72
16.8.3	Device Disconnection	16-73
16.8.3.1	Device Soft Disconnection	16-73
16.8.4	Endpoint Initialization	16-74
16.8.4.1	Initialization on USB Reset	16-74
16.8.4.2	Initialization on Enumeration Completion	16-75
16.8.4.3	Initialization on SetAddress Command	16-75
16.8.4.4	Initialization on SetConfiguration/SetInterface Command	16-76
16.8.4.5	Endpoint Activation	16-76
16.8.4.6	Endpoint Deactivation	16-76
16.8.5	Programming OUT Endpoint Features	16-77
16.8.5.1	Disabling an OUT Endpoint	16-77
16.8.5.2	Stalling a Non-Isochronous OUT Endpoint	16-77
16.8.5.3	Setting the Global OUT NAK	16-78
16.8.5.4	Transfer Stop Programming for OUT Endpoints	16-79
16.8.6	Programming IN Endpoint Features	16-79

Table of Contents

16.8.6.1	Setting IN Endpoint NAK	16-79
16.8.6.2	IN Endpoint Disable	16-80
16.8.6.3	Timeout for Control IN Endpoints	16-81
16.8.6.4	Stalling Non-Isochronous IN Endpoints	16-81
16.8.6.5	Transfer Stop Programming for IN Endpoints	16-82
16.8.6.6	Non-Periodic IN Endpoint Sequencing	16-82
16.8.7	Worst-Case Response Time	16-83
16.8.8	Choosing the Value of GUSBCFG.USBTrdTim	16-83
16.8.9	Handling Babble Conditions	16-84
16.8.10	Device Programming Operations in Buffer DMA or Slave Mode ...	16-85
16.9	Device Programming in Slave Mode	16-87
16.9.1	Control Transfers	16-87
16.9.1.1	Control Write Transfers (SETUP, Data OUT, Status IN)	16-87
16.9.1.2	Control Read Transfers (SETUP, Data IN, Status OUT)	16-88
16.9.1.3	Two-Stage Control Transfers (SETUP/Status IN)	16-89
16.9.1.4	Packet Read from FIFO	16-91
16.9.2	IN Data Transfers	16-93
16.9.2.1	Packet Write	16-93
16.9.3	OUT Data Transfers	16-94
16.9.3.1	Control Setup Transactions	16-94
16.9.3.2	Handling More Than Three Back-to-Back SETUP Packets	16-97
16.9.4	Non-Periodic (Bulk and Control) IN Data Transfers	16-98
16.9.4.1	Examples	16-99
16.9.5	Non-Isochronous OUT Data Transfers	16-104
16.9.6	Isochronous OUT Data Transfers	16-108
16.9.7	Isochronous OUT Data Transfers Using Periodic Transfer Interrupt	16-109
16.9.8	Incomplete Isochronous OUT Data Transfers	16-114
16.9.9	Incomplete Isochronous IN Data Transfers	16-116
16.9.10	Periodic IN (Interrupt and Isochronous) Data Transfers	16-117
16.9.11	Periodic IN Data Transfers Using the Periodic Transfer Interrupt ..	16-119
16.9.12	Interrupt OUT Data Transfers Using Periodic Transfer Interrupt ..	16-125
16.10	Device Programming in Buffer DMA Mode	16-127
16.10.1	Control Transfers	16-127
16.10.1.1	Control Write Transfers (SETUP, Data OUT, Status IN)	16-127
16.10.1.2	Control Read Transfers (SETUP, Data IN, Status OUT)	16-128
16.10.1.3	Two-Stage Control Transfers (SETUP/Status IN)	16-129
16.10.2	OUT Data Transfers	16-129
16.10.2.1	Control Setup Transactions	16-129
16.10.3	Non-Periodic (Bulk and Control) IN Data Transfers	16-132
16.10.4	Non-Isochronous OUT Data Transfers	16-134
16.10.5	Incomplete Isochronous OUT Data Transfers	16-136
16.10.6	Periodic IN (Interrupt and Isochronous) Data Transfers	16-138

Table of Contents

16.10.7	Periodic IN Data Transfers Using the Periodic Transfer Interrupt . . .	16-140
16.10.8	Interrupt OUT Data Transfers Using Periodic Transfer Interrupt . . .	16-146
16.11	Device Programming in Scatter-Gather DMA Mode	16-148
16.11.1	Programming Overview	16-148
16.11.2	SPRAM Requirements	16-149
16.11.3	Descriptor Memory Structures	16-149
16.11.3.1	OUT Data Memory Structure	16-150
16.11.3.2	Isochronous OUT	16-156
16.11.3.3	Non-Isochronous OUT	16-156
16.11.3.4	IN Data Memory Structure	16-156
16.11.3.5	Descriptor Update Interrupt Enable Modes	16-162
16.11.3.6	DMA Arbitration in Scatter/Gather DMA Mode	16-162
16.11.3.7	Buffer Data Access on AHB in Scatter/Gather DMA Mode	16-162
16.11.4	Control Transfer Handling	16-163
16.11.5	Interrupt Usage for Control Transfers	16-163
16.11.6	Application Programming Sequence	16-164
16.11.7	Internal Data Flow	16-171
16.11.7.1	Three-Stage Control Write	16-171
16.11.7.2	Three-Stage Control Read	16-174
16.11.7.3	Two-Stage Control Transfer	16-176
16.11.7.4	Back to Back SETUP During Control Write	16-177
16.11.7.5	Back-to-Back SETUPS During Control Read	16-180
16.11.7.6	Extra Tokens During Control Write Data Phase	16-182
16.11.7.7	Extra Tokens During Control Read Data Phase	16-184
16.11.7.8	Premature SETUP During Control Write Data Phase	16-186
16.11.7.9	Premature SETUP During Control Read Data Phase	16-189
16.11.7.10	Premature Status During Control Write	16-191
16.11.7.11	Premature Status During Control Read	16-193
16.11.7.12	Lost ACK During Last Packet of Control Read	16-195
16.11.8	Bulk Transfer Handling in Scatter/Gather DMA Mode	16-195
16.11.8.1	Bulk IN Transfer in Scatter-Gather DMA Mode	16-196
16.11.8.2	Bulk OUT Transfer in Scatter-Gather DMA Mode	16-201
16.11.9	Interrupt Transfer Handling in Scatter/Gather DMA Mode	16-206
16.11.9.1	Interrupt IN Transfer in Scatter/Gather DMA Mode	16-207
16.11.9.2	Interrupt OUT Transfer in Scatter/Gather DMA Mode	16-207
16.11.10	Isochronous Transfer Handling in Scatter/Gather DMA Mode	16-207
16.11.10.1	Isochronous IN Transfer in Scatter/Gather DMA Mode	16-207
16.11.10.2	Isochronous OUT Transfer in Scatter/Gather DMA Mode	16-212
16.12	OTG Revision 1.3 Programming Model	16-214
16.12.1	A-Device Session Request Protocol	16-214
16.12.2	B-Device Session Request Protocol	16-215
16.12.3	A-Device Host Negotiation Protocol	16-217
16.12.4	B-Device Host Negotiation Protocol	16-218

Table of Contents

16.13	Clock Gating Programming Model	16-219
16.13.1	Host Mode Suspend and Resume With Clock Gating	16-219
16.13.2	Host Mode Suspend and Remote Wakeup With Clock Gating	16-220
16.13.3	Host Mode Session End and Start With Clock Gating	16-221
16.13.4	Host Mode Session End and SRP With Clock Gating	16-221
16.13.5	Device Mode Suspend and Resume With Clock Gating	16-222
16.13.6	Device Mode Suspend and Remote Wakeup With Clock Gating	16-222
16.13.7	Device Mode Session End and Start With Clock Gating	16-223
16.13.8	Device Mode Session End and SRP With Clock Gating	16-223
16.14	FIFO RAM Allocation	16-223
16.14.1	Data FIFO RAM Allocation	16-223
16.14.1.1	Device Mode RAM Allocation	16-225
16.14.1.2	Host Mode RAM Allocation	16-228
16.14.2	Dynamic FIFO Allocation	16-230
16.14.2.1	Dynamic FIFO Reallocation in Host Mode	16-230
16.14.2.2	Dynamic FIFO Reallocation in Device Mode	16-231
16.14.2.3	Flushing TxFIFOs in the Core	16-231
16.15	Service Request Generation	16-232
16.16	Debug Behaviour	16-233
16.17	Power, Reset and Clock	16-234
16.18	Initialization and System Dependencies	16-234
16.19	Registers	16-235
16.19.1	Register Description	16-242
16.20	Interconnects	16-345
17	Universal Serial Interface Channel (USIC)	17-1
17.1	Overview	17-1
17.1.1	Features	17-1
17.2	Operating the USIC	17-6
17.2.1	USIC Structure Overview	17-6
17.2.1.1	Channel Structure	17-6
17.2.1.2	Input Stages	17-6
17.2.1.3	Output Signals	17-8
17.2.1.4	Baud Rate Generator	17-9
17.2.1.5	Channel Events and Interrupts	17-10
17.2.1.6	Data Shifting and Handling	17-10
17.2.2	Operating the USIC Communication Channel	17-14
17.2.2.1	Protocol Control and Status	17-15
17.2.2.2	Mode Control	17-16
17.2.2.3	General Channel Events and Interrupts	17-17
17.2.2.4	Data Transfer Events and Interrupts	17-18
17.2.2.5	Baud Rate Generator Event and Interrupt	17-20
17.2.2.6	Protocol-specific Events and Interrupts	17-22

Table of Contents

17.2.3	Operating the Input Stages	17-22
17.2.3.1	General Input Structure	17-23
17.2.3.2	Digital Filter	17-25
17.2.3.3	Edge Detection	17-25
17.2.3.4	Selected Input Monitoring	17-26
17.2.3.5	Loop Back Mode	17-26
17.2.4	Operating the Baud Rate Generator	17-26
17.2.4.1	Fractional Divider	17-26
17.2.4.2	External Frequency Input	17-27
17.2.4.3	Divider Mode Counter	17-27
17.2.4.4	Capture Mode Timer	17-28
17.2.4.5	Time Quanta Counter	17-29
17.2.4.6	Master and Shift Clock Output Configuration	17-30
17.2.5	Operating the Transmit Data Path	17-31
17.2.5.1	Transmit Buffering	17-31
17.2.5.2	Transmit Data Shift Mode	17-32
17.2.5.3	Transmit Control Information	17-33
17.2.5.4	Transmit Data Validation	17-34
17.2.6	Operating the Receive Data Path	17-36
17.2.6.1	Receive Buffering	17-36
17.2.6.2	Receive Data Shift Mode	17-37
17.2.6.3	Baud Rate Constraints	17-38
17.2.7	Hardware Port Control	17-38
17.2.8	Operating the FIFO Data Buffer	17-39
17.2.8.1	FIFO Buffer Partitioning	17-40
17.2.8.2	Transmit Buffer Events and Interrupts	17-41
17.2.8.3	Receive Buffer Events and Interrupts	17-45
17.2.8.4	FIFO Buffer Bypass	17-50
17.2.8.5	FIFO Access Constraints	17-51
17.2.8.6	Handling of FIFO Transmit Control Information	17-52
17.3	Asynchronous Serial Channel (ASC = UART)	17-54
17.3.1	Signal Description	17-54
17.3.2	Frame Format	17-55
17.3.2.1	Idle Time	17-56
17.3.2.2	Start Bit Detection	17-57
17.3.2.3	Data Field	17-57
17.3.2.4	Parity Bit	17-57
17.3.2.5	Stop Bit(s)	17-57
17.3.3	Operating the ASC	17-58
17.3.3.1	Bit Timing	17-58
17.3.3.2	Baud Rate Generation	17-59
17.3.3.3	Noise Detection	17-60
17.3.3.4	Collision Detection	17-60

Table of Contents

17.3.3.5	Pulse Shaping	17-60
17.3.3.6	Automatic Shadow Mechanism	17-62
17.3.3.7	End of Frame Control	17-62
17.3.3.8	Mode Control Behavior	17-63
17.3.3.9	Disabling ASC Mode	17-63
17.3.3.10	Protocol Interrupt Events	17-63
17.3.3.11	Data Transfer Interrupt Handling	17-64
17.3.3.12	Baud Rate Generator Interrupt Handling	17-64
17.3.3.13	Protocol-Related Argument and Error	17-65
17.3.3.14	Receive Buffer Handling	17-65
17.3.3.15	Sync-Break Detection	17-65
17.3.3.16	Transfer Status Indication	17-65
17.3.4	ASC Protocol Registers	17-66
17.3.4.1	ASC Protocol Control Register	17-66
17.3.4.2	ASC Protocol Status Register	17-69
17.3.5	Hardware LIN Support	17-72
17.4	Synchronous Serial Channel (SSC)	17-74
17.4.1	Signal Description	17-74
17.4.1.1	Transmit and Receive Data Signals	17-76
17.4.1.2	Shift Clock Signals	17-77
17.4.1.3	Slave Select Signals	17-80
17.4.2	Operating the SSC	17-82
17.4.2.1	Automatic Shadow Mechanism	17-82
17.4.2.2	Mode Control Behavior	17-82
17.4.2.3	Disabling SSC Mode	17-83
17.4.2.4	Data Frame Control	17-83
17.4.2.5	Parity Mode	17-83
17.4.2.6	Transfer Mode	17-85
17.4.2.7	Data Transfer Interrupt Handling	17-85
17.4.2.8	Baud Rate Generator Interrupt Handling	17-86
17.4.2.9	Protocol-Related Argument and Error	17-86
17.4.2.10	Receive Buffer Handling	17-86
17.4.2.11	Multi-IO SSC Protocols	17-86
17.4.3	Operating the SSC in Master Mode	17-88
17.4.3.1	Baud Rate Generation	17-89
17.4.3.2	MSLS Generation	17-90
17.4.3.3	Automatic Slave Select Update	17-91
17.4.3.4	Slave Select Delay Generation	17-92
17.4.3.5	Protocol Interrupt Events	17-93
17.4.3.6	End-of-Frame Control	17-94
17.4.4	Operating the SSC in Slave Mode	17-96
17.4.4.1	Protocol Interrupts	17-96
17.4.4.2	End-of-Frame Control	17-97

Table of Contents

17.4.5	SSC Protocol Registers	17-98
17.4.5.1	SSC Protocol Control Registers	17-98
17.4.5.2	SSC Protocol Status Register	17-102
17.4.6	SSC Timing Considerations	17-104
17.4.6.1	Closed-loop Delay	17-104
17.4.6.2	Delay Compensation in Master Mode	17-107
17.4.6.3	Complete Closed-loop Delay Compensation	17-108
17.5	Inter-IC Bus Protocol (IIC)	17-109
17.5.1	Introduction	17-109
17.5.1.1	Signal Description	17-109
17.5.1.2	Symbols	17-110
17.5.1.3	Frame Format	17-111
17.5.2	Operating the IIC	17-112
17.5.2.1	Transmission Chain	17-113
17.5.2.2	Byte Stretching	17-113
17.5.2.3	Master Arbitration	17-113
17.5.2.4	Non-Acknowledge and Error Conditions	17-114
17.5.2.5	Mode Control Behavior	17-114
17.5.2.6	Data Transfer Interrupt Handling	17-114
17.5.2.7	IIC Protocol Interrupt Events	17-115
17.5.2.8	Baud Rate Generator Interrupt Handling	17-116
17.5.2.9	Receiver Address Acknowledge	17-116
17.5.2.10	Receiver Handling	17-117
17.5.2.11	Receiver Status Information	17-117
17.5.3	Symbol Timing	17-118
17.5.3.1	Start Symbol	17-119
17.5.3.2	Repeated Start Symbol	17-119
17.5.3.3	Stop Symbol	17-120
17.5.3.4	Data Bit Symbol	17-120
17.5.4	Data Flow Handling	17-121
17.5.4.1	Transmit Data Formats	17-121
17.5.4.2	Valid Master Transmit Data Formats	17-123
17.5.4.3	Master Transmit/Receive Modes	17-126
17.5.4.4	Slave Transmit/Receive Modes	17-128
17.5.5	IIC Protocol Registers	17-129
17.5.5.1	IIC Protocol Control Registers	17-129
17.5.5.2	IIC Protocol Status Register	17-132
17.6	Inter-IC Sound Bus Protocol (IIS)	17-135
17.6.1	Introduction	17-135
17.6.1.1	Signal Description	17-135
17.6.1.2	Protocol Overview	17-136
17.6.1.3	Transfer Delay	17-137
17.6.1.4	Connection of External Audio Components	17-137

Table of Contents

17.6.2	Operating the IIS	17-138
17.6.2.1	Frame Length and Word Length Configuration	17-138
17.6.2.2	Automatic Shadow Mechanism	17-139
17.6.2.3	Mode Control Behavior	17-139
17.6.2.4	Transfer Delay	17-139
17.6.2.5	Parity Mode	17-141
17.6.2.6	Transfer Mode	17-141
17.6.2.7	Data Transfer Interrupt Handling	17-141
17.6.2.8	Baud Rate Generator Interrupt Handling	17-142
17.6.2.9	Protocol-Related Argument and Error	17-142
17.6.2.10	Transmit Data Handling	17-142
17.6.2.11	Receive Buffer Handling	17-143
17.6.2.12	Loop-Delay Compensation	17-143
17.6.3	Operating the IIS in Master Mode	17-143
17.6.3.1	Baud Rate Generation	17-144
17.6.3.2	WA Generation	17-145
17.6.3.3	Master Clock Output	17-145
17.6.3.4	Protocol Interrupt Events	17-146
17.6.4	Operating the IIS in Slave Mode	17-146
17.6.4.1	Protocol Events and Interrupts	17-147
17.6.5	IIS Protocol Registers	17-147
17.6.5.1	IIS Protocol Control Registers	17-147
17.6.5.2	IIS Protocol Status Register	17-150
17.7	Service Request Generation	17-153
17.8	Debug Behaviour	17-153
17.9	Power, Reset and Clock	17-153
17.10	Initialization and System Dependencies	17-153
17.11	Registers	17-153
17.11.1	Address Map	17-157
17.11.2	Module Identification Registers	17-158
17.11.3	Channel Control and Configuration Registers	17-159
17.11.3.1	Channel Control Register	17-159
17.11.3.2	Channel Configuration Register	17-163
17.11.3.3	Kernel State Configuration Register	17-164
17.11.3.4	Interrupt Node Pointer Register	17-167
17.11.4	Protocol Related Registers	17-168
17.11.4.1	Protocol Control Registers	17-168
17.11.4.2	Protocol Status Register	17-169
17.11.4.3	Protocol Status Clear Register	17-170
17.11.5	Input Stage Register	17-171
17.11.5.1	Input Control Registers	17-171
17.11.6	Baud Rate Generator Registers	17-177
17.11.6.1	Fractional Divider Register	17-177

Table of Contents

17.11.6.2	Baud Rate Generator Register	17-178
17.11.6.3	Capture Mode Timer Register	17-181
17.11.7	Transfer Control and Status Registers	17-181
17.11.7.1	Shift Control Register	17-181
17.11.7.2	Transmission Control and Status Register	17-185
17.11.7.3	Flag Modification Registers	17-191
17.11.8	Data Buffer Registers	17-193
17.11.8.1	Transmit Buffer Locations	17-193
17.11.8.2	Receive Buffer Registers RBUF0, RBUF1	17-194
17.11.8.3	Receive Buffer Registers RBUF, RBUFD, RBUFSR	17-200
17.11.9	FIFO Buffer and Bypass Registers	17-204
17.11.9.1	Bypass Registers	17-204
17.11.9.2	General FIFO Buffer Control Registers	17-207
17.11.9.3	Transmit FIFO Buffer Control Registers	17-213
17.11.9.4	Receive FIFO Buffer Control Registers	17-217
17.11.9.5	FIFO Buffer Data Registers	17-222
17.11.9.6	FIFO Buffer Pointer Registers	17-225
17.12	Interconnects	17-226
17.12.1	USIC Module 0 Interconnects	17-227
17.12.2	USIC Module 1 Interconnects	17-235
18	Controller Area Network Controller (MultiCAN+)	18-1
18.1	CAN Basics	18-2
18.1.1	Addressing and Bus Arbitration	18-2
18.1.2	CAN Frame Types	18-3
18.1.2.1	Data Frames	18-3
18.1.2.2	Remote Frames	18-5
18.1.2.3	Error Frames	18-5
18.1.3	The Nominal Bit Time	18-6
18.1.4	Error Detection and Error Handling	18-7
18.2	Overview	18-10
18.2.1	Features List	18-11
18.3	MultiCAN+ Kernel Functional Description	18-13
18.3.1	Module Structure	18-13
18.3.2	Clock Control	18-15
18.3.3	Port Input Control	18-18
18.3.4	CAN Node Control	18-19
18.3.4.1	Bit Timing Unit	18-20
18.3.4.2	Bitstream Processor	18-21
18.3.4.3	Error Handling Unit	18-22
18.3.4.4	CAN Frame Counter	18-23
18.3.4.5	CAN Node Interrupts	18-23
18.3.5	Message Object List Structure	18-25

Table of Contents

18.3.5.1	Basics	18-25
18.3.5.2	List of Unallocated Elements	18-26
18.3.5.3	Connection to the CAN Nodes	18-26
18.3.5.4	List Command Panel	18-27
18.3.6	CAN Node Analyzer Mode	18-29
18.3.6.1	Analyzer Mode	18-30
18.3.6.2	Loop-Back Mode	18-30
18.3.6.3	Bit Timing Analysis	18-31
18.3.7	Message Acceptance Filtering	18-32
18.3.7.1	Receive Acceptance Filtering	18-32
18.3.7.2	Transmit Acceptance Filtering	18-34
18.3.8	Message Postprocessing	18-35
18.3.8.1	Message Object Interrupts	18-35
18.3.8.2	Pending Messages	18-37
18.3.9	Message Object Data Handling	18-39
18.3.9.1	Frame Reception	18-39
18.3.9.2	Frame Transmission	18-42
18.3.10	Message Object Functionality	18-45
18.3.10.1	Standard Message Object	18-45
18.3.10.2	Single Data Transfer Mode	18-45
18.3.10.3	Single Transmit Trial	18-45
18.3.10.4	Message Object FIFO Structure	18-46
18.3.10.5	Receive FIFO	18-48
18.3.10.6	Transmit FIFO	18-48
18.3.10.7	Gateway Mode	18-49
18.3.10.8	Foreign Remote Requests	18-52
18.4	Use Case Example MultiCAN+	18-53
18.5	MultiCAN+ Kernel Registers	18-57
18.5.1	Global Module Registers	18-61
18.5.2	CAN Node Registers	18-74
18.5.3	Message Object Registers	18-92
18.6	MultiCAN+ Module Implementation	18-113
18.6.1	Interfaces of the MultiCAN+ Module	18-113
18.6.2	MultiCAN+ Module External Registers	18-114
18.6.3	Module Clock Generation	18-115
18.6.3.1	Clock Selection	18-115
18.6.3.2	Fractional Divider	18-115
18.6.4	Port and I/O Line Control	18-119
18.6.4.1	Input/Output Function Selection in Ports	18-119
18.6.4.2	Node Receive Input Selection	18-119
18.6.5	Connections with other modules	18-120
18.6.5.1	Connections to Interrupt Router Inputs	18-120
18.6.5.2	Connections to Capcom4 Inputs	18-120

Table of Contents

18.6.5.3	Connections to USIC Inputs	18-121
18.6.6	Interrupt Control	18-122
18.6.7	MultiCAN+ Module Register Address Map	18-124
19	Versatile Analog-to-Digital Converter (VADC)	19-1
19.1	Overview	19-1
19.2	Introduction and Basic Structure	19-4
19.3	Configuration of General Functions	19-9
19.3.1	General Clocking Scheme and Control	19-9
19.3.2	Priority Channel Assignment	19-10
19.4	Module Activation and Power Saving	19-10
19.5	Conversion Request Generation	19-11
19.5.1	Queued Request Source Handling	19-13
19.5.2	Channel Scan Request Source Handling	19-16
19.6	Request Source Arbitration	19-20
19.6.1	Arbiter Operation and Configuration	19-21
19.6.2	Conversion Start Mode	19-22
19.7	Analog Input Channel Configuration	19-24
19.7.1	Channel Parameters	19-24
19.7.2	Conversion Timing	19-26
19.7.3	Alias Feature	19-27
19.7.4	Conversion Modes	19-28
19.7.5	Compare with Standard Conversions (Limit Checking)	19-29
19.7.6	Utilizing Fast Compare Mode	19-31
19.7.7	Boundary Flag Control	19-32
19.8	Conversion Result Handling	19-34
19.8.1	Storage of Conversion Results	19-34
19.8.2	Data Alignment	19-36
19.8.3	Wait-for-Read Mode	19-37
19.8.4	Result FIFO Buffer	19-38
19.8.5	Result Event Generation	19-39
19.8.6	Data Modification	19-40
19.9	Synchronization of Conversions	19-47
19.9.1	Synchronized Conversions for Parallel Sampling	19-47
19.9.2	Equidistant Sampling	19-50
19.10	Safety Features	19-51
19.10.1	Broken Wire Detection	19-51
19.10.2	Signal Path Test Modes	19-52
19.10.3	Configuration of Test Functions	19-53
19.11	External Multiplexer Control	19-54
19.12	Service Request Generation	19-56
19.13	Registers	19-58
19.13.1	Module Identification	19-61

Table of Contents

19.13.2	System Registers	19-63
19.13.3	General Registers	19-66
19.13.4	Arbitration and Source Registers	19-68
19.13.5	Channel Control Registers	19-96
19.13.6	Result Registers	19-101
19.13.7	Miscellaneous Registers	19-109
19.13.8	Service Request Registers	19-120
19.14	Interconnects	19-132
19.14.1	Product-Specific Configuration	19-132
19.14.2	Analog Module Connections in the XMC4300	19-134
19.14.3	Digital Module Connections in the XMC4300	19-135
20	Digital to Analog Converter (DAC)	20-1
20.1	Overview	20-1
20.1.1	Features	20-1
20.1.2	Block Diagram	20-2
20.2	Operating Modes	20-3
20.2.1	Hardware features	20-3
20.2.1.1	Trigger Generators (TG)	20-3
20.2.1.2	Data FIFO buffer (FIFO)	20-4
20.2.1.3	Data output stage	20-5
20.2.1.4	Pattern Generators (PG) - Waveform Generator	20-6
20.2.1.5	Noise Generators (NG) - Pseudo Random Number Generator	20-7
20.2.1.6	Ramp Generators (RG)	20-7
20.2.2	Entering any Operating Mode	20-8
20.2.3	Single Value Mode	20-8
20.2.4	Data Processing Mode	20-9
20.2.4.1	FIFO Data Handling	20-9
20.2.5	Pattern Generation Mode	20-10
20.2.6	Noise Generation Mode	20-11
20.2.7	Ramp Generation Mode	20-12
20.3	Service Request Generation	20-12
20.4	Power, Reset and Clock	20-13
20.5	Initialization	20-13
20.6	Registers	20-15
20.6.1	Address Map	20-15
20.6.2	Register Overview	20-15
20.6.3	Register Description	20-16
20.6.3.1	DAC_ID Register	20-16
20.6.3.2	DAC Configuration Registers	20-17
20.6.3.3	DAC Data Registers	20-24
20.6.3.4	DAC Pattern Registers	20-26
20.7	Interconnects	20-28

Table of Contents

20.7.1	Analog Connections	20-28
20.7.2	Digital Connections	20-29
20.7.2.1	Service Request Connections	20-29
20.7.2.2	Trigger Connections	20-29
20.7.2.3	Synchronization Interface of the Pattern Generator	20-30
21	Capture/Compare Unit 4 (CCU4)	21-1
21.1	Overview	21-1
21.1.1	Features	21-2
21.1.2	Block Diagram	21-4
21.2	Functional Description	21-6
21.2.1	CC4y Overview	21-6
21.2.2	Input Selector	21-8
21.2.3	Connection Matrix	21-10
21.2.4	Starting/Stopping the Timer	21-12
21.2.5	Counting Modes	21-13
21.2.5.1	Calculating the PWM Period and Duty Cycle	21-14
21.2.5.2	Updating the Period and Duty Cycle	21-15
21.2.5.3	Edge Aligned Mode	21-19
21.2.5.4	Center Aligned Mode	21-20
21.2.5.5	Single Shot Mode	21-21
21.2.6	Active/Passive Rules	21-22
21.2.7	External Events Control	21-22
21.2.7.1	External Start/Stop	21-23
21.2.7.2	External Counting Direction	21-25
21.2.7.3	External Gating Signal	21-27
21.2.7.4	External Count Signal	21-27
21.2.7.5	External Load	21-28
21.2.7.6	External Capture	21-29
21.2.7.7	Capture Extended Read Back Mode	21-35
21.2.7.8	External Modulation	21-38
21.2.7.9	TRAP Function	21-41
21.2.7.10	Status Bit Override	21-43
21.2.8	Multi-Channel Control	21-44
21.2.9	Timer Concatenation	21-47
21.2.10	PWM Dithering	21-52
21.2.11	Prescaler	21-57
21.2.11.1	Normal Prescaler Mode	21-58
21.2.11.2	Floating Prescaler Mode	21-58
21.2.12	CCU4 Usage	21-60
21.2.12.1	PWM Signal Generation	21-60
21.2.12.2	Prescaler Usage	21-62
21.2.12.3	PWM Dither	21-64

Table of Contents

21.2.12.4	Capture Mode Usage	21-67
21.3	Service Request Generation	21-73
21.4	Debug Behavior	21-77
21.5	Power, Reset and Clock	21-77
21.5.1	Clocks	21-77
21.5.2	Module Reset	21-78
21.5.3	Power	21-79
21.6	Initialization and System Dependencies	21-79
21.6.1	Initialization Sequence	21-79
21.6.2	System Dependencies	21-79
21.7	Registers	21-81
21.7.1	Global Registers	21-87
21.7.2	Slice (CC4y) Registers	21-102
21.8	Interconnects	21-138
21.8.1	CCU40 pins	21-138
21.8.2	CCU41 pins	21-143
22	Capture/Compare Unit 8 (CCU8)	22-1
22.1	Overview	22-1
22.1.1	Features	22-2
22.1.2	Block Diagram	22-5
22.2	Functional Description	22-7
22.2.1	Overview	22-7
22.2.2	Input Selector	22-10
22.2.3	Connection Matrix	22-12
22.2.4	Start/Stop Control	22-14
22.2.5	Counting Modes	22-15
22.2.5.1	Calculating the PWM Period and Duty Cycle	22-16
22.2.5.2	Updating the Period and Duty Cycle	22-17
22.2.5.3	Edge Aligned Mode	22-24
22.2.5.4	Center Aligned Mode	22-24
22.2.5.5	Single Shot Mode	22-25
22.2.6	Active/Passive Rules	22-26
22.2.7	Compare Modes	22-27
22.2.7.1	Edge Aligned Compare Modes	22-31
22.2.7.2	Center Aligned Compare Modes	22-35
22.2.8	External Events Control	22-38
22.2.8.1	External Start/Stop	22-38
22.2.8.2	External Counting Direction	22-41
22.2.8.3	External Gating Signal	22-42
22.2.8.4	External Count Signal	22-43
22.2.8.5	External Load	22-44
22.2.8.6	External Capture	22-45

Table of Contents

22.2.8.7	Capture Extended Read Back Mode	22-51
22.2.8.8	External Modulation	22-54
22.2.8.9	Trap Function	22-56
22.2.8.10	Status Bit Override	22-59
22.2.9	Multi-Channel Support	22-60
22.2.10	Timer Concatenation	22-65
22.2.11	Output Parity Checker	22-70
22.2.12	PWM Dithering	22-74
22.2.13	Prescaler	22-78
22.2.13.1	Normal Prescaler Mode	22-79
22.2.13.2	Floating Prescaler Mode	22-79
22.2.14	CCU8 Usage	22-81
22.2.14.1	PWM Signal Generation	22-81
22.2.14.2	Prescaler Usage	22-83
22.2.14.3	PWM Dither	22-86
22.2.14.4	Capture Mode Usage	22-88
22.2.14.5	Parity Checker Usage	22-94
22.3	Service Request Generation	22-97
22.4	Debug Behavior	22-100
22.5	Power, Reset and Clock	22-100
22.5.1	Clocks	22-101
22.5.2	Module Reset	22-101
22.5.3	Power	22-102
22.6	Initialization and System Dependencies	22-102
22.6.1	Initialization Sequence	22-102
22.6.2	System Dependencies	22-103
22.7	Registers	22-104
22.7.1	Global Registers	22-112
22.7.2	Slice (CC8y) Registers	22-132
22.8	Interconnects	22-179
22.8.1	CCU80 Pins	22-179
23	General Purpose I/O Ports (PORTS)	23-1
23.1	Overview	23-1
23.1.1	Features	23-1
23.1.2	Block Diagram	23-2
23.1.3	Definition of Terms	23-3
23.2	GPIO and Alternate Function	23-4
23.2.1	Input Operation	23-4
23.2.2	Output Operation	23-5
23.3	Hardware Controlled I/Os	23-6
23.4	Power Saving Mode Operation	23-7
23.5	Analog Ports	23-8

Table of Contents

23.6	Power, Reset and Clock	23-9
23.7	Initialization and System Dependencies	23-10
23.8	Registers	23-11
23.8.1	Port Input/Output Control Registers	23-14
23.8.2	Pad Driver Mode Register	23-18
23.8.3	Pin Function Decision Control Register	23-22
23.8.4	Port Output Register	23-24
23.8.5	Port Output Modification Register	23-25
23.8.6	Port Input Register	23-26
23.8.7	Port Pin Power Save Register	23-27
23.8.8	Port Pin Hardware Select Register	23-28
23.9	Package Pin Summary	23-30
23.10	Port I/O Functions	23-35
23.10.1	Port I/O Function Table	23-37
24	Startup modes	24-1
24.1	Overview	24-1
24.1.1	Features	24-1
24.2	Boot and Startup	24-3
24.2.1	Initial boot sequence	24-3
24.2.2	Reset types and corresponding boot modes	24-4
24.2.2.1	Power On Reset (PORST)	24-4
24.2.2.2	System Reset	24-5
24.2.3	Boot mode selection	24-5
24.2.4	Unique Chip ID and Device Information Data	24-7
24.3	Boot mode details	24-9
24.3.1	Normal boot mode	24-9
24.3.2	Boot from PSRAM	24-9
24.3.3	Alternative boot mode - Address0 (ABM-0)	24-11
24.3.4	Alternative boot mode - Address1 (ABM-1)	24-13
24.3.5	Fallback ABM	24-13
24.3.6	ASC BSL mode	24-13
24.3.7	CAN BSL mode	24-16
24.3.8	Boot Mode Index (BMI)	24-19
24.4	Debug behavior	24-23
24.4.1	Boot modes and hardware debugger support	24-23
24.4.2	Failures and handling	24-24
24.5	Power, Reset and Clock	24-26
25	Debug and Trace System (DBG)	25-1
25.1	Overview	25-1
25.2	Debug System Operation	25-4
25.2.1	Flash Patch Breakpoint (FPB)	25-4
25.2.2	Data Watchpoint and Trace (DWT)	25-4

Table of Contents

25.2.3	Instrumentation Trace Macrocell (ITM)	25-4
25.2.4	Embedded Trace Macrocell (ETM)	25-4
25.2.5	Trace Port Interface Unit (TPIU)	25-5
25.3	Power, Reset and Clock	25-5
25.3.1	Reset	25-5
25.3.1.1	CoreSight™ resets	25-5
25.3.1.2	Serial Wire interface driven system reset	25-6
25.4	Initialization and System Dependencies	25-6
25.4.1	Debug accesses and Flash protection	25-6
25.4.2	Halt after reset	25-6
25.4.3	Halting Debug and Peripheral Suspend	25-9
25.4.4	Timestamping	25-11
25.4.5	Debug tool interface access (SWJ-DP)	25-11
25.4.5.1	Switch from JTAG to SWD	25-11
25.4.5.2	Switch from SWD to JTAG	25-11
25.4.6	ID Codes	25-11
25.4.7	ROM Table	25-12
25.4.8	JTAG debug port	25-12
25.5	Debug System Registers	25-14
25.6	Debug and Trace Signals	25-14
25.6.1	Internal pull-up and pull-down on JTAG pins	25-15
25.6.2	Debug Connector	25-16

About this Document

This Reference Manual is addressed to embedded hardware and software developers. It provides the reader with detailed descriptions about the behavior of the XMC4300 series functional units and their interaction.

The manual describes the functionality of the superset device of the XMC4300 microcontroller series. For the available functionality (features) of a specific XMC4300 derivative (derivative device), please refer to the respective Data Sheet. For simplicity, the various device types are referenced by the collective term XMC4300 throughout this manual.

XMC4000 Family User Documentation

The set of user documentation includes:

- **Reference Manual**
 - describes the functionality of the superset device.
- **Data Sheets**
 - list the complete ordering information, available features and electrical characteristics of derivative devices.
- **Errata Sheets**
 - list deviations from the specifications given in the related Reference Manual or Data Sheets. Errata Sheets are provided for the superset of devices.

Attention: Please consult all parts of the documentation set to attain consolidated knowledge about your device.

Application related guidance is provided by **Users Guides** and **Application Notes**.

Please refer to <http://www.infineon.com/xmc4000> to get access to the latest versions of those documents.

Related Documentation

The following documents are referenced:

- ARM® Cortex®-M4
 - Technical Reference Manual
 - User Guide, Reference Material
- ARM®v7-M Architecture Reference Manual
- AMBA® 3 AHB-Lite Protocol Specification

Copyright Notice

- Portions of SDMMC chapter Copyright © 2010 by Arasan Chip Systems, Inc. All rights reserved. Used with permission.
- Portions of CPU chapter Copyright © 2009, 2010 by ARM, Ltd. All rights reserved. Used with permission.

- Portions of ETH, USB and GPDMA chapter Copyright © 2009, 2010 by Synopsys, Inc. All rights reserved. Used with permission.

Text Conventions

This document uses the following naming conventions:

- Functional units of the device are given in plain UPPER CASE. For example: “The USIC0 unit supports...”.
- Pins using negative logic are indicated by an overline. For example: “The $\overline{\text{WAIT}}$ input has...”.
- Bit fields and bits in registers are generally referenced as “Module_RegisterName.BitField” or “Module_RegisterName.Bit”. For example: “The USIC0_PCR.MCLK bit enables the...”. Most of the register names contain a module name prefix, separated by an underscore character “_” from the actual register name (for example, “USIC0_PCR”, where “USIC0” is the module name prefix, and “PCR” is the kernel register name). In chapters describing the kernels of the peripheral modules, the registers are mainly referenced with their kernel register names. The peripheral module implementation sections mainly refer to the actual register names with module prefixes.
- Variables used to describe sets of processing units or registers appear in mixed upper and lower cases. For example, register name “MOFCRn” refers to multiple “MOFCR” registers with variable n. The bounds of the variables are always given where the register expression is first used (for example, “n = 0-31”), and are repeated as needed in the rest of the text.
- The default radix is decimal. Hexadecimal constants are suffixed with a subscript letter “H”, as in 100_H. Binary constants are suffixed with a subscript letter “B”, as in: 111_B.
- When the extent of register fields, groups register bits, or groups of pins are collectively named in the body of the document, they are represented as “NAME[A:B]”, which defines a range for the named group from B to A. Individual bits, signals, or pins are given as “NAME[C]” where the range of the variable C is given in the text. For example: CFG[2:0] and SRPN[0].
- Units are abbreviated as follows:
 - **MHz** = Megahertz
 - **μs** = Microseconds
 - **kBaud, kbit/s** = 1000 characters/bits per second
 - **MBaud, Mbit/s, Mbps** = 1,000,000 characters/bits per second
 - **Kbyte, KB** = 1024 bytes of memory
 - **Mbyte, MB** = 1048576 bytes of memoryIn general, the k prefix scales a unit by 1000 whereas the K prefix scales a unit by 1024. Hence, the Kbyte unit scales the expression preceding it by 1024. The kBaud unit scales the expression preceding it by 1000. The M prefix scales by 1,000,000 or 1048576. For example, 1 Kbyte is 1024 bytes, 1 Mbyte is

About this Document

1024 × 1024 bytes, 1 kBaud/kbit are 1000 characters/bits per second, 1 MBaud/Mbit are 1000000 characters/bits per second, and 1 MHz is 1,000,000 Hz.

- Data format quantities are defined as follows:
 - **Byte** = 8-bit quantity
 - **Half-word** = 16-bit quantity
 - **Word** = 32-bit quantity
 - **Double-word** = 64-bit quantity

Bit Function Terminology

In tables where register bits or bit fields are defined, the following conventions are used to indicate the access types.

Table 1 Bit Function Terminology

Bit Function	Description
rw	The bit or bit field can be read and written.
rwh	As rw, but bit or bit field can be also set or reset by hardware. If not otherwise documented the software takes priority in case of a write conflict between software and hardware.
r	The bit or bit field can only be read (read-only).
w	The bit or bit field can only be written (write-only). A read to this register will always give a default value back.
rh	This bit or bit field can be modified by hardware (read-hardware, typical example: status flags). A read of this bit or bit field give the actual status of this bit or bit field back. Writing to this bit or bit field has no effect to the setting of this bit or bit field.

Register Access Modes

Read and write access to registers and memory locations are sometimes restricted. In memory and register access tables, the following terms are used.

Table 2 Register Access Modes

Symbol	Description
U	Access permitted when software executes on Unprivileged level.
PV	Access permitted when software executes on Privileged level.
32	Only 32-bit word accesses are permitted to this register/address range.
NC	No change, indicated register is not changed.

Table 2 Register Access Modes (cont'd)

Symbol	Description
BE	Indicates that an access to this address range generates a Bus Error.
nBE	Indicates that no Bus Error is generated when accessing this address range.

Reserved Bits

Register bit fields named **Reserved** or **0** indicate unimplemented functions with the following behavior:

- Reading these bit fields returns 0.
- These bit fields should be written with 0 if the bit field is defined as r or rh.
- These bit fields must to be written with 0 if the bit field is defined as rw.

Abbreviations and Acronyms

The following acronyms and terms are used in this document:

ADC	Analog-to-Digital Converter
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
ASC	Asynchronous Serial Channel
BMI	Boot Mode Index
BROM	Boot ROM
CAN	Controller Area Network
CMSIS	Cortex Microcontroller Software Interface Standard
CPU	Central Processing Unit
CRC	Cyclic Redundancy Code
CCU4	Capture Compare Unit 4
CCU8	Capture Compare Unit 8
DAC	Digital to Analog Converter
DSD	Delta Sigma Demodulator
DSRAM	Data SRAM
DMA	Direct Memory Access
EBU	External Bus Interface
ECC	Error Correction Code

ERU	Event Request Unit
ETH	Ethernet Unit
FCE	Flexible CRC Engine
FCS	Flash Command State Machine
FIM	Flash Interface and Control Module
FPU	Floating Point Unit
GPDMA	General Purpose Direct Memory Access
GPIO	General Purpose Input/Output
HMI	Human-Machine Interface
HRPWM	High Resolution PWM
IIC	Inter Integrated Circuit (also known as I ² C)
IIS	Inter-IC Sound Interface
I/O	Input / Output
JTAG	Joint Test Action Group = IEEE1149.1
LED	Light Emitting Diode
LEDTS	LED and Touch Sense (Control Unit)
LIN	Local Interconnect Network
MPU	Memory Protection Unit
MSB	Most Significant Bit
NC	Not Connected
NMI	Non-Maskable Interrupt
NVIC	Nested Vectored Interrupt Controller
OCDS	On-Chip Debug Support
OTP	One Time Programmable
PBA	Peripheral Bridge AHB to AHB
PFLASH	Program Flash Memory
PLL	Phase Locked Loop
PMU	Program Memory Unit
POSIF	Position Interface
PWM	Pulse Width Modulation
PSRAM	Program SRAM
RAM	Random Access Memory

About this Document

RTC	Real Time Clock
SCU	System Control Unit
SDMMC	Secure Digital / Multi Media Card (Interface)
SDRAM	Synchronous Dynamic Random Access Memory
SFR	Special Function Register
SPI	Serial Peripheral Interface
SRAM	Static RAM
SR	Service Request
SSC	Synchronous Serial Channel
SSW	Startup Software
UART	Universal Asynchronous Receiver Transmitter
UCB	User Configuration Block
USB	Universal Serial Bus
USIC	Universal Serial Interface Channel
WDT	Watchdog Timer

Introduction

1 Introduction

The XMC4300 devices are members of the XMC4000 family of microcontrollers based on the ARM Cortex-M4 processor core. The XMC4000 is a family of high performance and energy efficient microcontrollers optimized for Industrial Connectivity, Industrial Control, Power Conversion, Sense & Control.

The growing complexity of today's energy efficient embedded control applications are demanding microcontroller solutions with higher performance CPUs along with DSC (Digital Signal Control) and FPU (Floating Point Unit) capabilities to handle complex control algorithms and integrated peripherals that are optimized for performance. Combined with a development environment designed to shorten product development time and increase productivity, the XMC4300 family of microcontrollers take advantage of Infineon's decades of experience in the industrial market to provide an optimized solution to meet the performance challenges of today's embedded control applications.

1.1 Architectural Overview

The XMC4300 combines the extended functionality and performance of the ARM Cortex-M4 core with powerful on-chip peripheral subsystems and on-chip memory units. The following key features are available in the XMC4300 devices:

- CPU Subsystem
 - High Performance 32-bit ARM Cortex-M4 CPU
 - 16-bit and 32-bit Thumb2 instruction set
 - DSP/MAC instructions
 - Floating Point Unit
 - Memory Protection Unit
 - Low power consumption
 - Low latency interrupt processing
 - Nested Vectored Interrupt Controller
 - 3 Bus interfaces
 - Little Endian
 - On-chip debug support
 - System timer (SysTick) for Operating System support
- Integrated On-Chip Memories
 - 16 KB on-chip boot ROM
 - 64 KB on-chip high-speed program memory
 - 64 KB on-chip high speed data memory
 - 256 on-chip Flash Memory with 8KB instruction cache
- Serial Interfaces
 - Two Universal Serial Interface Channel Units (USIC), providing UART, double-SPI, quad-SPI, IIC, IIS and LIN interfaces
 - Controller Area Network interface (MultiCAN), two nodes, 64 Message Objects, Full-CAN/Basic-CAN

Introduction

- Universal Serial Bus(USB), USB 2.0 host, Full-Speed,OTG, with internal PHY
- Ethernet MAC module (ETH0),capable of 10/100 Mbit/s transfer rates , (RMII only)
- EtherCat MAC module (ECAT0), capable of 100 Mbit/s transfer rates with 2 MII ports, 8 Fieldbus Memory Management Units (FMMU), 8 Sync Managers, supporting different external PHYs, enables a host to transmit and receive data over Ethernet, IEEE 802.3-2002 compliant
- External Memory Interfaces
 - Multi-Media Card and SD interface (SDMMC) for data storage memory cards
- On-Chip Peripheral Subsystems
 - General Purpose DMA up-to 8 channels
 - One Capture/Compare Units 8 (CCU8) for Motor Control and Power Conversion.
 - Two Capture/Compare Units 4 (CCU4) for use as general purpose timers
 - Die Temperature Sensor (DTS)
 - Window Watchdog Timer (WDT)for safety sensitive applications
 - Flexible CRC Engine (FCE) for multiple bit error detection
 - Real Time Clock module with alarm support
 - LED and Touch-Sense Controller (LEDTS) for Human-Machine interface
 - System Control Unit for system configuration and control
- Analog Frontend
 - Digital-Analogue Converter (DAC) with two channels of 12-bit resolution
 - TwoVersatile Analog-Digital Converters (VADC) of 12-bit resolution, 8 channels each
- On-Chip Debug Support
 - standard ARM-JTAG support with boundary scan board level support
 - dual wire I/O source level debugging with up to 8 breakpoints on code and data
 - one wire serial output for instrumentation data trace
 - Flash Patch and Breakpoint Unit
- Input/Output Lines
 - Programmable port driver control module (PORTS)
 - Individual Bit Addressability
 - Tri-stated in input mode
 - Push/pull or open drain output mode
- Packages
 - PG-LQFP-100

1.1.1 Block Diagram

The diagram below shows the functional blocks and their basic connectivity within the XMC4300 System.

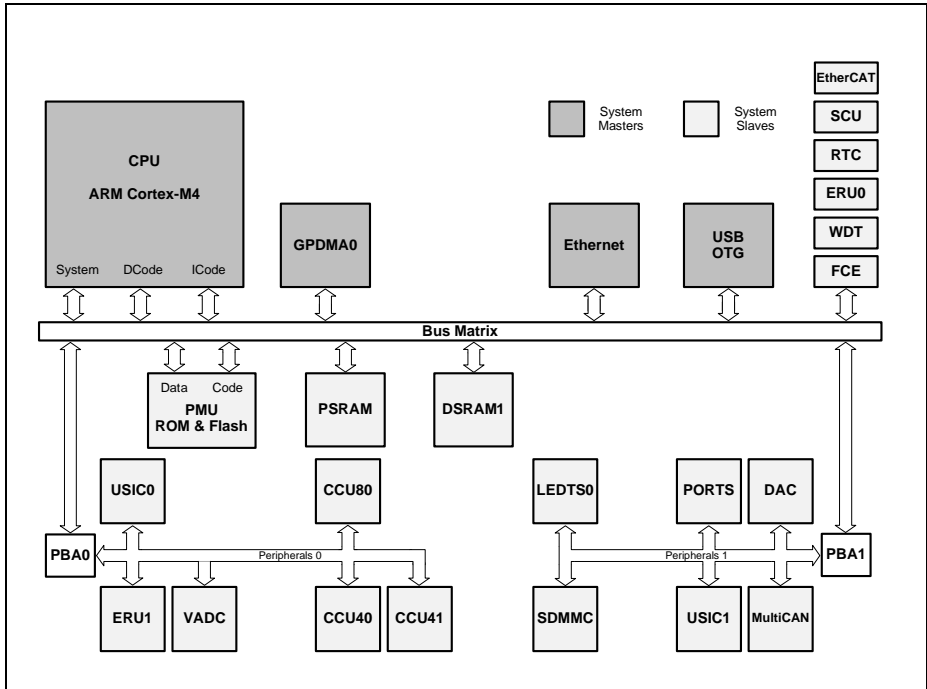


Figure 1-1 System Block Diagram

1.2 System Core Components

The XMC4300 system core consists of the CPU (including FPU and MPU) and the memory interface blocks for program and data memories - PMU.

Central Processing Unit (CPU)

The Cortex-M4 processor is built on a high-performance processor core with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and a design optimized for energy efficient control applications. To address the growing complexity of embedded control it also includes a IEEE754-compliant single-precision

Introduction

floating-point computation and a range of single-cycle/SIMD multiplication and multiply-with-accumulate capabilities, saturating arithmetic and dedicated hardware division.

To facilitate the design of cost-sensitive devices, the Cortex-M4 processor implements tightly-coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities.

To ensure high code density and reduced program memory requirements the processor also implements a version of the Thumb® instruction set based on Thumb-2 technology. The instruction set provides the exceptional performance expected of a modern 32-bit architecture with the high code density of 8-bit and 16-bit microcontrollers.

Floating Point Unit (FPU)

The Floating-point unit (FPU) provides IEEE754-compliant operations on single-precision, 32-bit, floating-point values.

Memory Protection Unit (MPU)

The MPU improves system reliability by defining the memory attributes for different memory regions. It provides fine grain memory control, enabling applications to utilize multiple privilege levels, separating and protecting code, data and stack on a task-by-task basis. Up to eight different regions are supported as well as an optional predefined background region. These features are becoming critical to support safety requirements in many Industrial embedded applications.

Programmable Multiple Priority Interrupt System (NVIC)

The XMC4300 provides separate interrupt nodes that may be assigned to 64 priority levels. Most interrupt sources are connected to a dedicated interrupt node. In some cases, multi-source interrupt nodes are incorporated for efficient use of system resources. These nodes can be activated by several source requests and are controlled via interrupt subnode control registers.

Direct Memory Access (GPDMA)

The GPDMA is a highly configurable DMA controller that allows high-speed data transfers between peripherals and memory, memory to memory and peripheral to peripheral. Complex data transfers can be done with minimal intervention of the processor, keeping the CPU resources free for other operations.

Flexible CRC Engine (FCE)

The FCE provides a parallel implementation of Cyclic Redundancy Code (CRC) algorithms. The FCE implements the IEEE 802.3 ethernet CRC32, the CCITT CRC16 and the SAE J1850 CRC8 polynomials. FCE's generic structure enables it to be extended with multiple independent CRC polynomials. The primary target of FCE is to

be used as a hardware acceleration engine for software applications or operating systems services using CRC signatures.

1.3 On-Chip System Resources

The XMC4300 controllers provide a number of system resources designed around the CPU. The combination of CPU and these resources results in the high performance of the members of this controller family.

The on-chip memories provide zero-waitstate accesses to code and data. The memories can be accessed concurrently from various system masters.

1.3.1 Memories

Various types of dedicated memories are available on-chip. The suggested use of the memories aims to improve performance and system stability in most typical application cases. However, the user has the flexibility to use the memories in any other way in order to fulfill application specific requirements.

Boot ROM (BROM)

The Boot ROM memory contains boot code and exception vector table. The basic system initialization sequence code, also referred to as firmware, is executed immediately after reset release.

Flash memory

The Flash is for non volatile code or constant data storage. The Flash module implements protected and efficient writing algorithms for programming and erasing as well as dynamic error correction providing high read data security for all read accesses.

Code RAM (PSRAM)

The Code RAM is intended for user code or Operating System data storage. The memory is accessed via the Bus Matrix and provides zero-wait-state access for the CPU for code execution or data access.

System RAM (DSRAM1)

The System RAM is intended for general user data storage. The System RAM is accessed via the Bus Matrix and provides zero-wait-state access for data.

1.4 Bus System

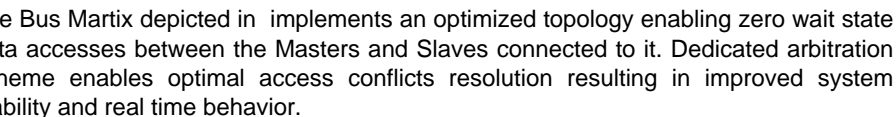
The XMC4300 is targeted for use in embedded systems. Therefore the key features are timing determinism and low latency on real time events. Bus bandwidth is required particularly for communication peripherals.

The bus system will therefore provide:

- Timing Determinism
- Low Latency
- Performance
- Throughput

1.4.1 Bus Matrix

The central part of the bus system is built up around a multilayer AHB-lite compliant matrix. By means of this technique the bus masters and bus slaves can be connected in a flexible way while maintaining high bus performance.

The Bus Matrix depicted in  implements an optimized topology enabling zero wait state data accesses between the Masters and Slaves connected to it. Dedicated arbitration scheme enables optimal access conflicts resolution resulting in improved system stability and real time behavior.

1.4.2 Bus Interfaces

This chapter describes the features for the two kinds of interfaces.

- **Memory Interface**
- **Peripheral Interface**

All on-chip peripherals and memories are attached to the Bus Matrix, in some cases via peripheral bridges. All on-chip modules implement Little Endian data organization. The following types of transfer are supported:

- Locked Transfers
- Burst Operation
- Protection Control

Pipelining is also supported for bandwidth critical transfers.

Memory Interface

The on-chip memories capable to accept a transfer request with each bus clock cycle.

The memory interface data bus width is 32-bit. Each memory slave support 32-bit, 16-bit and 8-bit access types.

Peripheral Interface

Each slave supports 32-bit accesses. Some slaves also support 8-bit and/or 16-bit accesses.

1.5 Communication

Communication features are key requirements in today's industrial systems. The XMC4300 offers units to realize Ethernet or CAN network or USB access. Further various memory interfaces and a unit to implement a Human Machine Interface (HMI) are provided.

LED and Touch Sense (LEDTS)

The LEDTS drives LEDs and controls touch pads used as human-machine interface (HMI) in an application. The LEDTS can measure the capacitance of up to 8 touch pads using the relaxation oscillator (RO) topology. The module can also drive up to 64 LEDs in an LED matrix. Touch pads and LEDs can share pins to minimize the number of pins needed for such applications.

SD/MMC interface (SDMMC)

The Secure Digital/ MultiMediaCard interface (SDMMC) provides an interface between SD/SDIO/MMC cards and the AHB bus. The CPU is programmed to support SD, SDIO, SDHC and MMC cards, and can operate up to 48 MHz. The SDMMC module is able to transfer a maximum of 24 MB/sec for SD cards and 48 MB/sec for MMC cards.

The SDMMC Host Controller handles SDIO/SD protocol at transmission level, packing data, adding cyclic redundancy check (CRC), start/end bit, and checking for transaction format correctness. Some useful applications of the SDMMC include memory extension, data logging, and firmware update.

Ethernet MAC (ETH)

The Ethernet MAC (ETH) is a major communication peripheral that supports 10/100 MBit/s data transfer rates in compliance with the IEEE 802.3-2002 standard.

The ETH may be used to implement Internet connected applications using IPv4 and IPv6. The ETH also includes support for IEEE1588 time synchronisation to allow implementation of Real Time Ethernet protocols.

Universal Serial Bus (USB)

The USB module is a Dual-Role Device (DRD) controller that supports both device and host functions and complies fully with the On-The-Go Supplement to the USB 2.0 Specification, Revision 1.3. It can also be configured as a host-only or device-only controller, fully compliant with the USB 2.0 Specification.

The USB core's USB 2.0 configurations support full-speed (12-Mbps) transfers.

The USB core is optimized for the following applications and systems:

- Portable electronic devices
- Point-to-point applications (direct connection to FS device)

Universal Serial Interface Channel (USIC)

The USIC is a flexible interface module covering several serial communication protocols. A USIC module contains two independent communication channels. The user can program during run-time which protocol will be handled by each communication channel and which pins are used.

Controller Area Network (CAN)

The MultiCAN module contains independently operating CAN nodes with Full-CAN functionality that are able to exchange Data and Remote Frames via a gateway function. Transmission and reception of CAN frames is handled in accordance with CAN specification V2.0 B (active). Each CAN node can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers.

All CAN nodes share a common set of message objects. Each message object can be individually allocated to one of the CAN nodes. Besides serving as a storage container for incoming and outgoing frames, message objects can be combined to build gateways between the CAN nodes or to setup a FIFO buffer.

1.6 Analog Frontend

XMC4300's interfaces from and to the analog world.

Versatile Analog-to Digital Convertor (VDAC)

The 12-bit VADC is a set of converter blocks that can be operated either independently or via a common request source that emulates a background converter. Each converter block is equipped with a dedicated input multiplexer and dedicated request sources, which together build separate groups.

This basic structure supports application-oriented programming and operating while still providing general access to all resources. The almost identical converter groups allow a flexible assignment of functions to channels.

Digital to Analog Convertor (DAC)

The module consists of two separate 12-bit digital to analog converters (DACs). It converts two digital input signals into two analog voltage signal outputs at a maximum conversion rate of 1MHz. The chosen design structure is based on a current steering architecture with internal reference generation and provides buffered voltage outputs. In

order to reduce power consumption during inactive periods, a power down mode is available.

1.7 Industrial Control

Core components needed for motion and motor control, power conversion and other time based applications.

Capture/Compare Unit 4 (CCU4)

The CCU4 peripheral is a major component for systems that need general purpose timers for signal monitoring/conditioning and Pulse Width Modulation (PWM) signal generation. Power electronic control systems like switched mode power supplies or uninterruptible power supplies can easily be implemented with the functions inside the CCU4 peripheral.

The internal modularity of CCU4, translates into a software friendly system for fast code development and portability between applications

Capture/Compare Unit 8 (CCU8)

The CCU8 peripheral functions play a major role in applications that need complex Pulse Width Modulation (PWM) signal generation, with complementary high side and low side switches, multi phase control or output parity checking. The CCU8 is optimized for state of the art motor control, multi phase and multi level power electronics systems.

The internal modularity of CCU8, translates into a software friendly system for fast code development and portability between applications.

1.8 On-Chip Debug Support

The On-Chip Debug Support system based on the ARM CoreSight™ provides a broad range of debug and emulation features built into the XMC4300. The user software running on the XMC4300 can thus be debugged within the target system environment.

The On-Chip Debug Support is controlled by an external debugging device via the debug interface and an optional break interface. The debugger controls the On-Chip Debug Support via a set of dedicated registers accessible via the debug interface. Additionally, the On-Chip Debug Support system can be controlled by the CPU, e.g. by a monitor program.

CPU Subsystem

2 Central Processing Unit (CPU)

The XMC4300 features the ARM Cortex-M4 processor. A high performance 32-bit processor designed for the microcontroller market. This CPU offers significant benefits to users, including:

- outstanding processing performance combined with fast interrupt handling
- enhanced system debug with extensive breakpoint and trace capabilities
- platform security robustness, with integrated memory protection unit (MPU).
- ultra-low power consumption with integrated sleep modes

References to ARM Documentation

The following documents can be found through <http://infocenter.arm.com>

- [1] Cortex-M4 Devices, Generic User Guide (ARM DUI 0553A)
- [2] Cortex Microcontroller Software Interface Standard (CMSIS)

References to ARM Figures

- [3] <http://www.arm.com>

References to IEEE Documentation

- [4] IEEE Standard IEEE Standard for Binary Floating-Point Arithmetic 754-2008.

2.1 Overview

The Cortex-M4 processor is built on a high-performance processor core, with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware including IEEE754-compliant single-precision floating-point computation, a range of single-cycle and SIMD multiplication and multiply-with-accumulate capabilities, saturating arithmetic and dedicated hardware division.

To facilitate the design of cost-sensitive devices, the Cortex-M4 processor implements tightly-coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. The Cortex-M4 processor implements a version of the Thumb® instruction set based on Thumb-2 technology, ensuring high code density and reduced program memory requirements. The Cortex-M4 instruction set provides the exceptional performance expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers.

The Cortex-M4 processor closely integrates a configurable NVIC, to deliver industry-leading interrupt performance. The NVIC includes a non-maskable interrupt (NMI), and provides up to 64 interrupt priority levels. The tight integration of the processor core and

Central Processing Unit (CPU)

NVIC provides fast execution of interrupt service routines (ISRs), dramatically reducing the interrupt latency. This is achieved through the hardware stacking of registers, and the ability to suspend load-multiple and store-multiple operations. Interrupt handlers do not require wrapping in assembler code, removing any code overhead from the ISRs. A tail-chain optimization also significantly reduces the overhead when switching from one ISR to another.

To optimize low-power designs, the NVIC integrates with the sleep modes, that include a deep sleep function that enables the entire device to be rapidly powered down while still retaining program state.

2.1.1 Features

The XMC4300 CPU features comprise

- Thumb2 instruction set combines high code density with 32-bit performance
- IEEE754-compliant single-precision FPU
- power control optimization of system components
- integrated sleep modes for low power consumption
- fast code execution permits slower processor clock or increases sleep mode time
- hardware division and fast digital-signal-processing orientated multiply accumulate
- saturating arithmetic for signal processing
- deterministic, high-performance interrupt handling for time-critical applications
- memory protection unit (MPU) for safety-critical applications
- extensive debug and trace capabilities:
 - Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging, tracing, and code profiling.

2.1.2 Block Diagram

The Cortex-M4 core components comprise:

Processor Core

The CPU provides 16-bit and 32-bit Thumb2 instruction set and DSP/MAC instructions.

Floating-point unit

The FPU provides IEEE754-compliant operations on single-precision, 32-bit, floating-point values.

Nested Vectored Interrupt Controller

The NVIC is an embedded interrupt controller that supports low latency interrupt processing.

Memory Protection Unit

The MPU improves system reliability by defining the memory attributes for different memory regions. It provides up to eight different regions, and an optional predefined background region.

Debug Solution

The XMC4300 implements a complete hardware debug solution.

- Embedded Trace Macrocell
- Traditional JTAG port or a 2-pin Serial Wire Debug Access Port
- Trace port or Serial Wire Viewer
- Flash breakpoints and Data watchpoints

This provides high system control and visibility of the processor and memory even in small package devices.

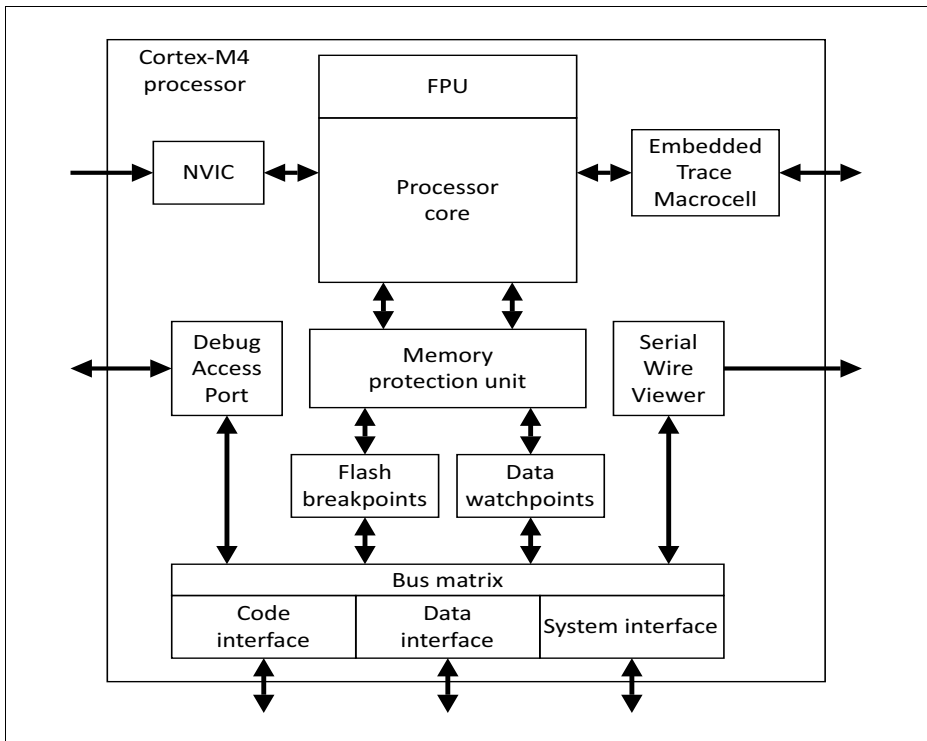


Figure 2-1 Cortex-M4 Block Diagram

System Level Interfaces

The Cortex-M4 processor provides a code, data and system interface using AMBA® technology to provide high speed, low latency accesses.

2.2 Programmers Model

This section describes the Cortex-M4 programmers model. In addition to the individual core register descriptions, it contains information about the processor modes and privilege levels for software execution and stacks.

2.2.1 Processor Mode and Privilege Levels for Software Execution

The processor modes are:

- **Thread mode**
Used to execute application software. The processor enters Thread mode when it comes out of reset.
- **Handler mode**
Used to handle exceptions. The processor returns to Thread mode when it has finished all exception processing.

The privilege levels for software execution are:

- **Unprivileged**
Unprivileged software executes at the unprivileged level.
The software:
 - has limited access to the MSR and MRS instructions, and cannot use the CPS instruction
 - cannot access the system timer, NVIC, or system control block
 - might have restricted access to memory or peripherals.
- **Privileged**
Privileged software executes at the privileged level.
The software can use all the instructions and has access to all resources.

In Thread mode, the CONTROL register controls whether software execution is privileged or unprivileged, see CONTROL register on [Page 2-15](#). In Handler mode, software execution is always privileged.

Only privileged software can write to the CONTROL register to change the privilege level for software execution in Thread mode. Unprivileged software can use the SVC instruction to make a supervisor call to transfer control to privileged software.

2.2.2 Stacks

The processor uses a full descending stack. This means the stack pointer holds the address of the last stacked item in memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory

Central Processing Unit (CPU)

location. The processor implements two stacks, the main stack and the process stack, with a pointer for each held in independent registers, see Stack Pointer on [Page 2-8](#).

In Thread mode, the CONTROL register controls whether the processor uses the main stack or the process stack, see CONTROL register on [Page 2-15](#). In Handler mode, the processor always uses the main stack. The options for processor operations are:

Table 2-1 Summary of processor mode, execution privilege level, and stack use options

Processor mode	Used to execute	Privilege level for software execution	Stack used
Thread	Applications	Privileged or unprivileged ¹⁾	Main stack or process stack ¹⁾
Handler	Exception handlers	Always privileged	Main stack

1) See CONTROL register on [Page 2-15](#).

2.2.3 Core Registers

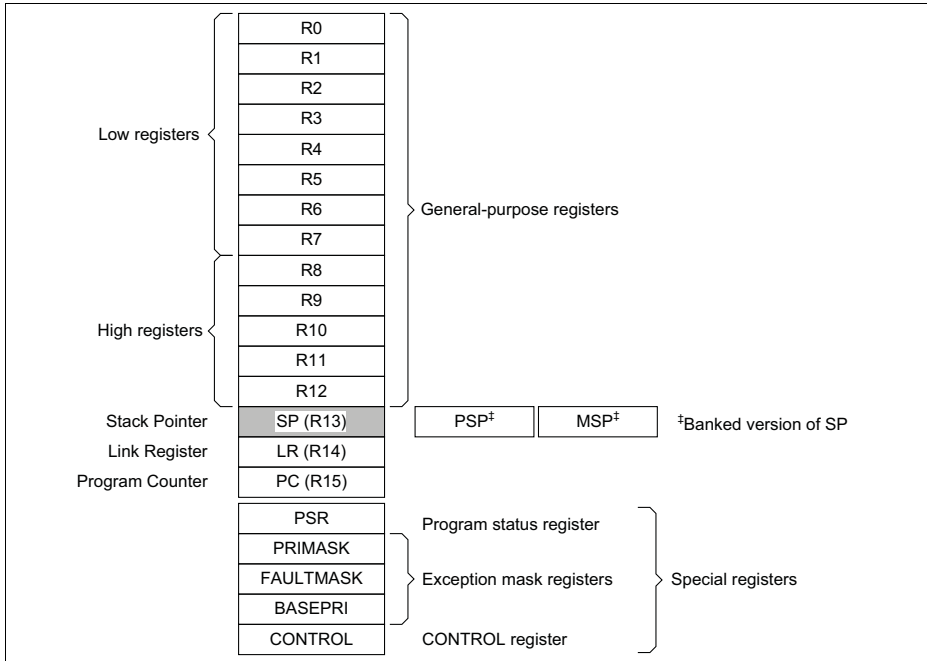


Figure 2-2 Core registers

The processor core registers are:

Table 2-2 Core register set summary

Name	Type ¹⁾	Required privilege ²⁾	Reset value	Description
R0-R12	rw	Either	Unknown	General-purpose registers on Page 2-7
MSP	rw	Privileged	See description	Stack Pointer on Page 2-8
PSP	rw	Either	Unknown	Stack Pointer on Page 2-8
LR	rw	Either	FFFFFFFF _H	Link Register on Page 2-8
PC	rw	Either	See description	Program Counter on Page 2-8

Central Processing Unit (CPU)

Table 2-2 Core register set summary (cont'd)

Name	Type ¹⁾	Required privilege ²⁾	Reset value	Description
PSR	rw	Privileged	01000000 _H	Program Status Register on Page 2-9
ASPR	rw	Either	Unknown	Application Program Status Register on Page 2-9
IPSR	r	Privileged	00000000 _H	Interrupt Program Status Register on Page 2-10
EPSR	r	Privileged	01000000 _H	Execution Program Status Register on Page 2-11
PRIMASK	rw	Privileged	00000000 _H	Priority Mask Register on Page 2-13
FAULTMASK	rw	Privileged	00000000 _H	Fault Mask Register on Page 2-14
BASEPRI	rw	Privileged	00000000 _H	Base Priority Mask Register on Page 2-14
CONTROL	rw	Privileged	00000000 _H	CONTROL register on Page 2-15

1) Describes access type during program execution in thread mode and Handler mode. Debug access can differ.

2) An entry of Either means privileged and unprivileged software can access the register.

General-purpose registers

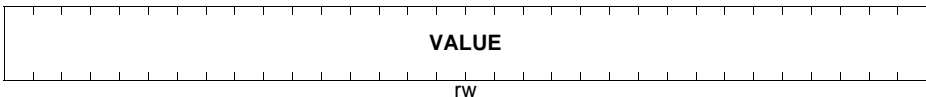
R0-R12 are 32-bit general-purpose registers for data operations

R_x (x=0-12)

General Purpose Register R_x

Reset Value: XXXX XXXX_H

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0



Field	Bits	Type	Description
VALUE	[31:0]	rw	Content of Register

Stack Pointer

The Stack Pointer (SP) is register R13. In Thread mode, bit[1] of the CONTROL register indicates the stack pointer to use:

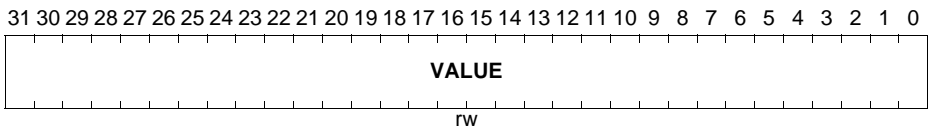
- 0 = Main Stack Pointer (MSP). This is the reset value.
- 1 = Process Stack Pointer (PSP).

On reset, the processor loads the MSP with the value from address 00000000_H.

SP

Stack Pointer

Reset Value: 2000 FF3C_H



Field	Bits	Type	Description
VALUE	[31:0]	rw	Content of Register

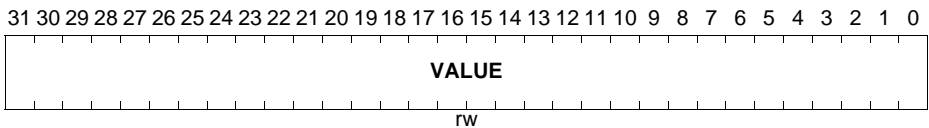
Link Register

The Link Register (LR) is register R14. It stores the return information for subroutines, function calls, and exceptions. On reset, the processor sets the LR value to FFFFFFFF_H.

LR

Link Register

Reset Value: FFFF FFFF_H



Field	Bits	Type	Description
VALUE	[31:0]	rw	Content of Register

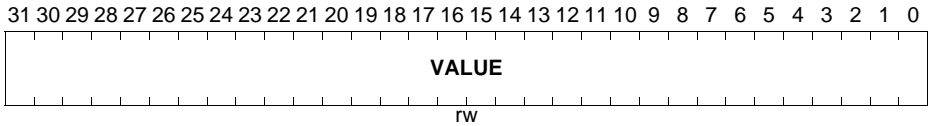
Program Counter

The Program Counter (PC) is register R15. It contains the current program address. On reset, the processor loads the PC with the value of the reset vector, which is at address 00000004_H. Bit[0] of the value is loaded into the EPSR T-bit at reset and must be 1.

PC

Program Counter

Reset Value: 0000 0004_H



Field	Bits	Type	Description
VALUE	[31:0]	rw	Content of Register

Program Status Register

The Program Status Register (PSR) combines:

- Application Program Status Register (APSR)
- Interrupt Program Status Register (IPSR)
- Execution Program Status Register (EPSR)

These registers are mutually exclusive bit fields in the 32-bit PSR.

Access these registers individually or as a combination of any two or all three registers, using the register name as an argument to the MSR or MRS instructions. For example:

- read all of the registers using PSR with the MRS instruction
- write to the APSR N, Z, C, V, and Q bits using APSR_nzcvq with the MSR instruction.

The PSR combinations and attributes are:

Table 2-3 PSR register combinations

Register	Type	Combination
PSR	rw ¹⁾²⁾	APSR, EPSR, and IPSR
IEPSR	r	EPSR and IPSR
IAPSR	rw ¹⁾	APSR and IPSR
EAPSR	rw ²⁾	APSR and EPSR

1) The processor ignores writes to the IPSR bits.

2) Reads of the EPSR bits return zero, and the processor ignores writes to the these bits

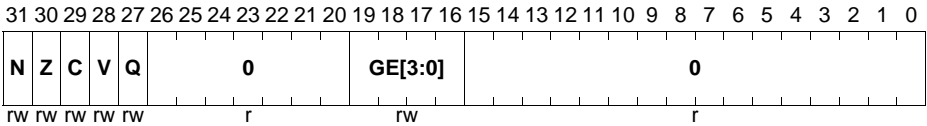
Application Program Status Register

The APSR contains the current state of the condition flags from previous instruction executions. See the register summary in [Table 2-2](#) on [Page 2-6](#) for its attributes.

APSR

Application Program Status Register

Reset Value: XXXX XXXX_H



Field	Bits	Type	Description
GE[3:0]	[19:16]	rw	Greater than or Equal flags Please refer also to SEL instruction.
Q	27	rw	DSP overflow and saturation flag
V	28	rw	Overflow flag
C	29	rw	Carry or borrow flag
Z	30	rw	Zero flag
N	31	rw	Negative flag
0	[26:20], [15:0]	r	Reserved Read as 0; should be written with 0.

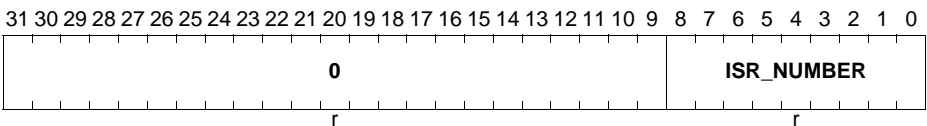
Interrupt Program Status Register

The IPSR contains the exception type number of the current Interrupt Service Routine (ISR). See the register summary in [Table 2-2](#) on [Page 2-6](#) for its attributes.

IPSR

Interrupt Program Status Register

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ISR_NUMBER	[8:0]	r	<p>Number of the current exception</p> <p>0_D Thread mode</p> <p>1_D Reserved</p> <p>2_D NMI</p> <p>3_D HardFault</p> <p>4_D MemManage</p> <p>5_D BusFault</p> <p>6_D UsageFault</p> <p>7_D Reserved</p> <p>8_D Reserved</p> <p>9_D Reserved</p> <p>10_D Reserved</p> <p>11_D SVCall</p> <p>12_D Reserved for Debug</p> <p>13_D Reserved</p> <p>14_D PendSV</p> <p>15_D SysTick</p> <p>16_D IRQ0</p> <p>...</p> <p>127_D IRQ111</p> <p>Values > 127_D undefined.</p> <p>See Exception types on Page 2-26 for more information.</p>
0	[31:9]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Execution Program Status Register

The EPSR contains the Thumb state bit, and the execution state bits for either the:

- If-Then (IT) instruction
- Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction.

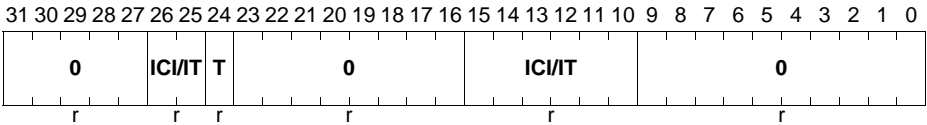
See the register summary in [Table 2-2](#) on [Page 2-6](#) for the EPSR attributes.

Attempts to read the EPSR directly through application software using the MSR instruction always return zero. Attempts to write the EPSR using the MSR instruction in application software are ignored.

EPSR

Execution Program Status Register

Reset Value: 0100 0000_H



Field	Bits	Type	Description
IC/IT	[26:25], [15:10]	r	Interruptible-continuable instruction bits/Execution state bits of the IT instruction Please refer also to IT instruction.
T	24	r	Thumb state bit Thumb state.
0	[31:27], [23:16], [9:0]	r	Reserved Read as 0; should be written with 0.

Interruptible-continuable instructions

When an interrupt occurs during the execution of an LDM, STM, PUSH, POP, VLDM, VSTM, V PUSH, or VPOP instruction, the processor:

- stops the load multiple or store multiple instruction operation temporarily
- stores the next register operand in the multiple operation to EPSR bits[15:12]

After servicing the interrupt, the processor:

- returns to the register pointed to by bits[15:12]
- resumes execution of the multiple load or store instruction.

When the EPSR holds ICI execution state, bits[26:25,11:10] are zero.

If-Then block

The If-Then block contains up to four instructions following an IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See IT on page 3-122 for more information.

Thumb state

The Cortex-M4 processor only supports execution of instructions in Thumb state. The following can clear the T bit to 0:

- instructions BLX, BX and POP{PC}

Central Processing Unit (CPU)

- restoration from the stacked xPSR value on an exception return
- bit[0] of the vector value on an exception entry or reset.

Attempting to execute instructions when the T bit is 0 results in a fault or lockup. See Lockup on [Page 2-39](#) for more information.

Exception mask registers

The exception mask registers disable the handling of exceptions by the processor. Disable exceptions where they might impact on timing critical tasks.

To access the exception mask registers use the MSR and MRS instructions, or the CPS instruction to change the value of PRIMASK or FAULTMASK.

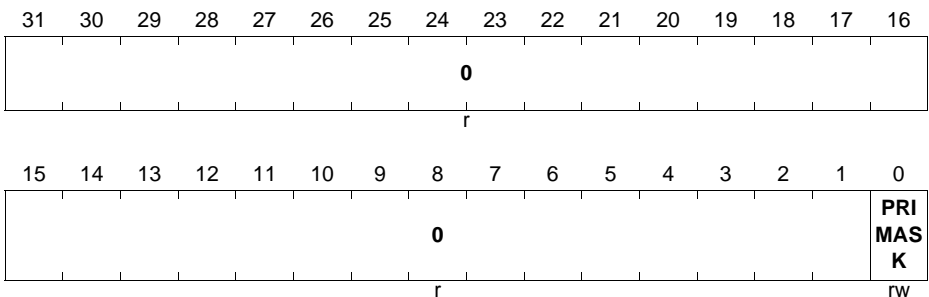
Priority Mask Register

The PRIMASK register prevents activation of all exceptions with configurable priority. See the register summary in [Table 2-2](#) on [Page 2-6](#) for its attributes.

PRIMASK

Priority Mask Register

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PRIMASK	0	rw	Priority Mask 0 _B No effect 1 _B Prevents the activation of all exceptions with configurable priority.
0	[31:1]	r	Reserved Read as 0; should be written with 0.

Central Processing Unit (CPU)

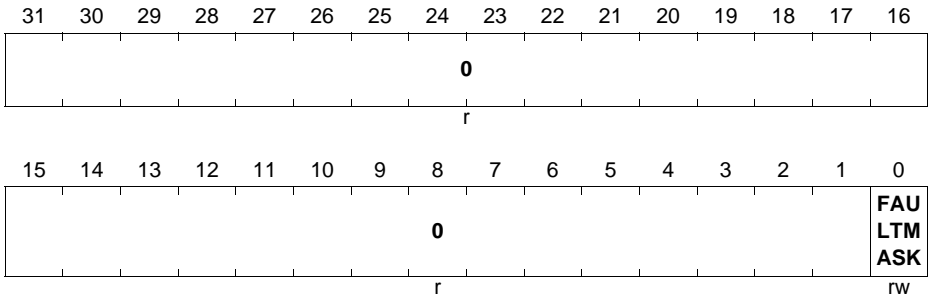
Fault Mask Register

The FAULTMASK register prevents activation of all exceptions except for Non-Maskable Interrupt (NMI). See the register summary in [Table 2-2](#) on [Page 2-6](#) for its attributes.

FAULTMASK

Fault Mask Register

Reset Value: 0000 0000_H



Field	Bits	Type	Description
FAULTMASK	0	rw	Fault Mask 0 _B no effect 1 _B prevents the activation of all exceptions except for NMI.
0	[31:1]	r	Reserved Read as 0; should be written with 0.

The processor clears the FAULTMASK bit to 0 on exit from any exception handler except the NMI handler.

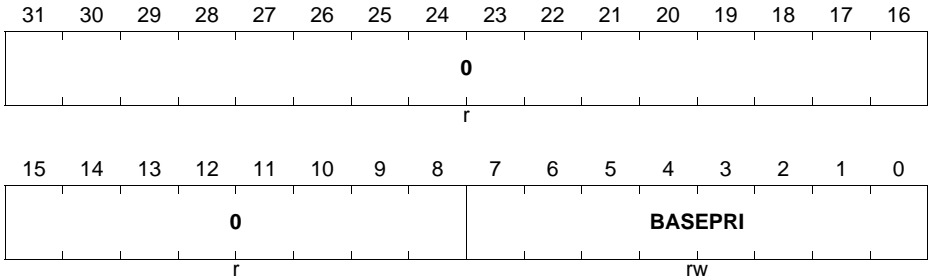
Base Priority Mask Register

The BASEPRI register defines the minimum priority for exception processing. When BASEPRI is set to a nonzero value, it prevents the activation of all exceptions with the same or lower priority level as the BASEPRI value. See the register summary in [Table 2-2](#) on [Page 2-6](#) for its attributes.

BASEPRI

Base Priority Mask Register

Reset Value: 0000 0000_H



Field	Bits	Type	Description
BASEPRI ¹⁾	[7:0]	rw	Priority mask bits 0 _H no effect others , defines the base priority for exception processing. The processor does not process any exception with a priority value greater than or equal to BASEPRI.
0	[31:8]	r	Reserved Read as 0; should be written with 0.

1) This field is similar to the priority fields in the interrupt priority registers. The XMC4300 implements only bits[7:2] of this field, bits[1:0] read as zero and ignore writes. See [Interrupt Priority Registers](#) on [Page 2-90](#) for more information. Remember that higher priority field values correspond to lower exception priorities.

CONTROL register

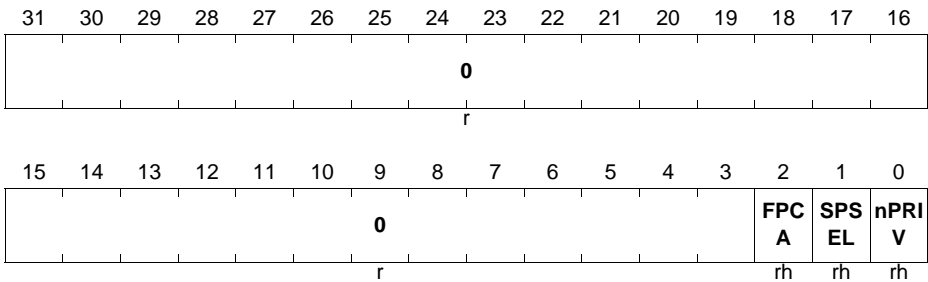
The CONTROL register controls the stack used and the privilege level for software execution when the processor is in Thread mode and indicates whether the FPU state is active. See the register summary in [Table 2-2](#) on [Page 2-6](#) for its attributes.

Central Processing Unit (CPU)

CONTROL

CONTROL register

Reset Value: 0000 0000_H



Field	Bits	Type	Description
nPRIV	0	rh	Thread mode privilege level 0 _B Privileged 1 _B Unprivileged
SPSEL	1	rh	Currently active stack pointer In Handler mode this bit reads as zero and ignores writes. The Cortex-M4 updates this bit automatically on exception return. 0 _B MSP is the current stack pointer 1 _B PSP is the current stack pointer
FPCA	2	rh	Floating-point context currently active 0 _B No floating-point context active 1 _B Floating-point context active The Cortex-M4 uses this bit to determine whether to preserve floating-point state when processing an exception.
0	[31:3]	r	Reserved Read as 0; should be written with 0.

Handler mode always uses the MSP, so the processor ignores explicit writes to the active stack pointer bit of the CONTROL register when in Handler mode. The exception entry and return mechanisms automatically update the CONTROL register based on the EXC_RETURN value, see [Table 2-9](#) on [Page 2-36](#).

In an OS environment, ARM recommends that threads running in Thread mode use the process stack and the kernel and exception handlers use the main stack.

Central Processing Unit (CPU)

By default, Thread mode uses the MSP. To switch the stack pointer used in Thread mode to the PSP, either:

- use the MSR instruction to set the Active stack pointer bit to 1.
- perform an exception return to Thread mode with the appropriate EXC_RETURN value, see [Table 2-9](#) on [Page 2-36](#).

Note: When changing the stack pointer, software must use an ISB instruction immediately after the MSR instruction. This ensures that instructions after the ISB instruction execute using the new stack pointer.

2.2.4 Exceptions and Interrupts

The Cortex-M4 processor supports interrupts and system exceptions. The processor and the NVIC prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses Handler mode to handle all exceptions except for reset. See Exception entry on [Page 2-33](#) and Exception return on [Page 2-36](#) for more information.

The NVIC registers control interrupt handling. See [Page 2-43](#) for more information.

2.2.5 Data Types

The processor:

- supports the following data types:
 - 32-bit words
 - 16-bit halfwords
 - 8-bit bytes
- manages all data memory accesses as little-endian. See Memory regions, types and attributes on [Page 2-20](#) for more information.

2.2.6 The Cortex Microcontroller Software Interface Standard

For a Cortex-M4 microcontroller system, the Cortex Microcontroller Software Interface Standard (CMSIS) [\[2\]](#) defines:

- a common way to:
 - access peripheral registers
 - define exception vectors
- the names of:
 - the registers of the core peripherals
 - the core exception vectors
- a device-independent interface for RTOS kernels, including a debug channel.

The CMSIS includes address definitions and data structures for the core peripherals in the Cortex-M4 processor.

Central Processing Unit (CPU)

CMSIS simplifies software development by enabling the reuse of template code and the combination of CMSIS-compliant software components from various middleware vendors. Software vendors can expand the CMSIS to include their peripheral definitions and access functions for those peripherals.

This document includes the register names defined by the CMSIS, and gives short descriptions of the CMSIS functions that address the processor core and the core peripherals.

Note: This document uses the register short names defined by the CMSIS. In a few cases these differ from the architectural short names that might be used in other documents.

The following sections give more information about the CMSIS:

- Power management programming hints on [Page 2-42](#)
- CMSIS functions on [Page 2-18](#)
- Using CMSIS functions to access NVIC on [Page 2-45](#)

For additional information please refer to <http://www.onarm.com/cmsis>

2.2.7 CMSIS functions

ISO/IEC C code cannot directly access some Cortex-M4 instructions. This section describes intrinsic functions that can generate these instructions, provided by the CMSIS and that might be provided by a C compiler. If a C compiler does not support an appropriate intrinsic function, you might have to use inline assembler to access some instructions.

The CMSIS provides the following intrinsic functions to generate instructions that ISO/IEC C code cannot directly access:

Table 2-4 CMSIS functions to generate some Cortex-M4 instructions

Instruction	CMSIS function
CPSIE I	<code>void __enable_irq(void)</code>
CPSID I	<code>void __disable_irq(void)</code>
CPSIE F	<code>void __enable_fault_irq(void)</code>
CPSID F	<code>void __disable_fault_irq(void)</code>
ISB	<code>void __ISB(void)</code>
DSB	<code>void __DSB(void)</code>
DMB	<code>void __DMB(void)</code>
REV	<code>uint32_t __REV(uint32_t int value)</code>
REV16	<code>uint32_t __REV16(uint32_t int value)</code>

Central Processing Unit (CPU)

Table 2-4 CMSIS functions to generate some Cortex-M4 instructions (cont'd)

Instruction	CMSIS function
REVSH	uint32_t __REVSH(uint32_t int value)
RBIT	uint32_t __RBIT(uint32_t int value)
SEV	void __SEV(void)
WFE	void __WFE(void)
WFI	void __WFI(void)

The CMSIS also provides a number of functions for accessing the special registers using MRS and MSR instructions:

Table 2-5 CMSIS functions to access the special registers

Special register	Access	CMSIS function
PRIMASK	Read	uint32_t __get_PRIMASK (void)
	Write	void __set_PRIMASK (uint32_t value)
FAULTMASK	Read	uint32_t __get_FAULTMASK (void)
	Write	void __set_FAULTMASK (uint32_t value)
BASEPRI	Read	uint32_t __get_BASEPRI (void)
	Write	void __set_BASEPRI (uint32_t value)
CONTROL	Read	uint32_t __get_CONTROL (void)
	Write	void __set_CONTROL (uint32_t value)
MSP	Read	uint32_t __get_MSP (void)
	Write	void __set_MSP (uint32_t TopOfMainStack)
PSP	Read	uint32_t __get_PSP (void)
	Write	void __set_PSP (uint32_t TopOfProcStack)

2.3 Memory Model

This section describes the processor memory map and the behavior of memory accesses. The processor has a fixed default memory map that provides up to 4GB of addressable memory. The memory map is:

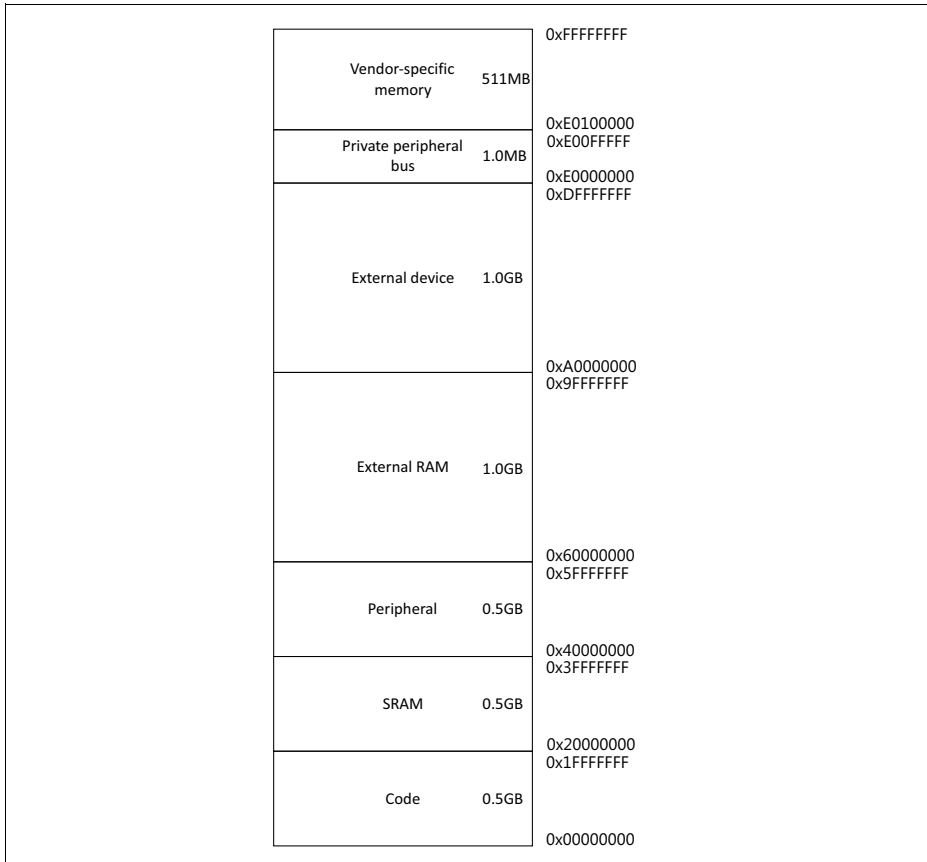


Figure 2-3 Memory map

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers, see About the Private Peripherals on [Page 2-42](#).

2.3.1 Memory Regions, Types and Attributes

The memory map and the programming of the MPU splits the memory map into regions. Each region has a defined memory type, and some regions have additional memory

Central Processing Unit (CPU)

attributes. The memory type and attributes determine the behavior of accesses to the region.

The memory types are:

Normal	The processor can re-order transactions for efficiency, or perform speculative reads.
Device	The processor preserves transaction order relative to other transactions to Device or Strongly-ordered memory.
Strongly-ordered	The processor preserves transaction order relative to all other transactions.

The different ordering requirements for Device and Strongly-ordered memory mean that the memory system can buffer a write to Device memory, but must not buffer a write to Strongly-ordered memory.

The additional memory attributes include:

Execute Never (XN) Means the processor prevents instruction accesses. A fault exception is generated only on execution of an instruction executed from an XN region.

2.3.2 Memory System Ordering of Memory Accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not guarantee that the order in which the accesses complete matches the program order of the instructions, providing this does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions. See Software ordering of memory accesses on [Page 2-23](#).

However, the memory system does guarantee some ordering of accesses to Device and Strongly-ordered memory. For two memory access instructions A1 and A2, if A1 occurs before A2 in program order, the ordering of the memory accesses caused by two instructions is:

A1 \ A2	Normal access	Device access	Strongly-ordered access
Normal access	-	-	-
Device access	-	<	<
Strongly-ordered access	-	<	<

Figure 2-4 Ordering of Memory Accesses

Where:

- “-” Means that the memory system does not guarantee the ordering of the accesses.
- “<” Means that accesses are observed in program order, that is, A1 is always observed before A2.

2.3.3 Behavior of Memory Accesses

The behavior of accesses to each region in the memory map is:

Table 2-6 Memory access behavior

Address range	Memory region	Memory type ¹⁾	XN ¹⁾	Description
0x00000000-0x1FFFFFFF	Code	Normal	-	Executable region for program code. You can also put data here.
0x20000000-0x3FFFFFFF	SRAM	Normal	-	Executable region for data. You can also put code here.
0x40000000-0x5FFFFFFF	Peripheral	Device	XN	Peripherals region.
0x60000000-0x9FFFFFFF	External RAM	Normal	-	Executable region for data.
0xA0000000-0xDFFFFFFF	External device	Device	XN	External Device memory.
0xE0000000-0xE00FFFFFFF	Private Peripheral Bus	Strongly-ordered	XN	This region includes the NVIC, System timer, and system control block.
0xE0100000-0xFFFFFFFF	Vendor-specific device	Device	XN	Accesses to this region are to vendor-specific peripherals.

1) See Memory regions, types and attributes on [Page 2-20](#) for more information.

The Code, SRAM, and external RAM regions can hold programs. However, it is recommended that programs always use the Code region. This is because the processor has separate buses that enable instruction fetches and data accesses to occur simultaneously.

The MPU can override the default memory access behavior described in this section. For more information, see Memory protection unit on [Page 2-46](#).

Instruction prefetch and branch prediction

The Cortex-M4 processor:

- prefetches instructions ahead of execution
- speculatively prefetches from branch target addresses.

2.3.4 Software Ordering of Memory Accesses

The order of instructions in the program flow does not always guarantee the order of the corresponding memory transactions. This is because:

- the processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence.
- The processor has multiple bus interfaces
- memory or devices in the memory map have different wait states
- some memory accesses are buffered or speculative.

Memory system ordering of memory accesses on [Page 2-21](#) describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The processor provides the following memory barrier instructions:

- | | |
|------------|--|
| DMB | The Data Memory Barrier (DMB) instruction ensures that outstanding memory transactions complete before subsequent memory transactions. |
| DSB | The Data Synchronization Barrier (DSB) instruction ensures that outstanding memory transactions complete before subsequent instructions execute. |
| ISB | The Instruction Synchronization Barrier (ISB) ensures that the effect of all completed memory transactions is recognizable by subsequent instructions. |

MPU programming

Use a DSB followed by an ISB instruction or exception return to ensure that the new MPU configuration is used by subsequent instructions.

2.3.5 Memory Endianness

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. The XMC4300 stores information “Little-endian” format.

Little-endian format

In little-endian format, the processor stores the least significant byte of a word at the lowest-numbered byte, and the most significant byte at the highest-numbered byte. For example:

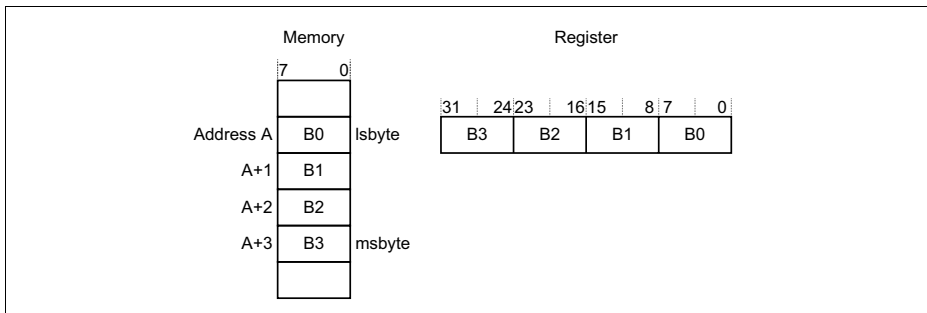


Figure 2-5 Little-endian format

2.3.6 Synchronization Primitives

The Cortex-M4 instruction set includes pairs of synchronization primitives. These provide a non-blocking mechanism that a thread or process can use to obtain exclusive access to a memory location. Software can use them to perform a guaranteed read-modify-write memory update sequence, or for a semaphore mechanism.

A pair of synchronization primitives comprises:

A Load-Exclusive instruction

Used to read the value of a memory location, requesting exclusive access to that location.

A Store-Exclusive instruction

Used to attempt to write to the same memory location, returning a status bit to a register. If this bit is:

- 0 it indicates that the thread or process gained exclusive access to the memory, and the write succeeds,
- 1 it indicates that the thread or process did not gain exclusive access to the memory, and no write was performed.

The pairs of Load-Exclusive and Store-Exclusive instructions are:

- the word instructions LDREX and STREX
- the halfword instructions LDREXH and STREXH
- the byte instructions LDREXB and STREXB.

Software must use a Load-Exclusive instruction with the corresponding Store-Exclusive instruction.

To perform an exclusive read-modify-write of a memory location, software must:

1. Use a Load-Exclusive instruction to read the value of the location.
2. Modify the value, as required.
3. Use a Store-Exclusive instruction to attempt to write the new value back to the memory location.
4. Test the returned status bit. If this bit is:
 - 0 The read-modify-write completed successfully.
 - 1 No write was performed. This indicates that the value returned at step 1 might be out of date. The software must retry the entire read-modify-write sequence.

Software can use the synchronization primitives to implement a semaphores as follows:

1. Use a Load-Exclusive instruction to read from the semaphore address to check whether the semaphore is free.
2. If the semaphore is free, use a Store-Exclusive to write the claim value to the semaphore address.
3. If the returned status bit from step 2 indicates that the Store-Exclusive succeeded then the software has claimed the semaphore. However, if the Store-Exclusive failed, another process might have claimed the semaphore after the software performed step 1.

The Cortex-M4 includes an exclusive access monitor, that tags the fact that the processor has executed a Load-Exclusive instruction.

The processor removes its exclusive access tag if:

- It executes a CLREX instruction.
- It executes a Store-Exclusive instruction, regardless of whether the write succeeds.
- An exception occurs. This means the processor can resolve semaphore conflicts between different threads.

2.3.7 Programming Hints for the Synchronization Primitives

ISO/IEC C cannot directly generate the exclusive access instructions. CMSIS provides intrinsic functions for generation of these instructions:

Table 2-7 CMSIS functions for exclusive access instructions

Instruction	CMSIS function
LDREX	<code>uint32_t __LDREXW (uint32_t *addr)</code>
LDREXH	<code>uint16_t __LDREXH (uint16_t *addr)</code>
LDREXB	<code>uint8_t __LDREXB (uint8_t *addr)</code>
STREX	<code>uint32_t __STREXW (uint32_t value, uint32_t *addr)</code>
STREXH	<code>uint32_t __STREXH (uint16_t value, uint16_t *addr)</code>
STREXB	<code>uint32_t __STREXB (uint8_t value, uint8_t *addr)</code>
CLREX	<code>void __CLREX (void)</code>

For example:

```
uint16_t value;
uint16_t *address = 0x20001002;
value = __LDREXH (address); // load 16-bit value from memory
address 0x20001002
```

2.4 Instruction Set

The Cortex-M4 instruction set reference is available through [\[1\]](#)

2.5 Exception Model

This section describes the exception model. It describes:

- Exception states
- Exception types
- Exception handlers on [Page 2-27](#)
- Vector table on [Page 2-30](#)
- Exception priorities on [Page 2-31](#)
- Interrupt priority grouping on [Page 2-31](#)
- Exception entry and return on [Page 2-32](#)

2.5.1 Exception States

Each exception is in one of the following states:

Inactive	The exception is not active and not pending.
Pending	The exception is waiting to be serviced by the processor. An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.
Active	An exception that is being serviced by the processor but has not completed. <i>Note: An exception handler can interrupt the execution of another exception handler. In this case both exceptions are in the active state.</i>
Active and pending	The exception is being serviced by the processor and there is a pending exception from the same source.

2.5.2 Exception Types

The exception types are:

Reset	Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts as privileged execution in Thread mode.
NMI	A NonMaskable Interrupt (NMI) can be signalled by a peripheral or triggered by software. This is the highest priority exception other than reset. It is permanently enabled and has a fixed priority of -2. NMIs cannot be: <ul style="list-style-type: none"> • masked or prevented from activation by any other exception • preempted by any exception other than Reset.
HardFault	A HardFault is an exception that occurs because of an error during exception processing, or because an exception cannot be managed by any other exception mechanism. HardFaults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority.
MemManage	A MemManage fault is an exception that occurs because of a memory protection related fault. The MPU or the fixed memory protection constraints determines this fault, for both instruction and data memory transactions. This fault is always used to abort instruction accesses to Execute Never (XN) memory regions.

Central Processing Unit (CPU)

- BusFault** A BusFault is an exception that occurs because of a memory related fault for an instruction or data memory transaction. This might be from an error detected on a bus in the memory system.
- UsageFault** A UsageFault is an exception that occurs because of a fault related to instruction execution. This includes:
- an undefined instruction
 - an illegal unaligned access
 - invalid state on instruction execution
 - an error on exception return.
- The following can cause a UsageFault when the core is configured to report them:
- an unaligned address on word and halfword memory access
 - division by zero.
- SVC** A supervisor call (SVC) is an exception that is triggered by the SVC instruction. In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers.
- PendSV** PendSV is an interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active.
- SysTick** A SysTick exception is an exception the system timer generates when it reaches zero. Software can also generate a SysTick exception. In an OS environment, the processor can use this exception as system tick.
- Interrupt (IRQ)** A interrupt, or IRQ, is an exception signalled by a peripheral, or generated by a software request. All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor.

Table 2-8 Properties of the different exception types

Exception number ¹⁾	IRQ number ¹⁾	Exception type	Priority	Vector address or offset ²⁾	Activation
1	-	Reset	-3, the highest	0x00000004	Asynchronous
2	-14	NMI	-2	0x00000008	Asynchronous
3	-13	HardFault	-1	0x0000000C	-
4	-12	MemManage	Configurable ³⁾	0x00000010	Synchronous

Table 2-8 Properties of the different exception types (cont'd)

Exception number ¹⁾	IRQ number ¹⁾	Exception type	Priority	Vector address or offset ²⁾	Activation
5	-11	BusFault	Configurable ³⁾	0x00000014	Synchronous when precise, asynchronous when imprecise
6	-10	UsageFault	Configurable ³⁾	0x00000018	Synchronous
7-10	-	Reserved	-	-	-
11	-5	SVCall	Configurable ³⁾	0x0000002C	Synchronous
12-13	-	Reserved	-	-	-
14	-2	PendSV	Configurable ³⁾	0x00000038	Asynchronous
15	-1	SysTick	Configurable ³⁾	0x0000003C	Asynchronous
16 and above	0 and above	Interrupt (IRQ)	Configurable ⁴⁾	0x00000040 and above ⁵⁾	Asynchronous

1) To simplify the software layer, the CMSIS only uses IRQ numbers and therefore uses negative values for exceptions other than interrupts. The IPSR returns the Exception number, see Interrupt Program Status Register on [Page 2-10](#).

2) See Vector table for more information.

3) See System Handler Priority Registers on [Page 2-70](#)

4) See Interrupt Priority Registers on [Page 2-90](#).

5) Increasing in steps of 4.

For an asynchronous exception, other than reset, the processor can execute another instruction between when the exception is triggered and when the processor enters the exception handler.

Privileged software can disable the exceptions that [Table 2-8](#) on [Page 2-28](#) shows as having configurable priority, see:

- System Handler Control and State Register on [Page 2-72](#)
- Interrupt Clear-enable Registers on [Page 2-88](#).

For more information about HardFaults, MemManage faults, BusFaults, and UsageFaults, see Fault handling on [Page 2-36](#).

2.5.3 Exception Handlers

The processor handles exceptions using:

- Interrupt Service Routines (ISRs)** Interrupts IRQ0 to IRQ111 are the exceptions handled by ISRs.
- Fault handlers** HardFault, MemManage fault, UsageFault, and BusFault are fault exceptions handled by the fault handlers.
- System handlers** NMI, PendSV, SVCall SysTick, and the fault exceptions are all system exceptions that are handled by system handlers.

2.5.4 Vector Table

The vector table contains the reset value of the stack pointer, and the start addresses, also called exception vectors, for all exception handlers. [Figure 2-6](#) on [Page 2-30](#) shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is Thumb code, see Thumb state on [Page 2-12](#).

Exception number	IRQ number	Offset	Vector
127	111	0x01FC	IRQ111
.	.	.	.
.	.	.	.
.	.	.	.
18	2	0x004C	IRQ2
17	1	0x0048	IRQ1
16	0	0x0044	IRQ0
15	-1	0x0040	Systick
14	-2	0x003C	PendSV
13		0x0038	Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
10			Reserved
9			
8			
7			
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value

Figure 2-6 Vector table

On system reset, the vector table is fixed at address 0x00000000. Privileged software can write to the VTOR to relocate the vector table start address to a different memory location, in the range 0x00000400 to 0x3FFFC00, see Vector Table Offset Register on [Page 2-63](#).

2.5.5 Exception Priorities

As [Table 2-8](#) on [Page 2-28](#) shows, all exceptions have an associated priority, with:

- a lower priority value indicating a higher priority
- configurable priorities for all exceptions except Reset, HardFault, and NMI.

If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities see

- System Handler Priority Registers on [Page 2-70](#)
- Interrupt Priority Registers on [Page 2-90](#).

Note: Configurable priority values are in the range 0-63. This means that the Reset, HardFault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.

For example, assigning a higher priority value to IRQ[0] and a lower priority value to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

2.5.6 Interrupt Priority Grouping

To increase priority control in systems with interrupts, the NVIC supports priority grouping. This divides each interrupt priority register entry into two fields:

- an upper field that defines the group priority
- a lower field that defines a subpriority within the group.

Only the group priority determines preemption of interrupt exceptions. When the processor is executing an interrupt exception handler, another interrupt with the same group priority as the interrupt being handled does not preempt the handler,

If multiple pending interrupts have the same group priority, the subpriority field determines the order in which they are processed. If multiple pending interrupts have the same group priority and subpriority, the interrupt with the lowest IRQ number is processed first.

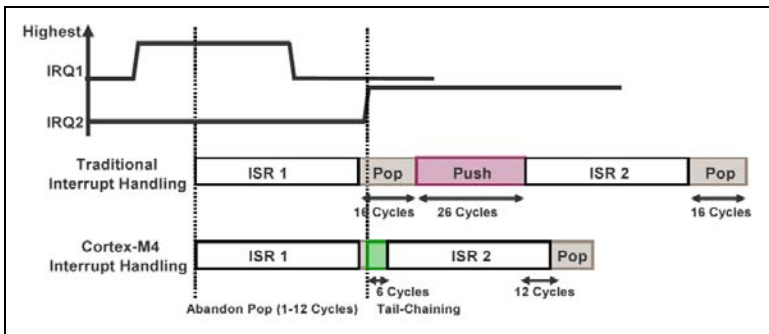
Central Processing Unit (CPU)

For information about splitting the interrupt priority fields into group priority and subpriority, see Application Interrupt and Reset Control Register on [Page 2-64](#).

2.5.7 Exception Entry and Return

Descriptions of exception handling use the following terms:

Preemption When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See Interrupt priority grouping for more information about preemption by an interrupt.
When one exception preempts another, the exceptions are called nested exceptions. See Exception entry on [Page 2-33](#) more information.



Source of figure [\[3\]](#).

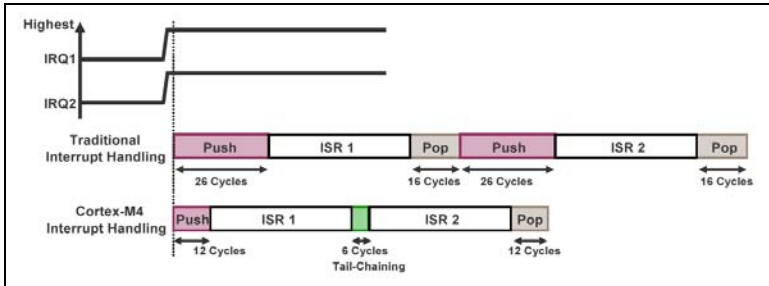
Return

This occurs when the exception handler is completed, and:

- there is no pending exception with sufficient priority to be serviced
- the completed exception handler was not handling a late-arriving exception.

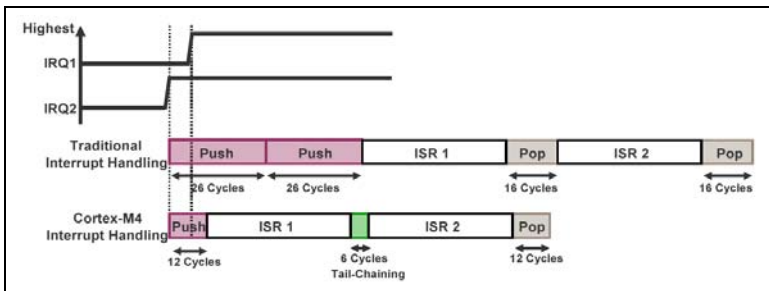
The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See Exception return on [Page 2-36](#) for more information.

Tail-chaining This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.



Source of figure [3].

Late-arriving This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved is the same for both exceptions. Therefore the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. On return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.



Source of figure [3].

Exception entry

Exception entry occurs when there is a pending exception with sufficient priority and either:

Central Processing Unit (CPU)

- the processor is in Thread mode
- the new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception.

When one exception preempts another, the exceptions are nested.

Sufficient priority means the exception has more priority than any limits set by the mask registers, see Exception mask registers on [Page 2-13](#). An exception with less priority than this is pending but is not handled by the processor.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred to as stacking and the structure of eight data words is referred to as the stack frame.

When using floating-point routines, the Cortex-M4 processor automatically stacks the architected floating-point state on exception entry. [Figure 2-7](#) on [Page 2-35](#) shows the Cortex-M4 stack frame layout when floating-point state is preserved on the stack as the result of an interrupt or an exception.

Note: Where stack space for floating-point state is not allocated, the stack frame is the same as that of ARMv7-M implementations without an FPU. [Figure 2-7](#) on [Page 2-35](#) shows this stack frame also.

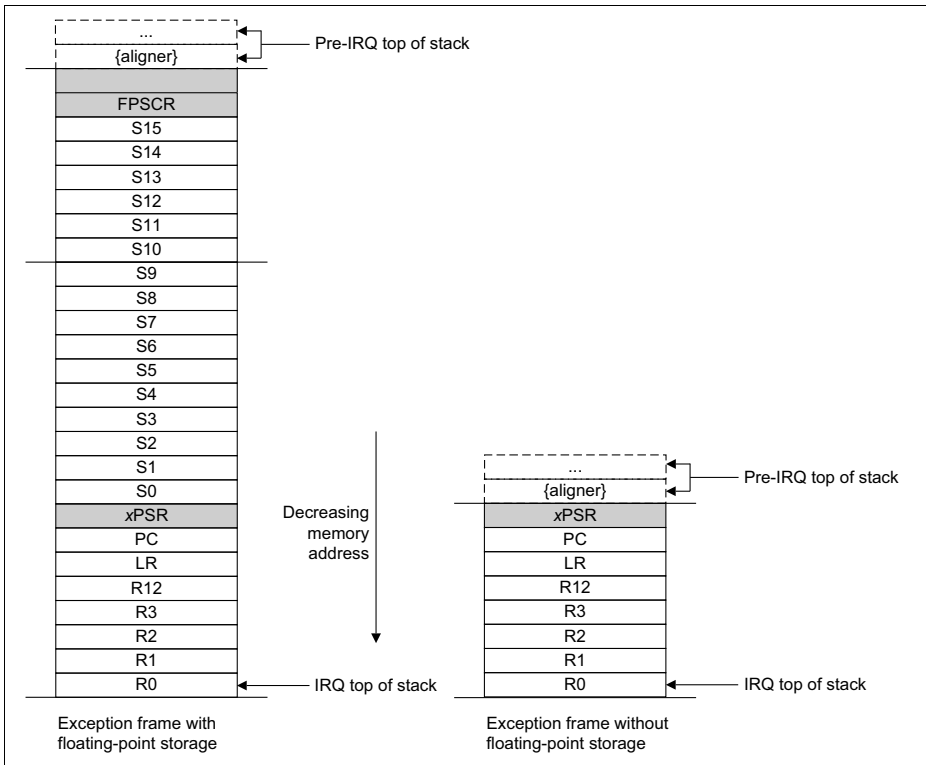


Figure 2-7 Exception stack frame

Immediately after stacking, the stack pointer indicates the lowest address in the stack frame. The alignment of the stack frame is controlled via the STKALIGN bit of the Configuration Control Register (CCR).

The stack frame includes the return address. This is the address of the next instruction in the interrupted program. This value is restored to the PC at exception return so that the interrupted program resumes.

In parallel to the stacking operation, the processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC_RETURN value to the LR. This indicates which stack pointer corresponds to the stack frame and what operation mode the processor was in before the entry occurred.

If no higher priority exception occurs during exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active.

Central Processing Unit (CPU)

If another higher priority exception occurs during exception entry, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception. This is the late arrival case.

Exception return

Exception return occurs when the processor is in Handler mode and executes one of the following instructions to load the EXC_RETURN value into the PC:

- an LDM or POP instruction that loads the PC
- an LDR instruction with PC as the destination
- a BX instruction using any register.

EXC_RETURN is the value loaded into the LR on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. The lowest five bits of this value provide information on the return stack and processor mode. **Table 2-9** shows the EXC_RETURN values with a description of the exception return behavior.

All EXC_RETURN values have bits[31:5] set to one. When this value is loaded into the PC it indicates to the processor that the exception is complete, and the processor initiates the appropriate exception return sequence.

Table 2-9 Exception return behavior

EXC_RETURN[31:0]	Description
0xFFFFFFFF1	Return to Handler mode, exception return uses non-floating-point state from the MSP and execution uses MSP after return.
0xFFFFFFFF9	Return to Thread mode, exception return uses non-floating-point state from MSP and execution uses MSP after return.
0xFFFFFFFFD	Return to Thread mode, exception return uses non-floating-point state from the PSP and execution uses PSP after return.
0xFFFFFFFFE1	Return to Handler mode, exception return uses floating-point state from MSP and execution uses MSP after return.
0xFFFFFFFFE9	Return to Thread mode, exception return uses floating-point state from MSP and execution uses MSP after return.
0xFFFFFFFFED	Return to Thread mode, exception return uses floating-point state from PSP and execution uses PSP after return.

2.6 Fault Handling

Faults are a subset of the exceptions, see Exception model on **Page 2-26**. Faults are generated by:

- a bus error on:

Central Processing Unit (CPU)

- an instruction fetch or vector table load
- a data access.
- an internally-detected error such as an undefined instruction
- attempting to execute an instruction from a memory region marked as Non-Executable (XN).
- a privilege violation or an attempt to access an unmanaged region causing an MPU fault

2.6.1 Fault Types

Table 2-10 shows the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates that the fault has occurred. See Configurable Fault Status Register on page 4-24 for more information about the fault status registers.

Table 2-10 Faults

Fault	Handler	Bit name	Fault status register
Bus error on a vector read	HardFault	VECTTBL	HardFault Status Register on Page 2-81
Fault escalated to a hard fault		FORCED	
MPU or default memory map mismatch:	MemManage	-	-
on instruction access		IACCVIOL ¹⁾	MemManage Fault Address Register on Page 2-82
on data access		DACCVIOL	
during exception stacking		MSTKERR	
during exception unstacking		MUNSKERR	
during lazy floating-point state preservation		MLSPERR	

Table 2-10 Faults (cont'd)

Fault	Handler	Bit name	Fault status register
Bus error:	BusFault	-	-
during exception stacking		STKERR	BusFault Status Register on Page 2-74
during exception unstacking		UNSTKERR	
during instruction prefetch		IBUSERR	
during lazy floating-point state preservation		LSPERR	
Precise data bus error		PRECISERR	
Imprecise data bus error		IMPRECISERR	
Attempt to access a coprocessor	UsageFault	NOCP	
Undefined instruction		UNDEFINSTR	
Attempt to enter an invalid instruction set state ²⁾		INVSTATE	
Invalid EXC_RETURN value		INVPC	
Illegal unaligned load or store		UNALIGNED	
Divide By 0		DIVBYZERO	

1) Occurs on an access to an XN region even if the processor does not include an MPU or the MPU is disabled.

2) Attempting to use an instruction set other than the Thumb instruction set or returns to a non load/store-multiple instruction with ICI continuation.

2.6.2 Fault Escalation and Hard Faults

All faults exceptions except for HardFault have configurable exception priority, see System Handler Priority Registers on page 4-21. Software can disable execution of the handlers for these faults, see System Handler Control and State Register on page 4-23.

Usually, the exception priority, together with the values of the exception mask registers, determines whether the processor enters the fault handler, and whether a fault handler can preempt another fault handler. as described in Exception model on [Page 2-26](#).

In some situations, a fault with configurable priority is treated as a HardFault. This is called priority escalation, and the fault is described as escalated to HardFault. Escalation to HardFault occurs when:

Central Processing Unit (CPU)

- A fault handler causes the same kind of fault as the one it is servicing. This escalation to HardFault occurs because a fault handler cannot preempt itself because it must have the same priority as the current priority level.
- A fault handler causes a fault with the same or lower priority as the fault it is servicing. This is because the handler for the new fault cannot preempt the currently executing fault handler.
- An exception handler causes a fault for which the priority is the same as or lower than the currently executing exception.
- A fault occurs and the handler for that fault is not enabled.

If a BusFault occurs during a stack push when entering a BusFault handler, the BusFault does not escalate to a HardFault. This means that if a corrupted stack causes a fault, the fault handler executes even though the stack push for the handler failed. The fault handler operates but the stack contents are corrupted.

Note: Only Reset and NMI can preempt the fixed priority HardFault. A HardFault can preempt any exception other than Reset, NMI, or another HardFault.

2.6.3 Fault Status Registers and Fault Address Registers

The fault status registers indicate the cause of a fault. For BusFaults and MemManage faults, the fault address register indicates the address accessed by the operation that caused the fault, as shown in [Table 2-11](#).

Table 2-11 Fault status and fault address registers

Handler	Status register name	Address register name	Register description
HardFault	HFSR	-	HardFault Status Register on Page 2-81
MemManage	MMFSR	MMFAR	MemManage Fault Status Register Page 2-74 MemManage Fault Address Register Page 2-82
BusFault	BFSR	BFAR	BusFault Status Register on Page 2-74 BusFault Address Register on Page 2-83
UsageFault	UFSR	-	UsageFault Status Register on Page 2-74

2.6.4 Lockup

The processor enters a lockup state if a fault occurs when executing the NMI or HardFault handlers. When the processor is in lockup state it does not execute any instructions. The processor remains in lockup state until either:

- it is reset

- an NMI occurs
- it is halted by a debugger

Note: If lockup state occurs from the NMI handler a subsequent NMI does not cause the processor to leave lockup state.

2.7 Power Management

The Cortex-M4 processor sleep modes reduce power consumption:

- Sleep mode stops the processor clock.
- Deep sleep mode stops the system clock and switches off the PLL and flash memory.

The SLEEPDEEP bit of the SCR selects which sleep mode is used, see System Control Register on [Page 2-67](#). For more information about the behavior of the sleep modes see section “Power Management” in SCU chapter.

The following section describes the mechanisms for entering sleep mode, and the conditions for waking up from sleep mode.

2.7.1 Entering Sleep Mode

This section describes the mechanisms software can use to put the processor into sleep mode

The system can generate spurious wakeup events, for example a debug operation wakes up the processor. Therefore software must be able to put the processor back into sleep mode after such an event. A program might have an idle loop to put the processor back to sleep mode.

Wait for interrupt

The wait for interrupt instruction, WFI, causes immediate entry to sleep mode unless the wake-up condition is true, see Wakeup from WFI or sleep-on-exit on [Page 2-41](#). When the processor executes a WFI instruction it stops executing instructions and enters sleep mode.

Wait for event

The wait for event instruction, WFE, causes entry to sleep mode depending on the value of a one-bit event register. When the processor executes a WFE instruction, it checks the value of the event register:

- 0 The processor stops executing instructions and enters sleep mode.
- 1 The processor clears the register to 0 and continues executing instructions without entering sleep mode.

If the event register is 1, this indicate that the processor must not enter sleep mode on execution of a WFE instruction. Typically, this is because an external event signal is

asserted, or a processor in the system has executed an SEV instruction, see SEV on page 3-166. Software cannot access this register directly.

Sleep-on-exit

If the SLEEPONEXIT bit of the SCR is set to 1, when the processor completes the execution of all exception handlers it returns to Thread mode and immediately enters sleep mode. Use this mechanism in applications that only require the processor to run when an exception occurs.

2.7.2 Wakeup from Sleep Mode

The conditions for the processor to wakeup depend on the mechanism that cause it to enter sleep mode.

Wakeup from WFI or sleep-on-exit

Normally, the processor wakes up only when it detects an exception with sufficient priority to cause exception entry. Some embedded systems might have to execute system restore tasks after the processor wakes up, and before it executes an interrupt handler. To achieve this set the PRIMASK bit to 1 and the FAULTMASK bit to 0. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor sets PRIMASK to zero. For more information about PRIMASK and FAULTMASK see Exception mask registers on [Page 2-13](#).

Wakeup from WFE

The processor wakes up if:

- it detects an exception with sufficient priority to cause exception entry
- it detects an external event signal, see The external event input

In addition, if the SEVONPEND bit in the SCR is set to 1, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause exception entry. For more information about the SCR see System Control Register on [Page 2-67](#).

2.7.3 The External Event Input

The processor provides an external event input signal. Peripherals can drive this signal, either to wake the processor from WFE, or to set the internal WFE event register to one to indicate that the processor must not enter sleep mode on a later WFE instruction. See Wait for event on [Page 2-40](#) for more information.

2.7.4 Power Management Programming Hints

ISO/IEC C cannot directly generate the WFI and WFE instructions. The CMSIS provides the following functions for these instructions:

```
void __WFE(void)    // Wait for Event
void __WFI(void)    // Wait for Interrupt
```

2.8 Private Peripherals

The following sections are the reference material for the ARM Cortex-M4 core peripherals.

2.8.1 About the Private Peripherals

The address map of the Private Peripheral Bus (PPB) is:

Table 2-12 Core peripheral register regions

Address	Core peripheral	Description
0xE000E008-0xE000E00F	System control block	Section 2.8.2 and Section 2.9.1
0xE000E010-0xE000E01F	System timer	Section 2.8.3 and Section 2.9.2
0xE000E100-0xE000E4EF	Nested Vectored Interrupt Controller	Section 2.8.4 and Section 2.9.3
0xE000ED00-0xE000ED3F	System control block	Section 2.8.2 and Section 2.9.1
0xE000ED90-0xE000EDB8	Memory protection unit	Section 2.8.5 and Section 2.9.4
0xE000EF00-0xE000EF03	Nested Vectored Interrupt Controller	Section 2.8.4 and Section 2.9.3
0xE000EF30-0xE000EF44	Floating Point Unit	Section 2.8.6 and Section 2.9.5

2.8.2 System control block

The System control block (SCB) provides system implementation information, and system control. This includes configuration, control, and reporting of the system exceptions. The system control block registers are:

2.8.2.1 System control block design hints and tips

Ensure software uses aligned accesses of the correct size to access the system control block registers:

- except for the CFSR and SHPR1-SHPR3, it must use aligned word accesses
- for the CFSR and SHPR1-SHPR3 it can use byte or aligned halfword or word accesses.

The processor does not support unaligned accesses to system control block registers.

In a fault handler, to determine the true faulting address:

1. Read and save the MMFAR or BFAR value.
2. Read the MMARVALID bit in the MMFSR, or the BFARVALID bit in the BFSR. The MMFAR or BFAR address is valid only if this bit is 1.

Software must follow this sequence because another higher priority exception might change the MMFAR or BFAR value. For example, if a higher priority handler preempts the current fault handler, the other fault might change the MMFAR or BFAR value.

2.8.3 System timer, SysTick

The processor has a 24-bit system timer, SysTick, that counts down from the reload value to zero, reloads, that is wraps to, the value in the **SYST_RVR** register on the next clock edge, then counts down on subsequent clocks.

Note: When the processor is halted for debugging the counter does not decrement.

2.8.3.1 SysTick design hints and tips

The SysTick counter runs on the clock selected by **SYST_CSR.CLKSOURCE**. If the selected clock signal is stopped, the SysTick counter stops.

Ensure software uses aligned word accesses to access the SysTick registers.

The SysTick counter reload and current value are undefined at reset, the correct initialization sequence for the SysTick counter is:

1. Program reload value.
2. Clear current value.
3. Program Control and Status register.

2.8.4 Nested Vectored Interrupt Controller (NVIC)

This section describes the NVIC and the registers it uses. The XMC4300 NVIC supports:

- 112 interrupts.
- A programmable priority level of 0-63 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Level and pulse detection of interrupt signals.
- Dynamic reprioritization of interrupts.

- Grouping of priority values into group priority and subpriority fields.
- Interrupt tail-chaining.
- An external Non-maskable interrupt (NMI)

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead. This provides low latency exception handling. The hardware implementation of the NVIC registers is:

2.8.4.1 Level-sensitive and pulse interrupts

The processor supports both level-sensitive and pulse interrupts. Pulse interrupts are also described as edge-triggered interrupts.

A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Typically this happens because the ISR accesses the peripheral, causing it to clear the interrupt request. A pulse interrupt is an interrupt signal sampled synchronously on the rising edge of the processor clock. To ensure the NVIC detects the interrupt, the peripheral must assert the interrupt signal for at least one clock cycle, during which the NVIC detects the pulse and latches the interrupt.

When the processor enters the ISR, it automatically removes the pending state from the interrupt, see next section. For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. This means that the peripheral can hold the interrupt signal asserted until it no longer requires servicing.

See section “Service Request Distribution” in the “Service Request Processing” chapter for details about which interrupts are level-based and which are pulsed.

Hardware and software control of interrupts

The Cortex-M4 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

- the NVIC detects that the interrupt signal is HIGH and the interrupt is not active
- the NVIC detects a rising edge on the interrupt signal
- software writes to the corresponding interrupt set-pending register bit, see Interrupt Set-pending Registers on [Page 2-89](#) or to the STIR to make an interrupt pending, see Software Trigger Interrupt Register on [Page 2-92](#).

A pending interrupt remains pending until one of the following:

- The processor enters the ISR for the interrupt. This changes the state of the interrupt from pending to active. Then:
 - For a level-sensitive interrupt, when the processor returns from the ISR, the NVIC samples the interrupt signal. If the signal is asserted, the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.

Central Processing Unit (CPU)

- For a pulse interrupt, the NVIC continues to monitor the interrupt signal, and if this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. If the interrupt signal is not pulsed while the processor is in the ISR, when the processor returns from the ISR the state of the interrupt changes to inactive.
- Software writes to the corresponding interrupt clear-pending register bit. For a level-sensitive interrupt, if the interrupt signal is still asserted, the state of the interrupt does not change. Otherwise, the state of the interrupt changes to inactive. For a pulse interrupt, state of the interrupt changes to:
 - inactive, if the state was pending
 - active, if the state was active and pending.

2.8.4.2 NVIC design hints and tips

Ensure software uses correctly aligned register accesses. The processor does not support unaligned accesses to NVIC registers. See the individual register descriptions for the supported access sizes.

A interrupt can enter pending state even if it is disabled. Disabling an interrupt only prevents the processor from taking that interrupt.

Before programming VTOR to relocate the vector table, ensure the vector table entries of the new vector table are setup for fault handlers, NMI and all enabled exception like interrupts. For more information see Vector Table Offset Register on [Page 2-63](#).

2.8.4.3 Using CMSIS functions to access NVIC

CMSIS functions enable software portability between different Cortex-M profile processors. To ensure Cortex-M portability, use the functions marked for Cortex-M portability in the table below.

CMSIS provides a number of functions for NVIC control, including:

Table 2-13 CMSIS functions for NVIC control

CMSIS interrupt control function	Description	Cortex-M Portable
<code>void NVIC_SetPriorityGrouping(uint32_t priority_grouping)</code>	Set the priority grouping.	No
<code>uint32_t NVIC_GetPriorityGrouping(void)</code>	Get the priority grouping.	No
<code>void NVIC_EnableIRQ(IRQn_t IRQn)</code>	Enables IRQn.	Yes

Table 2-13 CMSIS functions for NVIC control (cont'd)

CMSIS interrupt control function	Description	Cortex-M Portable
<code>void NVIC_DisableIRQ(IRQn_t IRQn)</code>	Disables IRQn.	Yes
<code>uint32_t NVIC_GetPendingIRQ(IRQn_t IRQn)</code>	Return IRQ-Number (true) if IRQn is pending.	Yes
<code>void NVIC_SetPendingIRQ(IRQn_t IRQn)</code>	Set IRQn pending.	Yes
<code>void NVIC_ClearPendingIRQ(IRQn_t IRQn)</code>	Clear IRQn pending.	Yes
<code>uint32_t NVIC_GetActive(IRQn_t IRQn)</code>	Return the IRQ number of the active interrupt.	No
<code>void NVIC_SetPriority(IRQn_t IRQn, uint32_t priority)</code>	Set priority for IRQn.	Yes
<code>uint32_t NVIC_GetPriority(IRQn_t IRQn)</code>	Read priority of IRQn.	Yes
<code>uint32_t NVIC_EncodePriority(uint32_t PriorityGroup, uint32_t PreemptPriority, uint32_t SubPriority)</code>	Encodes the priority for an interrupt with the given priority group, preemptive priority value and sub priority value.	No
<code>void NVIC_DecodePriority(uint32_t Priority, uint32_t PriorityGroup, uint32_t* pPreemptPriority, uint32_t* pSubPriority)</code>	Decodes an interrupt priority value with the given priority group to preemptive priority value and sub priority value.	No
<code>void NVIC_SystemReset(void)</code>	Reset the system	Yes

The parameter IRQn is the IRQ number, see [Table 2-8](#) on [Page 2-28](#). For more information about these functions see the CMSIS documentation [\[4\]](#).

2.8.5 Memory Protection Unit (MPU)

The MPU divides the memory map into a number of regions, and defines the location, size, access permissions, and memory attributes of each region. It supports:

- independent attribute settings for each region
- overlapping regions
- export of memory attributes to the system

Central Processing Unit (CPU)

The memory attributes affect the behavior of memory accesses to the region. The Cortex-M4 MPU defines:

- eight separate memory regions, 0-7
- a background region

When memory regions overlap, a memory access is affected by the attributes of the region with the highest number. For example, the attributes for region 7 take precedence over the attributes of any region that overlaps region 7.

The background region has the same memory access attributes as the default memory map, but is accessible from privileged software only.

The Cortex-M4 MPU memory map is unified. This means instruction accesses and data accesses have same region settings.

If a program accesses a memory location that is prohibited by the MPU, the processor generates a MemManage fault. This causes a fault exception, and might cause termination of the process in an OS environment.

In an OS environment, the kernel can update the MPU region setting dynamically based on the process to be executed. Typically, an embedded OS uses the MPU for memory protection.

Configuration of MPU regions is based on memory types, see Memory regions, types and attributes on [Page 2-20](#).

Table 2-14 shows the possible MPU region attributes.

Note: The shareability and cache attributes are not relevant to the XMC4300.

Table 2-14 Memory attributes summary

Address	Shareability	Other attributes	Description
Strongly-ordered	-	-	All accesses to Strongly-ordered memory occur in program order. All Strongly-ordered regions are assumed to be shared.
Device	Shared	-	Memory-mapped peripherals that several processors share.
	Non-shared	-	Memory-mapped peripherals that only a single processor uses.
Normal	Shared	Non-cacheable Write-through or Write-back Cacheable	Normal memory that is shared between several processors.
	Non-shared	Non-cacheable Write-through or Write-back Cacheable	Normal memory that only a single processor uses.

2.8.5.1 MPU Access Permission Attributes

This section describes the MPU access permission attributes. The access permission bits, TEX, C, B, S, AP, and XN, of the RASR, control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault. [Table 2-15](#) shows encodings for the TEX, C, B, and S access permission bits.

Table 2-15 TEX, C, B, and S encoding

TEX	C	B	S	Memory type	Shareability	Other attributes
0b000	0	0	x	Strongly-ordered	Shareable	-
		1	x	Device	Shareable	-
	1	0	0	Normal	Not shareable	Outer and inner write-through. No write allocate.
			1		Shareable	
		1	0	Normal	Not shareable	
					1	
0b001	0	0	0	Normal	Not shareable	Outer and inner noncacheable.
					1	
		1	x ¹⁾	Reserved encoding		
	1	0	x ¹⁾	Implementation defined attributes.		
		1	0	Normal	Not shareable	Outer and inner write-back. Write and read allocate.
					1	
1	x	x ¹⁾	Reserved encoding		-	
0b010	0	0	x ¹⁾	Device	Not shareable	Nonshared Device.
		1	x ¹⁾	Reserved encoding		-
	1	x	x ¹⁾	Reserved encoding		-
0b1BB	A	A	0	Normal	Not shareable	Cached memory, BB = outer policy, AA = inner policy. See Table 2-16 on Page 2-49 for the encoding of the AA and BB bits.
	A		1		Shareable	

1) The MPU ignores the value of this bit.

Central Processing Unit (CPU)

Table 2-16 shows the cache policy for memory attribute encodings with a TEX value is in the range 4-7.

Table 2-16 Cache policy for memory attribute encoding

Encoding, AA or BB	Corresponding cache policy
00	Non-cacheable
01	Write back, write and read allocate
10	Write through, no write allocate
11	Write back, no write allocate

MPU configuration for the XMC4300

The XMC4300 has only a single processor and no caches. However to enable portability it is recommended to program the MPU as follows:

Table 2-17 Memory region attributes for a microcontroller

Memory region	TEX	C	B	S	Memory type and attributes
Internal Flash memory	0b000	1	0	0	Normal memory, Non-shareable, write-through
Internal SRAM memories	0b000	1	0	1	Normal memory, Shareable, write-through
External memories	0b000	1	1	1	Normal memory, Shareable, write-back, write-allocate
Peripherals	0b000	0	1	1	Device memory, Shareable

Table 2-18 shows the AP encodings that define the access permissions for privileged and unprivileged software.

Table 2-18 AP encoding

AP[2:0]	Privileged permissions	Unprivileged permissions	Description
000	No access	No access	All accesses generate a permission fault
001	rw	No access	Access from privileged software only
010	rw	r	Writes by unprivileged software generate a permission fault
011	rw	rw	Full access

Table 2-18 AP encoding (cont'd)

AP[2:0]	Privileged permissions	Unprivileged permissions	Description
100	Unpredictable	Unpredictable	Reserved
101	r	No access	Reads by privileged software only
110	r	r	Read only, by privileged or unprivileged software
111	r	r	Read only, by privileged or unprivileged software

2.8.5.2 MPU Mismatch

When an access violates the MPU permissions, the processor generates a MemManage fault, see Exceptions and interrupts on [Page 2-17](#). The MMFSR indicates the cause of the fault. See MemManage Fault Status Register on [Page 2-74](#) for more information.

2.8.5.3 Updating an MPU Region

To update the attributes for an MPU region, update the MPU_RNR, MPU_RBAR and MPU_RASR registers. You can program each register separately, or use a multiple-word write to program all of these registers. You can use the MPU_RBAR and MPU_RASR aliases to program up to four regions simultaneously using an STM instruction.

Updating an MPU region using separate words

Simple code to configure one region:

```

; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,=MPU_RNR           ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]       ; Region Number
STR R4, [R0, #0x4]       ; Region Base Address
STRH R2, [R0, #0x8]      ; Region Size and Enable
STRH R3, [R0, #0xA]      ; Region Attribute

```

Disable a region before writing new region settings to the MPU if you have previously enabled the region being changed. For example:

```

; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address

```

Central Processing Unit (CPU)

```
LDR R0, =MPU_RNR           ; 0xE000ED98, MPU region number
register
STR R1, [R0, #0x0]        ; Region Number
BIC R2, R2, #1            ; Disable
STRH R2, [R0, #0x8]       ; Region Size and Enable
STR R4, [R0, #0x4]        ; Region Base Address
STRH R3, [R0, #0xA]       ; Region Attribute
ORR R2, #1                ; Enable
STRH R2, [R0, #0x8]       ; Region Size and Enable
```

Software must use memory barrier instructions:

- before MPU setup if there might be outstanding memory transfers, such as buffered writes, that might be affected by the change in MPU settings
- after MPU setup if it includes memory transfers that must use the new MPU settings.

However, memory barrier instructions are not required if the MPU setup process starts by entering an exception handler, or is followed by an exception return, because the exception entry and exception return mechanism cause memory barrier behavior.

Software does not require any memory barrier instructions during MPU setup, because it accesses the MPU through the PPB, which is a Strongly-Ordered memory region.

For example, if you want all of the memory access behavior to take effect immediately after the programming sequence, use a DSB instruction and an ISB instruction. A DSB is required after changing MPU settings, such as at the end of context switch. An ISB is required if the code that programs the MPU region or regions is entered using a branch or call. If the programming sequence is entered using a return from exception, or by taking an exception, then you do not require an ISB.

Updating an MPU region using multi-word writes

You can program directly using multi-word writes, depending on how the information is divided. Consider the following reprogramming:

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =MPU_RNR           ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]         ; Region Number
STR R2, [R0, #0x4]         ; Region Base Address
STR R3, [R0, #0x8]         ; Region Attribute, Size and Enable
```

Use an STM instruction to optimize this:

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =MPU_RNR           ; 0xE000ED98, MPU region number register
```


Central Processing Unit (CPU)

```
STM R0, {R1-R3}      ; Region Number, address, attribute, size and  
enable
```

You can do this in two words for pre-packed information. This means that the MPU_RBAR contains the required region number and had the VALID bit set to 1, see MPU Region Base Address Register on [Page 2-96](#). Use this when the data is statically packed, for example in a boot loader:

```
; R1 = address and region number in one  
; R2 = size and attributes in one  
LDR R0, =MPU_RBAR    ; 0xE000ED9C, MPU Region Base register  
STR R1, [R0, #0x0]   ; Region base address and  
                    ; region number combined with VALID (bit 4)  
set to 1  
STR R2, [R0, #0x4]   ; Region Attribute, Size and Enable
```

Subregions

Regions of 256 bytes or more are divided into eight equal-sized subregions. Set the corresponding bit in the SRD field of the MPU_RASR to disable a subregion, see MPU Region Attribute and Size Register on [Page 2-98](#). The least significant bit of SRD controls the first subregion, and the most significant bit controls the last subregion. Disabling a subregion means another region overlapping the disabled range matches instead. If no other enabled region overlaps the disabled subregion the MPU issues a fault.

Regions of 32, 64, and 128 bytes do not support subregions, With regions of these sizes, you must set the SRD field to 0x00, otherwise the MPU behavior is Unpredictable.

Example of SRD use

Two regions with the same base address overlap. Region one is 128KB, and region two is 512KB. To ensure the attributes from region one apply to the first 128KB region, set the SRD field for region two to 0b00000011 to disable the first two subregions, as the figure shows.

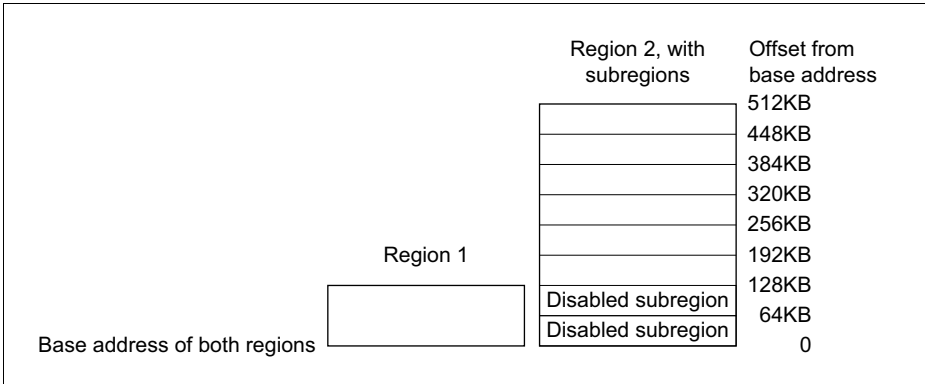


Figure 2-8 Example of SRD use

2.8.5.4 MPU Design Hints and Tips

To avoid unexpected behavior, disable the interrupts before updating the attributes of a region that the interrupt handlers might access.

Ensure software uses aligned accesses of the correct size to access MPU registers:

- except for the MPU_RASR, it must use aligned word accesses
- for the MPU_RASR it can use byte or aligned halfword or word accesses.

The processor does not support unaligned accesses to MPU registers.

When setting up the MPU, and if the MPU has previously been programmed, disable unused regions to prevent any previous region settings from affecting the new MPU setup.

In the XMC4300 the shareability and cache policy attributes do not affect the system behavior. However, using these settings for the MPU regions can make the application code more portable.

2.8.6 Floating Point Unit (FPU)

The Cortex-M4 FPU implements the FPv4-SP floating-point extension.

The FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions.

The FPU provides floating-point computation functionality that is compliant with the ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic, referred to as the IEEE 754 standard [4].

The FPU contains 32 single-precision extension registers, which you can also access as 16 doubleword registers for load, store, and move operations.

2.8.6.1 Enabling the FPU

The FPU is disabled from reset. You must enable it before you can use any floating-point instructions. The Example shows an example code sequence for enabling the FPU in both privileged and user modes. The processor must be in privileged mode to read from and write to the CPACR.

Example: Enabling the FPU

```

; CPACR is located at address 0xE000ED88
LDR.W  R0, =0xE000ED88
; Read CPACR
LDR    R1, [R0]
; Set bits 20-23 to enable CP10 and CP11 coprocessors
ORR    R1, R1, #(0xF << 20)
; Write back the modified value to the CPACR
STR    R1, [R0]; wait for store to complete
DSB
;reset pipeline now the FPU is enabled
ISB

```

2.9 PPB Registers

The CPU private peripherals registers base address is E000E000_H.

Table 2-19 Registers Overview

Register Short Name	Register Long Name	Offset Address	Access Mode		Description see
			Read	Write	
SCS					
ACTLR	Auxiliary Control Register	008 _H	PV, 32	PV, 32	Page 2-58
CPUID	CPUID Base Register	D00 _H	PV, 32	PV, 32	Page 2-60
ICSR	Interrupt Control and State Register	D04 _H	PV, 32	PV, 32	Page 2-60
VTOR	Vector Table Offset Register	D08 _H	PV, 32	PV, 32	Page 2-63
AIRCR	Application Interrupt and Reset Control Register	D0C _H	PV, 32	PV, 32	Page 2-64
SCR	System Control Register	D10 _H	PV, 32	PV, 32	Page 2-67

Central Processing Unit (CPU)

Table 2-19 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Address	Access Mode		Description see
			Read	Write	
CCR	Configuration and Control Register	D14 _H	PV, 32	PV, 32	Page 2-68
SHPR1	System Handler Priority Register 1	D18 _H	PV, 32	PV, 32	Page 2-71
SHPR2	System Handler Priority Register 2	D1C _H	PV, 32	PV, 32	Page 2-71
SHPR3	System Handler Priority Register 3	D20 _H	PV, 32	PV, 32	Page 2-72
SHCRS	System Handler Control and State Register	D24 _H	PV, 32	PV, 32	Page 2-72
CFSR	Configurable Fault Status Register	D28 _H	PV, 32	PV, 32	Page 2-74
MMSR ¹⁾	MemManage Fault Status Register	D28 _H	PV, 32	PV, 32	Page 2-74
BFSR ¹⁾	BusFault Status Register	D29 _H	PV, 32	PV, 32	Page 2-74
UFSR ¹⁾	UsageFault Status Register	D2A _H	PV, 32	PV, 32	Page 2-74
HFSR	HardFault Status Register	D2C _H	PV, 32	PV, 32	Page 2-81
MMAR	MemManage Fault Address Register	D34 _H	PV, 32	PV, 32	Page 2-82
BFAR	BusFault Address Register	D38 _H	PV, 32	PV, 32	Page 2-83
AFSR	Auxiliary Fault Status Register	D3C _H	PV, 32	PV, 32	Page 2-83
SysTick					
SYST_CSR	SysTick Control and Status Register	010 _H	PV, 32	PV, 32	Page 2-84
SYST_RVR	SysTick Reload Value Register	014 _H	PV, 32	PV, 32	Page 2-85
SYST_CVR	SysTick Current Value Register	018 _H	PV, 32	PV, 32	Page 2-86

Table 2-19 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Address	Access Mode		Description see
			Read	Write	
SYST_CALIB	SysTick Calibration Value Register	01C _H	PV, 32	-	Page 2-86
NVIC					
NVIC_ISER0-NVIC_ISER3	Interrupt Set-enable Registers	100 _H	PV, 32	PV, 32	Page 2-87
NVIC_ICER0-NVIC_ICER3	Interrupt Clear-enable Registers	180 _H	PV, 32	PV, 32	Page 2-88
NVIC_ISPR0-NVIC_ISPR3	Interrupt Set-pending Registers	200 _H	PV, 32	PV, 32	Page 2-89
NVIC_ICPR0-NVIC_ICPR3	Interrupt Clear-pending Registers	280 _H	PV, 32	PV, 32	Page 2-89
NVIC_IABR0-NVIC_IABR3	Interrupt Active Bit Registers	300 _H	PV, 32	PV, 32	Page 2-90
NVIC_IPR0-NVIC_IPR27	Interrupt Priority Registers	400 _H	PV, 32	PV, 32	Page 2-90
STIR	Software Trigger Interrupt Register	F00 _H	Configurable ²⁾		Page 2-92
MPU					
MPU_TYPE	MPU Type Register	D90 _H	PV, 32	PV, 32	Page 2-93
MPU_CTRL	MPU Control Register	D94 _H	PV, 32	PV, 32	Page 2-93
MPU_RNR	MPU Region Number Register	D98 _H	PV, 32	PV, 32	Page 2-96
MPU_RBAR	MPU Region Base Address Register	D9C _H	PV, 32	PV, 32	Page 2-96
MPU_RASR	MPU Region Attribute and Size Register	DA0 _H	PV, 32	PV, 32	Page 2-98
MPU_RBAR_A1	Alias of RBAR, see MPU Region Base Address Register	DA4 _H	PV, 32	PV, 32	Page 2-96
MPU_RASR_A1	Alias of RASR, see MPU Region Attribute and Size Register	DA8 _H	PV, 32	PV, 32	Page 2-98

Central Processing Unit (CPU)

Table 2-19 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Address	Access Mode		Description see
			Read	Write	
MPU_RBAR_A2	Alias of RBAR, see MPU Region Base Address Register	DAC _H	PV, 32	PV, 32	Page 2-96
MPU_RASR_A2	Alias of RASR, see MPU Region Attribute and Size Register	DB0 _H	PV, 32	PV, 32	Page 2-98
MPU_RBAR_A3	Alias of RBAR, see MPU Region Base Address Register	DB4 _H	PV, 32	PV, 32	Page 2-96
MPU_RASR_A3	Alias of RASR, see MPU Region Attribute and Size Register	DB8 _H	PV, 32	PV, 32	Page 2-98

FPU

CPACR	Coprocessor Access Control Register	D88 _H	PV, 32	PV, 32	Page 2-101
FPCCR	Floating-point Context Control Register	F34 _H	U, PV, 32	U, PV, 32	Page 2-102
FPCAR	Floating-point Context Address Register	F38 _H	U, PV, 32	U, PV, 32	Page 2-104
FPSCR	Floating-point Status Control Register	-	U, PV, 32	U, PV, 32	Page 2-105
FPDSCR	Floating-point Default Status Control Register	F3C _H	U, PV, 32	U, PV, 32	Page 2-107
MVFR0	Media and FP Feature Register 0	F40 _H	U, PV, 32	U, PV, 32	Page 2-107
MVFR1	Media and FP Feature Register 1	F44 _H	U, PV, 32	U, PV, 32	Page 2-109
Reserved	Unused address space	All gaps	nBE	nBE	

1) A subregister of the CFSR.

2) See the register description for more information.

2.9.1 SCS Registers

Auxiliary Control Register

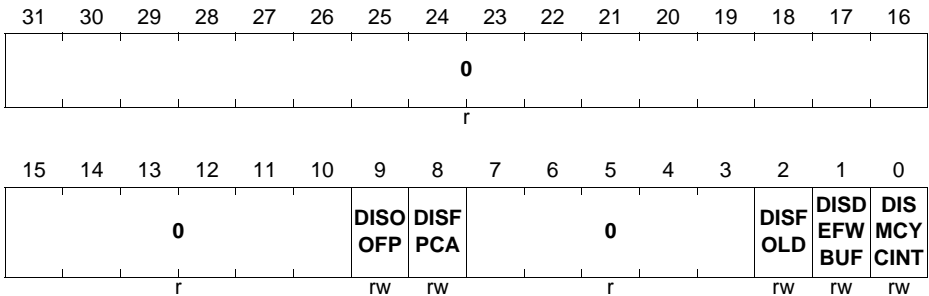
The ACTLR provides disable bits for the following processor functions:

- IT folding
- write buffer use for accesses to the default memory map
- interruption of multi-cycle instructions.

By default this register is set to provide optimum performance from the Cortex-M4 processor, and does not normally require modification.

ACTLR

Auxiliary Control Register (E000 E008_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
DISMCYCINT	0	rw	<p>Disable load/store multiple</p> <p>When set to 1, disables interruption of load multiple and store multiple instructions. This increases the interrupt latency of the processor because any LDM or STM must complete before the processor can stack the current state and enter the interrupt handler.</p>

Central Processing Unit (CPU)

Field	Bits	Type	Description
DISDEFWBUF	1	rw	<p>Disable write buffer</p> <p>When set to 1, disables write buffer use during default memory map accesses. This causes all BusFaults to be precise BusFaults but decreases performance because any store to memory must complete before the processor can execute the next instruction.</p> <p><i>Note: This bit only affects write buffers implemented in the Cortex-M4 processor.</i></p>
DISFOLD	2	rw	<p>Disable IT folding</p> <p>When set to 1, disables IT folding.</p>
DISFPCA	8	rw	<p>Disable FPCA update</p> <p>Disable automatic update of CONTROL.FPCA.</p>
DISOFP	9	rw	<p>Disable out of order FP execution</p> <p>Disables floating point instructions completing out of order with respect to integer instructions.</p>
0	[31:10], [7:3]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

About IT folding

In some situations, the processor can start executing the first instruction in an IT block while it is still executing the IT instruction. This behavior is called IT folding, and improves performance. However, IT folding can cause jitter in looping. If a task must avoid jitter, set the DISFOLD bit to 1 before executing the task, to disable IT folding.

Central Processing Unit (CPU)

CPUID Base Register

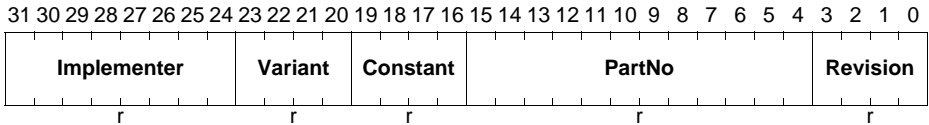
The CPUID register contains the processor part number, version, and implementation information.

CPUID

CPUID Base Register

(E000 ED00_H)

Reset Value: 410F C241_H



Field	Bits	Type	Description
Revision	[3:0]	r	Revision number the y value in the “rxy” product revision identifier 1 _H Patch 1
PartNo	[15:4]	r	Part number of the processor C24 _H Cortex-M4
Constant	[19:16]	r	Reads as 0xF
Variant	[23:20]	r	Variant number the x value in the “rxy” product revision identifier 0 _H Revision 0
Implementer	[31:24]	r	Implementer code 41 _H ARM

Interrupt Control and State Register

The ICSR:

- provides:
 - a set-pending bit for the Non-Maskable Interrupt (NMI) exception
 - set-pending and clear-pending bits for the PendSV and SysTick exceptions
- indicates:
 - the exception number of the exception being processed
 - whether there are preempted active exceptions
 - the exception number of the highest priority pending exception
 - whether any interrupts are pending.

ICSR

Interrupt Control and State Register

(E000 ED04_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NMI PEN DSE T	0		PEN DSV SET	PEN DSV CLR	PEN DST SET	PEN DST CLR	0	Res	ISRP ENDI NG	0			VECTPEN DING		
r/w	r		r/w	w	r/w	w	r	r	r	r			r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VECTPENDING				RET TOB ASE	0	VECTACTIVE									
r				r	r	r									

Field	Bits	Type	Description
VECTACTIVE ¹⁾	[8:0]	r	<p>Active exception number</p> <p>00_H Thread mode</p> <p>Nonzero = The exception number of the currently active exception.</p> <p><i>Note: Subtract 16 from this value to obtain the CMSIS IRQ number required to index into the Interrupt Clear-Enable, Set-Enable, Clear-Pending, Set-Pending, or Priority Registers.</i></p>
RETTOBASE	11	r	<p>Return to Base</p> <p>Indicates whether there are preempted active exceptions:</p> <p>0_B there are preempted active exceptions to execute</p> <p>1_B there are no active exceptions, or the currently-executing exception is the only active exception.</p>

Central Processing Unit (CPU)

Field	Bits	Type	Description
VECTPENDING	[17:12]	r	<p>Vector Pending</p> <p>Indicates the exception number of the highest priority pending enabled exception:</p> <p>0_H no pending exceptions</p> <p>Nonzero = the exception number of the highest priority pending enabled exception.</p> <p>The value indicated by this field includes the effect of the BASEPRI and FAULTMASK registers, but not any effect of the PRIMASK register.</p>
ISR_PENDING	22	r	<p>Interrupt pending flag</p> <p>excluding NMI and Faults:</p> <p>0_B interrupt not pending</p> <p>1_B interrupt pending.</p>
Res	23	r	<p>Reserved</p> <p>This bit is reserved for Debug use and reads-as-zero when the processor is not in Debug.</p>
PENDSTCLR	25	w	<p>SysTick exception clear-pending bit</p> <p>0_B no effect</p> <p>1_B removes the pending state from the SysTick exception.</p> <p><i>Note: This bit is w. On a register read its value is Unknown.</i></p>
PENDSTSET	26	rw	<p>SysTick exception set-pending bit</p> <p>0_B Write: no effect</p> <p>Read: SysTick exception is not pending</p> <p>1_B Write: changes SysTick exception state to pending</p> <p>Read: SysTick exception is pending</p>
PENDSVCLR	27	w	<p>PendSV clear-pending bit</p> <p>0_B no effect</p> <p>1_B removes the pending state from the PendSV exception.</p> <p><i>Note: This bit is w. On a register read its value is Unknown.</i></p>

Central Processing Unit (CPU)

Field	Bits	Type	Description
PENDSVSET	28	rw	PendSV set-pending bit 0 _B Write: no effect Read: PendSV exception is not pending 1 _B Write: changes PendSV exception state to pending Read: PendSV exception is pending Writing 1 to this bit is the only way to set the PendSV exception state to pending.
NMIPENDSET	31	rw	NMI set-pending bit 0 _B Write: no effect Read: NMI exception is not pending 1 _B Write: changes NMI exception state to pending Read: NMI exception is pending Because NMI is the highest-priority exception, normally the processor enter the NMI exception handler as soon as it registers a write of 1 to this bit, and entering the handler clears this bit to 0. A read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.
0	[30:29], 24, [21:18], [10:9]	r	Reserved Read as 0; should be written with 0.

1) This is the same value as IPSR bits[8:0], see Interrupt Program Status Register on page 2-6.

When you write to the ICSR, the effect is Unpredictable if you:

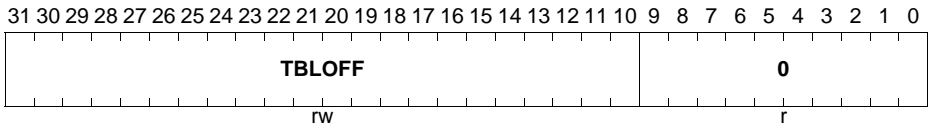
*Note: write 1 to the PENDSVSET bit and write 1 to the PENDSVCLR bit
write 1 to the PENDSTSET bit and write 1 to the PENDSTCLR bit.*

Vector Table Offset Register

The VTOR indicates the offset of the vector table base address from memory address 0x00000000.

VTOR

Vector Table Offset Register (E000 ED08_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TBLOFF	[31:10]	rw	<p>Vector table base offset field It contains bits[29:10] of the offset of the table base from the bottom of the memory map.</p> <p><i>Note: Bit[29] determines whether the vector table is in the code or SRAM memory region:</i> 0 = code 1 = SRAM Bit[29] is sometimes called the TBLBASE bit.</p>
0	[9:0]	r	<p>Reserved Read as 0; should be written with 0.</p>

When setting TBLOFF, you must align the offset to the number of exception entries in the vector table. The XMC4300 provides 112 interrupt nodes - minimum alignment is therefore 256 words, enough for up to 128 interrupts.

Notes

1. XMC4300 implements 112 interrupts, the remaining nodes to 128 are not used.
2. Table alignment requirements mean that bits[9:0] of the table offset must always be zero.

Application Interrupt and Reset Control Register

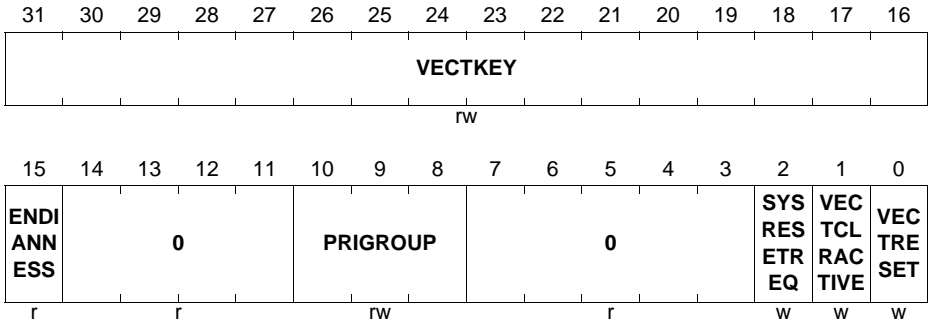
The AIRCR provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system.

To write to this register, you must write 0x5FA to the VECTKEY field, otherwise the processor ignores the write.

AIRCR

Application Interrupt and Reset Control Register
(E000 ED0C_H)

Reset Value: FA05 0000_H



Field	Bits	Type	Description
VECTRESET	0	w	Reserved for Debug use. This bit reads as 0. When writing to the register you must write 0 to this bit, otherwise behavior is Unpredictable
VECTCLRACTIVE	1	w	Reserved for Debug use. This bit reads as 0. When writing to the register you must write 0 to this bit, otherwise behavior is Unpredictable.
SYSRESETREQ	2	w	System reset request 0 _B no system reset request 1 _B asserts a signal to the outer system that requests a reset. This is intended to force a large system reset of all major components except for debug. This bit reads as 0.
PRIGROUP	[10:8]	rw	Interrupt priority grouping field This field determines the split of group priority from subpriority, see Binary point on Page 2-66 .
ENDIANNESS	15	r	Data endianness bit 0 _B Little-endian 1 _B Big-endian.

Central Processing Unit (CPU)

Field	Bits	Type	Description
VECTKEY	[31:16]	rw	Register key Read: = VECTKEY, reads as 0xFA05 Write: = VECTKEYSTAT, On writes, write 0x5FA to VECTKEY, otherwise the write is ignored.
0	[14:11], [7:3]	r	Reserved Read as 0; should be written with 0.

Binary point

The PRIGROUP field indicates the position of the binary point that splits the PRI_n fields in the Interrupt Priority Registers into separate group priority and subpriority fields. **Table 2-20** shows how the PRIGROUP value controls this split.

Table 2-20 Priority grouping

Interrupt priority level value, PRI_N[7:0]				Number of	
PRIGROUP	Binary point ¹⁾	Group priority bits	Subpriority bits	Group priorities	Sub-priorities
0b000	bxxxxxx0.0	[7:2]	None	64	1
0b001	bxxxxxx.00	[7:2]	None	64	1
0b010	bxxxxx.y00	[7:3]	[2]	32	2
0b011	bxxxx.yy00	[7:4]	[3:2]	16	4
0b100	bxxx.yyy00	[7:5]	[4:2]	8	8
0b101	bxx.yyyy00	[7:6]	[5:2]	4	16
0b110	bx.yyyyy00	7	[6:2]	2	32
0b111	b.yyyyyy00	None	[7:2]	1	64

1) PRI_n[7:0] field showing the binary point. x denotes a group priority field bit, and y denotes a subpriority field bit.

Note: Determining preemption of an exception uses only the group priority field, see Interrupt Priority Grouping on [Page 2-31](#).

Central Processing Unit (CPU)

System Control Register

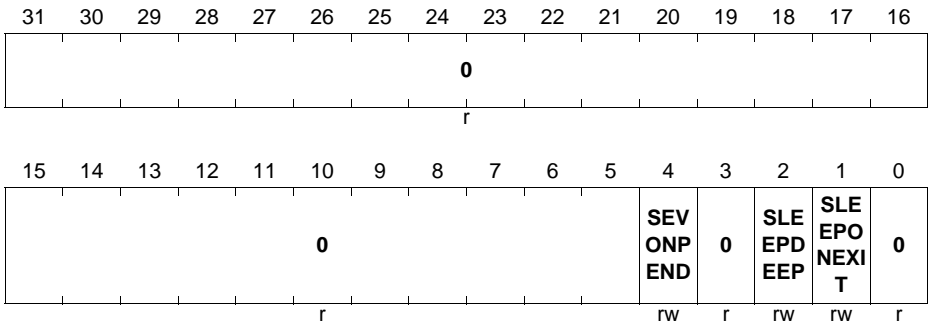
The SCR controls features of entry to and exit from low power state.

SCR

System Control Register

(E000 ED10_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SLEEPONEXIT	1	rw	<p>Sleep on Exit</p> <p>Indicates sleep-on-exit when returning from Handler mode to Thread mode:</p> <p>0_B do not sleep when returning to Thread mode.</p> <p>1_B enter sleep, or deep sleep, on return from an ISR.</p> <p>Setting this bit to 1 enables an interrupt driven application to avoid returning to an empty main application.</p>
SLEEPDEEP	2	rw	<p>Sleep or Deep Sleep</p> <p>Controls whether the processor uses sleep or deep sleep as its low power mode:</p> <p>0_B sleep</p> <p>1_B deep sleep</p>

Central Processing Unit (CPU)

Field	Bits	Type	Description
SEVONPEND	4	rw	Send Event on Pending bit: 0 _B only enabled interrupts or events can wakeup the processor, disabled interrupts are excluded 1 _B enabled events and all interrupts, including disabled interrupts, can wakeup the processor. When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE. The processor also wakes up on execution of an SEV instruction or an external event.
0	[31:5], 3, 0	r	Reserved Read as 0; should be written with 0.

Configuration and Control Register

The CCR controls entry to Thread mode and enables:

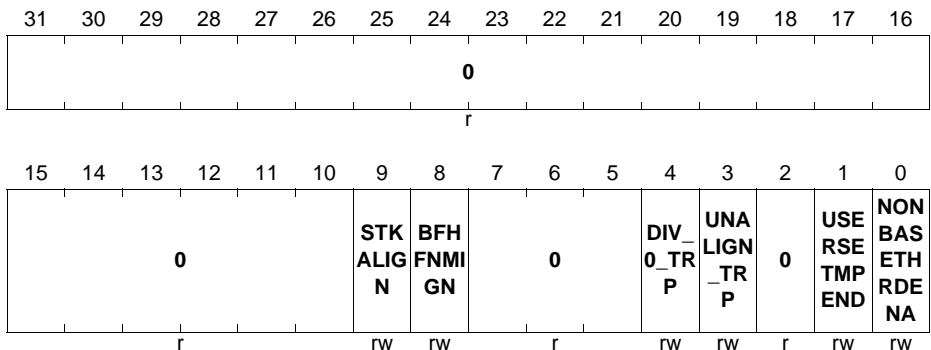
- the handlers for NMI, hard fault and faults escalated by FAULTMASK to ignore BusFaults
- trapping of divide by zero and unaligned accesses
- access to the STIR by unprivileged software, see [Software Trigger Interrupt Register](#) on [Page 2-92](#)

CCR

Configuration and Control Register

(E000 ED14_H)

Reset Value: 0000 0200_H



Central Processing Unit (CPU)

Field	Bits	Type	Description
NONBASETHR DENA	0	rw	Non Base Thread Mode Enable Indicates how the processor enters Thread mode: 0 _B processor can enter Thread mode only when no exception is active. 1 _B processor can enter Thread mode from any level under the control of an EXC_RETURN value, see Exception return .
USERSETMPE ND	1	rw	User Set Pending Enable Enables unprivileged software access to the STIR, see Software Trigger Interrupt Register . 0 _B disable 1 _B enable
UNALIGN_TRP	3	rw	Unaligned Access Trap Enable Enables unaligned access traps: 0 _B do not trap unaligned halfword and word accesses 1 _B trap unaligned halfword and word accesses. If this bit is set to 1, an unaligned access generates a UsageFault. Unaligned LDM, STM, LDRD, and STRD instructions always fault irrespective of whether UNALIGN_TRP is set to 1.
DIV_0_TRP	4	rw	Divide by Zero Trap Enable Enables faulting or halting when the processor executes an SDIV or UDIV instruction with a divisor of 0: 0 _B do not trap divide by 0 1 _B trap divide by 0. When this bit is set to 0, a divide by zero returns a quotient of 0.

Central Processing Unit (CPU)

Field	Bits	Type	Description
BFHFNMI	8	rw	<p>Bus Fault Hard Fault and NMI Ignore Enables handlers with priority -1 or -2 to ignore data BusFaults caused by load and store instructions. This applies to the hard fault, NMI, and FAULTMASK escalated handlers:</p> <p>0_B data bus faults caused by load and store instructions cause a lock-up</p> <p>1_B handlers running at priority -1 and -2 ignore data bus faults caused by load and store instructions.</p> <p>Set this bit to 1 only when the handler and its data are in absolutely safe memory. The normal use of this bit is to probe system devices and bridges to detect control path problems and fix them.</p>
STKALIGN	9	rw	<p>Stack Alignment Indicates stack alignment on exception entry:</p> <p>0_B 4-byte aligned</p> <p>1_B 8-byte aligned.</p> <p>On exception entry, the processor uses bit[9] of the stacked PSR to indicate the stack alignment. On return from the exception it uses this stacked bit to restore the correct stack alignment.</p>
0	[31:10], [7:5], 2	r	<p>Reserved Read as 0; should be written with 0.</p>

System Handler Priority Registers

The SHPR1-SHPR3 registers set the priority level, 0 to 63 of the exception handlers that have configurable priority.

SHPR1-SHPR3 are byte accessible.

The system fault handlers and the priority field and register for each handler are:

Table 2-21 System fault handler priority fields

Handler	Field	Register description
MemManage	PRI_4	System Handler Priority Register 1 on Page 2-71
BusFault	PRI_5	
UsageFault	PRI_6	
SVCall	PRI_11	System Handler Priority Register 2 on Page 2-71

Central Processing Unit (CPU)

Table 2-21 System fault handler priority fields (cont'd)

Handler	Field	Register description
PendSV	PRI_14	System Handler Priority Register 3 on Page 2-72
SysTick	PRI_15	

Each PRI_N field is 8 bits wide, but the XMC4300 implements only bits[7:2] of each field, and bits[1:0] read as zero and ignore writes.

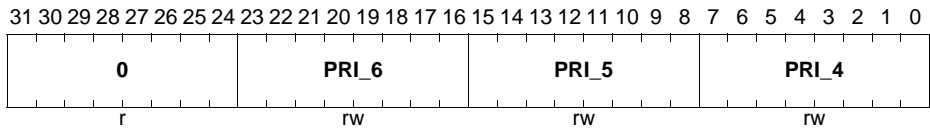
System Handler Priority Register 1

SHPR1

System Handler Priority Register 1

(E000 ED18_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PRI_4	[7:0]	rw	Priority of system handler 4, MemManage
PRI_5	[15:8]	rw	Priority of system handler 5, BusFault
PRI_6	[23:16]	rw	Priority of system handler 6, UsageFault
0	[31:24]	r	Reserved Read as 0; should be written with 0.

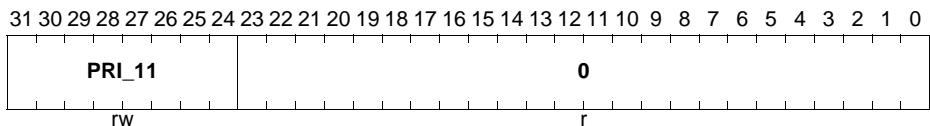
System Handler Priority Register 2

SHPR2

System Handler Priority Register 2

(E000 ED1C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PRI_11	[31:24]	rw	Priority of system handler 11, SVCcall
0	[23:0]	r	Reserved Read as 0; should be written with 0.

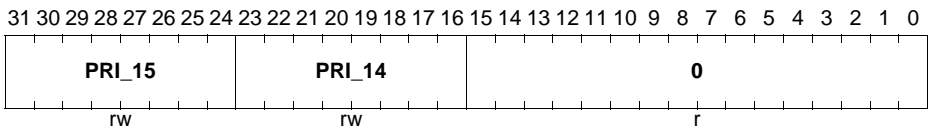
System Handler Priority Register 3

SHPR3

System Handler Priority Register 3

(E000 ED20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PRI_14	[23:16]	rw	Priority of system handler 14 PendSV
PRI_15	[31:24]	rw	Priority of system handler 15 SysTick exception
0	[15:0]	r	Reserved Read as 0; should be written with 0.

System Handler Control and State Register

The SHCSR enables the system handlers, and indicates:

- the pending status of the BusFault, MemManage fault, and SVC exceptions
- the active status of the system handlers.

Central Processing Unit (CPU)

SHCSR

System Handler Control and State Register

(E000 ED24_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0													USG FAU LTE NA	BUS FAU LTE NA	MEM FAU LTE NA
r													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SVC ALL PEN DED	BUS FAU LTP END ED	MEM FAU LTP END ED	USG FAU LTP END ED	SYS TICK ACT	PEN DSV ACT	0	MON ITOR ACT	SVC ALL ACT	0			USG FAU LTA CT	0	BUS FAU LTA CT	MEM FAU LTA CT
rw	rw	rw	rw	rw	rw	r	rw	rw	r			rw	r	rw	rw

Field	Bits	Type	Description
MEMFAULTACT	0	rw	MemManage exception active bit Reads as 1 if exception is active.
BUSFAULTACT	1	rw	BusFault exception active bit Reads as 1 if exception is active.
USGFAULTACT	3	rw	UsageFault exception active bit Reads as 1 if exception is active.
SVCALLACT	7	rw	SVCall active bit Reads as 1 if SVC call is active.
MONITORACT	8	rw	Debug monitor active bit Reads as 1 if Debug monitor is active.
PENDSVACT	10	rw	PendSV exception active bit Reads as 1 if exception is active.
SYSTICKACT	11	rw	SysTick exception active bit Reads as 1 if exception is active.
USGFAULTPENDED	12	rw	UsageFault exception pending bit Reads as 1 if exception is pending.
MEMFAULTPENDED	13	rw	MemManage exception pending bit Reads as 1 if exception is pending.

Central Processing Unit (CPU)

Field	Bits	Type	Description
BUSFAULTPENDED	14	rw	BusFault exception pending bit Reads as 1 if exception is pending.
SVCALLPENDED	15	rw	SVCall pending bit Reads as 1 if exception is pending.
MEMFAULTENA	16	rw	MemManage enable bit Set to 1 to enable.
BUSFAULTENA	17	rw	BusFault enable bit Set to 1 to enable.
USGFAULTENA	18	rw	UsageFault enable bit Set to 1 to enable.
0	[31:19], 9, [6:4], 2	r	Reserved Read as 0; should be written with 0.

Notes

1. Active bits, read as 1 if the exception is active, or as 0 if it is not active. You can write to these bits to change the active status of the exceptions, but see the Caution in this section.
2. Pending bits, read as 1 if the exception is pending, or as 0 if it is not pending. You can write to these bits to change the pending status of the exceptions.
3. Enable bits, set to 1 to enable the exception, or set to 0 to disable the exception.

If you disable a system handler and the corresponding fault occurs, the processor treats the fault as a hard fault.

You can write to this register to change the pending or active status of system exceptions. An OS kernel can write to the active bits to perform a context switch that changes the current exception type.

Note: Software that changes the value of an active bit in this register without correct adjustment to the stacked content can cause the processor to generate a fault exception. Ensure software that writes to this register retains and subsequently restores the current active status.

Note: After you have enabled the system handlers, if you have to change the value of a bit in this register you must use a read-modify-write procedure to ensure that you change only the required bit.

Configurable Fault Status Register

The CFSR indicates the cause of a MemManage fault, BusFault, or UsageFault.

The flags in the MMFSR indicate the cause of memory access faults.

Central Processing Unit (CPU)

The flags in the BFSR indicate the cause of a bus access fault.

The UFSR indicates the cause of a UsageFault.

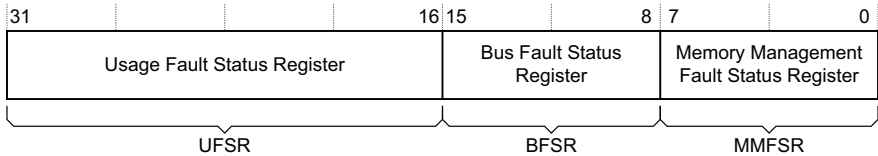


Figure 2-9 CFSR

The CFSR is byte accessible. You can access the CFSR or its subregisters as follows:

- access the complete CFSR with a word access to 0xE000ED28
- access the MMFSR with a byte access to 0xE000ED28
- access the MMFSR with a byte access to 0xE000ED28
- access the MMFSR and BFSR with a halfword access to 0xE000ED28
- access the BFSR with a byte access to 0xE000ED29
- access the UFSR with a halfword access to 0xE000ED2A

Note: The UFSR bits are sticky. This means as one or more fault occurs, the associated bits are set to 1. A bit that is set to 1 is cleared to 0 only by writing 1 to that bit, or by a reset.

CFSR

Configurable Fault Status Register

(E000 ED28_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						DIV YZE RO	UNA LIGN ED	0				NOC P	INVP C	INVS TAT E	UND EFIN STR
r						rw	rw	r				rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BFA RVA LID	0	LSP ERR	STK ERR	UNS TKE RR	IMP RECI SER R	PRE CISE RR	IBUS ERR	MMA RVA LID	0	MLS PER R	MST KER R	MUN STK ERR	0	DAC CVIO L	IACC VIOL
rw	r	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	r	rw	rw

Central Processing Unit (CPU)

Field	Bits	Type	Description
IACCVIOL	0	rw	<p>Instruction access violation flag</p> <p>0_B no instruction access violation fault 1_B the processor attempted an instruction fetch from a location that does not permit execution.</p> <p>This fault occurs on any access to an XN region, even when the MPU is disabled or not present. When this bit is 1, the PC value stacked for the exception return points to the faulting instruction. The processor has not written a fault address to the MMAR.</p>
DACCVIOL	1	rw	<p>Data access violation flag</p> <p>0_B no data access violation fault 1_B the processor attempted a load or store at a location that does not permit the operation.</p> <p>When this bit is 1, the PC value stacked for the exception return points to the faulting instruction. The processor has loaded the MMAR with the address of the attempted access.</p>
MUNSTKERR	3	rw	<p>MemManage fault on unstacking for a return from exception</p> <p>0_B no unstacking fault 1_B unstack for an exception return has caused one or more access violations.</p> <p>This fault is chained to the handler. This means that when this bit is 1, the original return stack is still present. The processor has not adjusted the SP from the failing return, and has not performed a new save. The processor has not written a fault address to the MMAR.</p>
MSTKERR	4	rw	<p>MemManage fault on stacking for exception entry</p> <p>0_B no stacking fault 1_B stacking for an exception entry has caused one or more access violations.</p> <p>When this bit is 1, the SP is still adjusted but the values in the context area on the stack might be incorrect. The processor has not written a fault address to the MMAR.</p>

Central Processing Unit (CPU)

Field	Bits	Type	Description
MLSPERR	5	rw	<p>MemManage fault during floating point lazy state preservation</p> <p>0_B No MemManage fault occurred during floating-point lazy state preservation</p> <p>1_B A MemManage fault occurred during floating-point lazy state preservation</p>
MMARVALID	7	rw	<p>MemManage Fault Address Register (MMFAR) valid flag</p> <p>0_B value in MMAR is not a valid fault address</p> <p>1_B MMAR holds a valid fault address.</p> <p>If a MemManage fault occurs and is escalated to a HardFault because of priority, the HardFault handler must set this bit to 0. This prevents problems on return to a stacked active MemManage fault handler whose MMAR value has been overwritten.</p>
IBUSERR	8	rw	<p>Instruction bus error</p> <p>0_B no instruction bus error</p> <p>1_B instruction bus error.</p> <p>The processor detects the instruction bus error on prefetching an instruction, but it sets the IBUSERR flag to 1 only if it attempts to issue the faulting instruction.</p> <p>When the processor sets this bit is 1, it does not write a fault address to the BFAR.</p>
PRECISERR	9	rw	<p>Precise data bus error</p> <p>0_B no precise data bus error</p> <p>1_B a data bus error has occurred, and the PC value stacked for the exception return points to the instruction that caused the fault.</p> <p>When the processor sets this bit is 1, it writes the faulting address to the BFAR.</p>

Central Processing Unit (CPU)

Field	Bits	Type	Description
IMPRECISERR	10	rw	<p>Imprecise data bus error</p> <p>0_B no imprecise data bus error 1_B a data bus error has occurred, but the return address in the stack frame is not related to the instruction that caused the error.</p> <p>When the processor sets this bit to 1, it does not write a fault address to the BFAR.</p> <p>This is an asynchronous fault. Therefore, if it is detected when the priority of the current process is higher than the BusFault priority, the BusFault becomes pending and becomes active only when the processor returns from all higher priority processes. If a precise fault occurs before the processor enters the handler for the imprecise BusFault, the handler detects both IMPRECISERR set to 1 and one of the precise fault status bits set to 1.</p>
UNSTKERR	11	rw	<p>BusFault on unstacking for a return from exception</p> <p>0_B no unstacking fault 1_B stacking for an exception entry has caused one or more BusFaults.</p> <p>This fault is chained to the handler. This means that when the processor sets this bit to 1, the original return stack is still present. The processor does not adjust the SP from the failing return, does not performed a new save, and does not write a fault address to the BFAR.</p>
STKERR	12	rw	<p>BusFault on stacking for exception entry</p> <p>0_B no stacking fault 1_B stacking for an exception entry has caused one or more BusFaults.</p> <p>When the processor sets this bit to 1, the SP is still adjusted but the values in the context area on the stack might be incorrect. The processor does not write a fault address to the BFAR.</p>

Central Processing Unit (CPU)

Field	Bits	Type	Description
LSPERR	13	rw	<p>BusFault during floating point lazy state preservation</p> <p>0_B No bus fault occurred during floating-point lazy state preservation.</p> <p>1_B A bus fault occurred during floating-point lazy state preservation</p>
BFARVALID	15	rw	<p>BusFault Address Register (BFAR) valid flag</p> <p>0_B value in BFAR is not a valid fault address</p> <p>1_B BFAR holds a valid fault address.</p> <p>The processor sets this bit to 1 after a BusFault where the address is known. Other faults can set this bit to 0, such as a MemManage fault occurring later. If a BusFault occurs and is escalated to a hard fault because of priority, the hard fault handler must set this bit to 0. This prevents problems if returning to a stacked active BusFault handler whose BFAR value has been overwritten.</p>
UNDEFINSTR	16	rw	<p>Undefined instruction UsageFault</p> <p>0_B no undefined instruction UsageFault</p> <p>1_B the processor has attempted to execute an undefined instruction.</p> <p>When this bit is set to 1, the PC value stacked for the exception return points to the undefined instruction. An undefined instruction is an instruction that the processor cannot decode.</p>
INVSTATE	17	rw	<p>Invalid state UsageFault</p> <p>0_B no invalid state UsageFault</p> <p>1_B the processor has attempted to execute an instruction that makes illegal use of the EPSR.</p> <p>When this bit is set to 1, the PC value stacked for the exception return points to the instruction that attempted the illegal use of the EPSR.</p> <p>This bit is not set to 1 if an undefined instruction uses the EPSR.</p>

Central Processing Unit (CPU)

Field	Bits	Type	Description
INVPC	18	rw	<p>Invalid PC load UsageFault caused by an invalid PC load by EXC_RETURN:</p> <p>0_B no invalid PC load UsageFault 1_B the processor has attempted an illegal load of EXC_RETURN to the PC, as a result of an invalid context, or an invalid EXC_RETURN value.</p> <p>When this bit is set to 1, the PC value stacked for the exception return points to the instruction that tried to perform the illegal load of the PC.</p>
NOCP	19	rw	<p>No coprocessor UsageFault 0_B no UsageFault caused by attempting to access a coprocessor 1_B the processor has attempted to access a coprocessor.</p>
UNALIGNED	24	rw	<p>Unaligned access UsageFault 0_B no unaligned access fault, or unaligned access trapping not enabled 1_B the processor has made an unaligned memory access.</p> <p>Enable trapping of unaligned accesses by setting the UNALIGN_TRP bit in the CCR to 1, see Configuration and Control Register on Page 2-68. Unaligned LDM, STM, LDRD, and STRD instructions always fault irrespective of the setting of UNALIGN_TRP.</p>
DIVBYZERO	25	rw	<p>Divide by zero UsageFault 0_B no divide by zero fault, or divide by zero trapping not enabled 1_B the processor has executed an SDIV or UDIV instruction with a divisor of 0</p> <p>When the processor sets this bit to 1, the PC value stacked for the exception return points to the instruction that performed the divide by zero. Enable trapping of divide by zero by setting the DIV_0_TRP bit in the CCR to 1, see Configuration and Control Register on Page 2-68.</p>
0	[31:26], [23:20], 14, 6, 2	r	<p>Reserved Read as 0; should be written with 0.</p>

HardFault Status Register

The HFSR gives information about events that activate the HardFault handler.

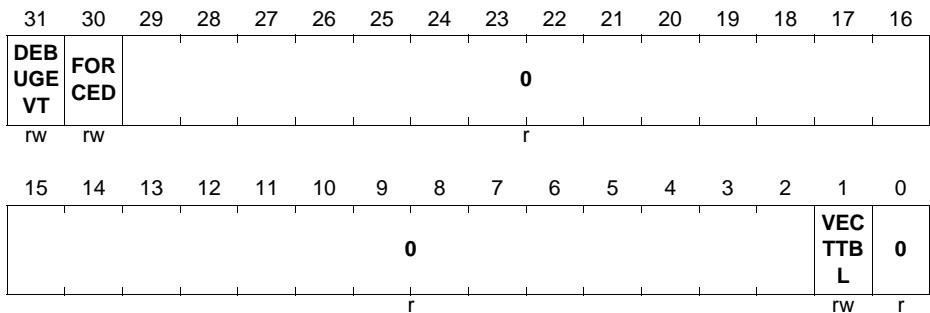
This register is read, write to clear. This means that bits in the register read normally, but writing 1 to any bit clears that bit to 0. The bit assignments are:

HFSR

HardFault Status Register

(E000 ED2C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
VECTBL	1	rw	<p>BusFault on vector table read</p> <p>Indicates a BusFault on a vector table read during exception processing:</p> <p>0_B no BusFault on vector table read 1_B BusFault on vector table read</p> <p>This error is always handled by the hard fault handler.</p> <p>When this bit is set to 1, the PC value stacked for the exception return points to the instruction that was preempted by the exception.</p>
FORCED	30	rw	<p>Forced HardFault</p> <p>Indicates a forced hard fault, generated by escalation of a fault with configurable priority that cannot be handles, either because of priority or because it is disabled:</p> <p>0_B no forced HardFault 1_B forced HardFault.</p> <p>When this bit is set to 1, the HardFault handler must read the other fault status registers to find the cause of the fault.</p>

Central Processing Unit (CPU)

Field	Bits	Type	Description
DEBUGEVT	31	rw	Reserved for Debug use When writing to the register you must write 0 to this bit, otherwise behavior is Unpredictable
0	[29:2], 0	r	Reserved Read as 0; should be written with 0.

Note: The HFSR bits are sticky. This means as one or more fault occurs, the associated bits are set to 1. A bit that is set to 1 is cleared to 0 only by writing 1 to that bit, or by a reset.

MemManage Fault Address Register

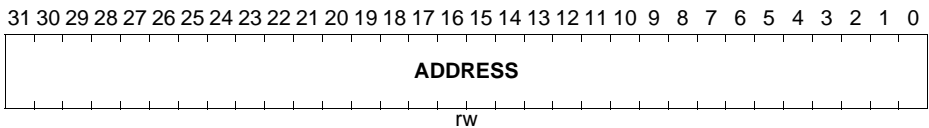
The MMFAR contains the address of the location that generated a MemManage fault.

MMFAR

MemManage Fault Address Register

(E000 ED34_H)

Reset Value: XXXX XXXX_H



Field	Bits	Type	Description
ADDRESS	[31:0]	rw	Address causing the fault When the MMARVALID bit of the MMFSR is set to 1, this field holds the address of the location that generated the MemManage fault

When an unaligned access faults, the address is the actual address that faulted. Because a single read or write instruction can be split into multiple aligned accesses, the fault address can be any address in the range of the requested access size.

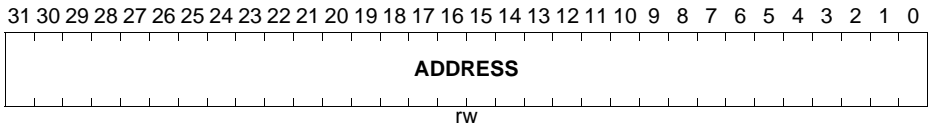
Flags in the MMFSR indicate the cause of the fault, and whether the value in the MMFAR is valid. See MemManage Fault Status Register on [Page 2-74](#).

BusFault Address Register

The BFAR contains the address of the location that generated a BusFault.

BFAR

BusFault Address Register (E000 ED38_H) Reset Value: XXXX XXXX_H



Field	Bits	Type	Description
ADDRESS	[31:0]	rw	Address causing the fault When the BFARVALID bit of the BFSR is set to 1, this field holds the address of the location that generated the BusFault

When an unaligned access faults the address in the BFAR is the one requested by the instruction, even if it is not the address of the fault.

Flags in the BFSR indicate the cause of the fault, and whether the value in the BFAR is valid. See BusFault Status Register on [Page 2-74](#).

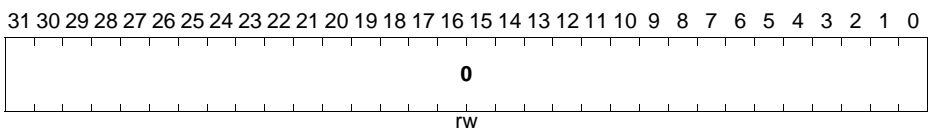
Auxiliary Fault Status Register

The AFSR contains additional system fault information.

This register is read, write to clear. This means that bits in the register read normally, but writing 1 to any bit clears that bit to 0.

AFSR

Auxiliary Fault Status Register (E000 ED3C_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
VALUE	[31:0]	rw	Reserved Read as 0; should be written with 0.

Central Processing Unit (CPU)

Each AFSR bit maps directly to an AUXFAULT input of the processor, and a single-cycle HIGH signal on the input sets the corresponding AFSR bit to one. It remains set to 1 until you write 1 to the bit to clear it to zero.

When an AFSR bit is latched as one, an exception does not occur. Use an interrupt if an exception is required.

2.9.2 SysTick Registers

SysTick Control and Status Register

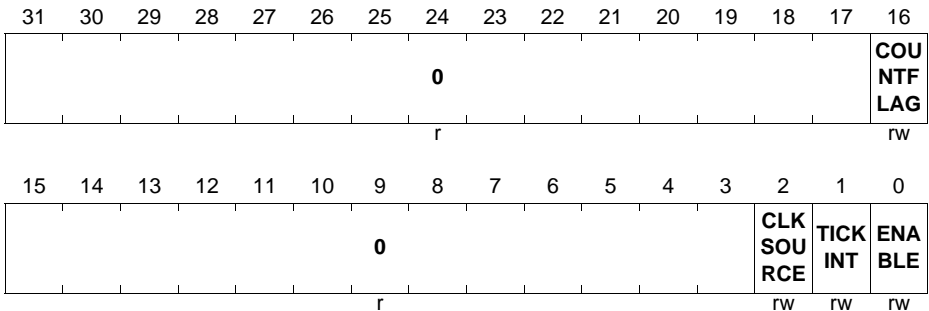
The SysTick SYST_CSR register enables the SysTick features.

SYST_CSR

SysTick Control and Status Register

(E000 E010_H)

Reset Value: 0000 0004_H



Field	Bits	Type	Description
ENABLE	0	rw	Enable Enables the counter: 0 _B counter disabled 1 _B counter enabled.
TICKINT	1	rw	Tick Interrupt Enable Enables SysTick exception request: 0 _B counting down to zero does not assert the SysTick exception request 1 _B counting down to zero to asserts the SysTick exception request. Software can use COUNTFLAG to determine if SysTick has ever counted to zero.

Central Processing Unit (CPU)

Field	Bits	Type	Description
CLKSOURCE	2	rw	Clock source $0_B \quad f_{STDBY} / 2$ $1_B \quad f_{CPU}$
COUNTFLAG	16	rw	Counter Flag Returns 1 if timer counted to 0 since last time this was read.
0	[31:17], [15:3]	r	Reserved Read as 0; should be written with 0.

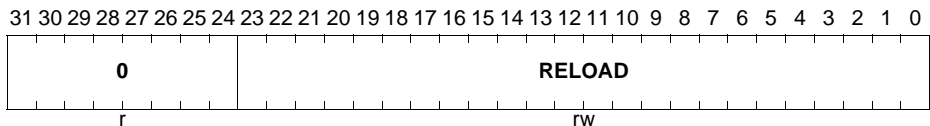
When ENABLE is set to 1, the counter loads the RELOAD value from the SYST_RVR register and then counts down. On reaching 0, it sets the COUNTFLAG to 1 and optionally asserts the SysTick depending on the value of TICKINT. It then loads the RELOAD value again, and begins counting.

SysTick Reload Value Register

The SYST_RVR register specifies the start value to load into the SYST_CVR register.

SYST_RVR

SysTick Reload Value Register (E000 E014_H) **Reset Value: XXXX XXXX_H**



Field	Bits	Type	Description
RELOAD	[23:0]	rw	Reload Value Value to load into the SYST_CVR register when the counter is enabled and when it reaches 0, see Calculating the RELOAD value.
0	[31:24]	r	Reserved Read as 0; should be written with 0.

Notes on calculating the RELOAD value

1. The RELOAD value can be any value in the range 0x00000001-0x00FFFFFF. A start value of 0 is possible, but has no effect because the SysTick exception request and COUNTFLAG are activated when counting from 1 to 0.

Central Processing Unit (CPU)

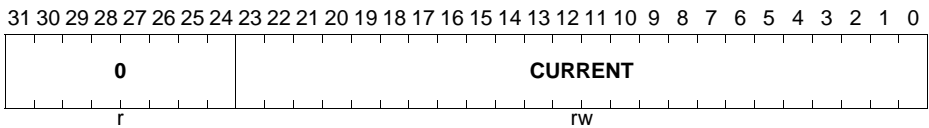
2. The RELOAD value is calculated according to its use. For example, to generate a multi-shot timer with a period of N processor clock cycles, use a RELOAD value of N-1. If the SysTick interrupt is required every 100 clock pulses, set RELOAD to 99.

SysTick Current Value Register

The SYST_CVR register contains the current value of the SysTick counter.

SYST_CVR

SysTick Current Value Register (E000 E018_H) **Reset Value: XXXX XXXX_H**



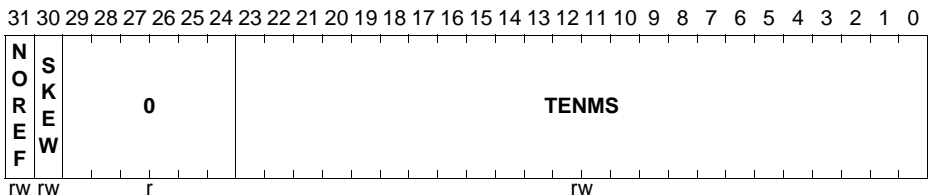
Field	Bits	Type	Description
CURRENT	[23:0]	rw	Current Value Reads return the current value of the SysTick counter. A write of any value clears the field to 0, and also clears the SYST_CSR COUNTFLAG bit to 0.
0	[31:24]	r	Reserved Read as 0; should be written with 0.

SysTick Calibration Value Register

The SYST_CALIB register indicates the SysTick calibration properties.

SYST_CALIB

SysTick Calibration Value Register
r **(E000 E01C_H)** **Reset Value: C000 0000_H**



Field	Bits	Type	Description
TENMS	[23:0]	rw	Ten Milliseconds Reload Value Reload value for 10ms (100Hz) timing, subject to system clock skew errors. If the value reads as zero, the calibration value is not known.
SKEW	30	rw	Ten Milliseconds Skewed Indicates whether the TENMS value is exact: 0 _B TENMS value is exact 1 _B TENMS value is inexact, or not given. An inexact TENMS value can affect the suitability of SysTick as a software real time clock.
NOREF	31	rw	No Reference Clock Indicates whether the device provides a reference clock to the processor: 0 _B reference clock provided 1 _B no reference clock provided. If your device does not provide a reference clock, the SYST_CSR.CLKSOURCE bit reads-as-one and ignores writes.
0	[29:24]	r	Reserved Read as 0; should be written with 0.

2.9.3 NVIC Registers

Interrupt Set-enable Registers

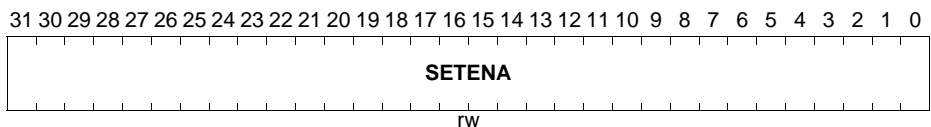
The NVIC_ISERx (x=0-3) registers enable interrupts, and show which interrupts are enabled.

NVIC_ISERx (x=0-3)

Interrupt Set-enable Register x

(E000 E100_H + 4*x)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SETENA	[31:0]	rw	Interrupt set-enable bits 0 _B Write: no effect Read: interrupt disabled 1 _B Write: enable interrupt Read: interrupt enabled

If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority. If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.

Interrupt Clear-enable Registers

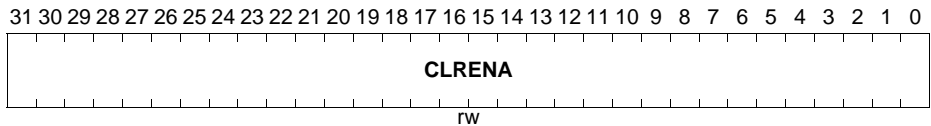
The NVIC_ICERx registers disable interrupts, and show which interrupts are enabled.

NVIC_ICERx (x=0-3)

Interrupt Clear-enable Register x

(E000 E180_H + 4*x)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CLRENA	[31:0]	rw	Interrupt clear-enable bits. 0 _B Write: no effect Read: interrupt disabled 1 _B Write: disable interrupt Read: interrupt enabled

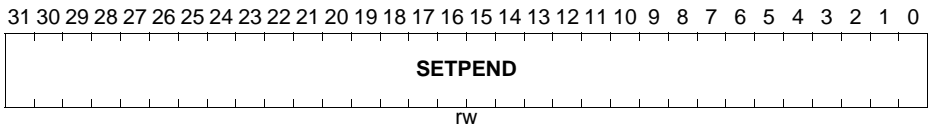
Interrupt Set-pending Registers

The NVIC_ISPRx registers force interrupts into the pending state, and show which interrupts are pending.

NVIC_ISPRx (x=0-3)

Interrupt Set-pending Register x

(E000 E200_H + 4*x) Reset Value: 0000 0000_H



Field	Bits	Type	Description
SETPEND	[31:0]	rw	Interrupt set-pending bits. 0 _B Write: no effect Read: interrupt is not pending 1 _B Write: changes interrupt state to pending Read: interrupt is pending

Writing 1 to the ISPR bit corresponding to:

- an interrupt that is pending has no effect
- a disabled interrupt sets the state of that interrupt to pending

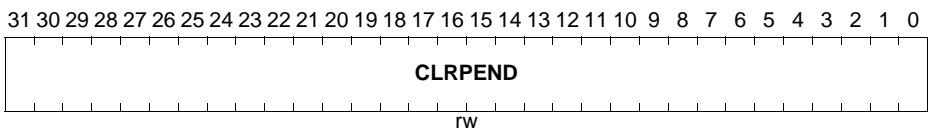
Interrupt Clear-pending Registers

The NVIC_ICPRx registers remove the pending state from interrupts, and show which interrupts are pending.

NVIC_ICPRx (x=0-3)

Interrupt Clear-pending Register x

(E000 E280_H + 4*x) Reset Value: 0000 0000_H



Field	Bits	Type	Description
CLRPEND	[31:0]	rw	Interrupt set-pending bits. 0 _B Write: no effect Read: interrupt is not pending 1 _B Write: removes pending state an interrupt Read: interrupt is pending

Note: Writing 1 to an ICPR bit does not affect the active state of the corresponding interrupt.

Interrupt Active Bit Registers

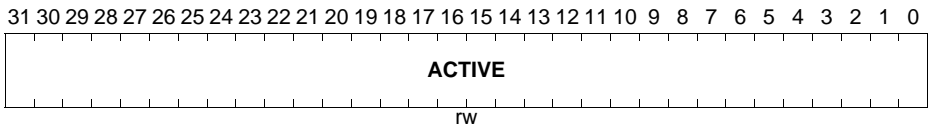
The NVIC_IABRx registers indicate which interrupts are active.

NVIC_IABRx (x=0-3)

Interrupt Active Bit Register x

(E000 E300_H + 4*x)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ACTIVE	[31:0]	rw	Interrupt active flags: 0 _B interrupt not active 1 _B interrupt active

A bit reads as one if the status of the corresponding interrupt is active or active and pending.

Interrupt Priority Registers

The NVIC_IPRx (x=0-27) registers provide an 8-bit priority field for each interrupt. These registers are byte-accessible. Each register holds four priority fields as shown:

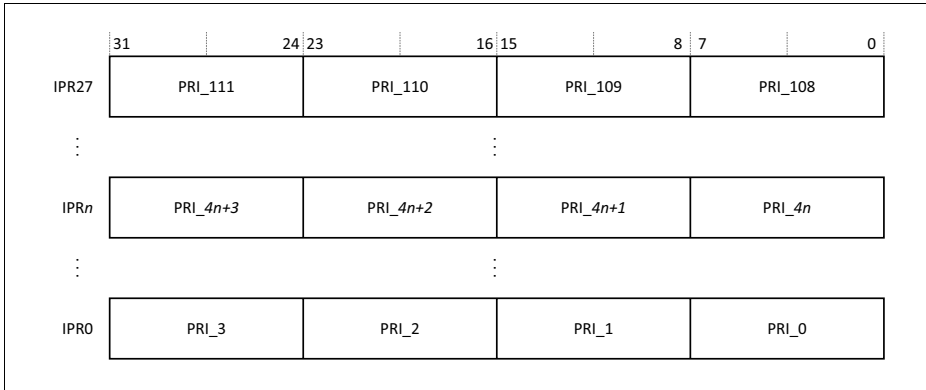


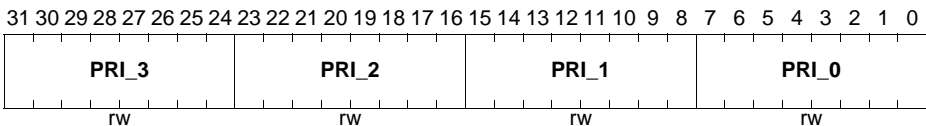
Figure 2-10 Interrupt Priority Register

NVIC_IPRx (x=0-27)

Interrupt Priority Register x

(E000 E400_H + 4*x)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PRI_0	[7:0]	rw	Priority value 0
PRI_1	[15:8]	rw	Priority value 1
PRI_2	[23:16]	rw	Priority value 2
PRI_3	[31:24]	rw	Priority value 3 The lower the value, the greater the priority of the corresponding interrupt. The processor implements only bits[7:n] of each field, bits[n-1:0] read as zero and ignore writes.

See “Using CMSIS functions to access NVIC” on [Page 2-45](#) for more information about the access to the interrupt priority array, which provides the software view of the interrupt priorities.

Find the IPR number and byte offset for interrupt m as follows:

Central Processing Unit (CPU)

- the corresponding IPR number n , see [Figure 2-10](#) on [Page 2-917](#), is given by $n = m \text{ DIV } 4$
- the byte offset of the required Priority field in this register is $m \text{ MOD } 4$, where:
 - byte offset 0 refers to register bits[7:0]
 - byte offset 1 refers to register bits[15:8]
 - byte offset 2 refers to register bits[23:16]
 - byte offset 3 refers to register bits[31:24].

Software Trigger Interrupt Register

Write to the STIR to generate an interrupt from software.

When the USERSETMPEND bit in the SCR is set to 1, unprivileged software can access the STIR, see System Control Register on [Page 2-67](#).

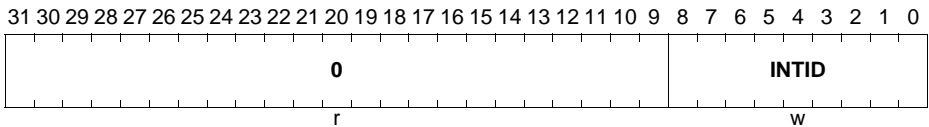
Note: Only privileged software can enable unprivileged access to the STIR.

STIR

Software Trigger Interrupt Register

(E000 EF00_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
INTID	[8:0]	w	Interrupt ID of the interrupt to trigger in the range 0-111. For example, a value of 0x03 specifies interrupt IRQ3.
0	[31:9]	r	Reserved Read as 0; should be written with 0.

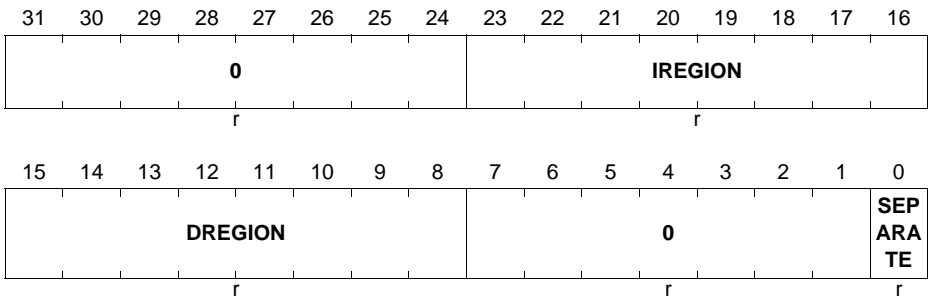
2.9.4 MPU Registers

MPU Type Register

The MPU_TYPE register indicates whether the MPU is present, and if so, how many regions it supports.

MPU_TYPE

MPU Type Register (E000 ED90_H) Reset Value: 0000 0800_H



Field	Bits	Type	Description
SEPARATE	0	r	Support for unified or separate instruction and data memory maps 0 _B unified
DREGION	[15:8]	r	Number of supported MPU data regions 08 _H Eight MPU regions
IREGION	[23:16]	r	Number of supported MPU instruction regions Always contains 0x00. The MPU memory map is unified and is described by the DREGION field.
0	[31:24], [7:1]	r	Reserved Read as 0; should be written with 0.

MPU Control Register

The MPU_CTRL register:

- enables the MPU
- enables the default memory map background region
- enables use of the MPU when in the hard fault, Non-maskable Interrupt (NMI), and FAULTMASK escalated handlers.

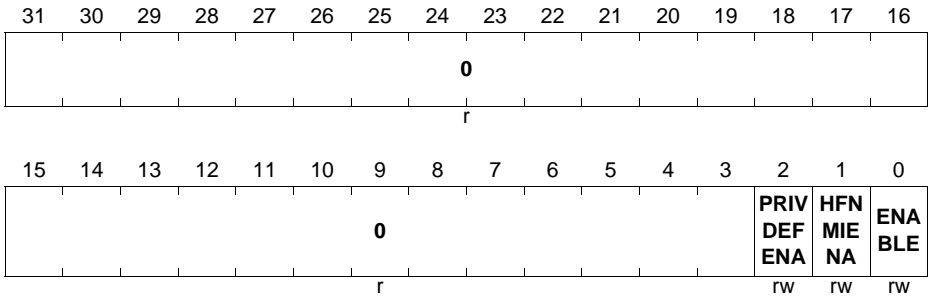
Central Processing Unit (CPU)

MPU_CTRL

MPU Control Register

(E000 ED94_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ENABLE	0	rw	Enable MPU 0 _B MPU disabled 1 _B MPU enabled.
HFNMIENA	1	rw	Enable the operation of MPU during hard fault, NMI, and FAULTMASK handlers When the MPU is enabled: 0 _B MPU is disabled during hard fault, NMI, and FAULTMASK handlers, regardless of the value of the ENABLE bit 1 _B the MPU is enabled during hard fault, NMI, and FAULTMASK handlers. When the MPU is disabled, if this bit is set to 1 the behavior is Unpredictable.

Central Processing Unit (CPU)

Field	Bits	Type	Description
PRIVDEFENA	2	rw	<p>Enables privileged software access to the default memory map</p> <p>0_B If the MPU is enabled, disables use of the default memory map. Any memory access to a location not covered by any enabled region causes a fault.</p> <p>1_B If the MPU is enabled, enables use of the default memory map as a background region for privileged software accesses.</p> <p>When enabled, the background region acts as if it is region number -1. Any region that is defined and enabled has priority over this default map. f the MPU is disabled, the processor ignores this bit.</p>
0	[31:3]	r	<p>Reserved Read as 0; should be written with 0.</p>

When ENABLE and PRIVDEFENA are both set to 1:

- For privileged accesses, the default memory map is as described in Memory model on [Page 2-20](#). Any access by privileged software that does not address an enabled memory region behaves as defined by the default memory map.
- Any access by unprivileged software that does not address an enabled memory region causes a MemManage fault.

XN and Strongly-ordered rules always apply to the System Control Space regardless of the value of the ENABLE bit.

When the ENABLE bit is set to 1, at least one region of the memory map must be enabled for the system to function unless the PRIVDEFENA bit is set to 1. If the PRIVDEFENA bit is set to 1 and no regions are enabled, then only privileged software can operate.

When the ENABLE bit is set to 0, the system uses the default memory map. This has the same memory attributes as if the MPU is not implemented, see [Table 2-6](#) on [Page 2-22](#). The default memory map applies to accesses from both privileged and unprivileged software.

When the MPU is enabled, accesses to the System Control Space and vector table are always permitted. Other areas are accessible based on regions and whether PRIVDEFENA is set to 1.

Unless HFNMIENA is set to 1, the MPU is not enabled when the processor is executing the handler for an exception with priority –1 or –2. These priorities are only possible when handling a hard fault or NMI exception, or when FAULTMASK is enabled. Setting the HFNMIENA bit to 1 enables the MPU when operating with these two priorities.

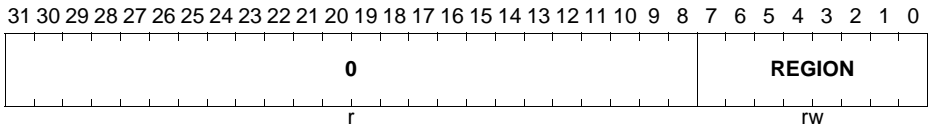
Central Processing Unit (CPU)

MPU Region Number Register

The MPU_RNR selects which memory region is referenced by the MPU_RBAR and MPU_RASR registers.

MPU_RNR

MPU Region Number Register (E000 ED98_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
REGION	[7:0]	rw	Region Indicates the MPU region referenced by the MPU_RBAR and MPU_RASR registers. The MPU supports 8 memory regions, so the permitted values of this field are 0-7.
0	[31:8]	r	Reserved Read as 0; should be written with 0.

Normally, you write the required region number to this register before accessing the MPU_RBAR or MPU_RASR. However you can change the region number by writing to the MPU_RBAR with the VALID bit set to 1, see MPU Region Base Address Register. This write updates the value of the REGION field.

MPU Region Base Address Register

The MPU_RBAR defines the base address of the MPU region selected by the MPU_RNR, and can update the value of the MPU_RNR.

Central Processing Unit (CPU)

MPU_RBAR

MPU Region Base Address Register

(E000 ED9C_H)

Reset Value: 0000 0000_H

MPU_RBAR_A1

MPU Region Base Address Register A1

(E000 EDA4_H)

Reset Value: 0000 0000_H

MPU_RBAR_A2

MPU Region Base Address Register A2

(E000 EDAC_H)

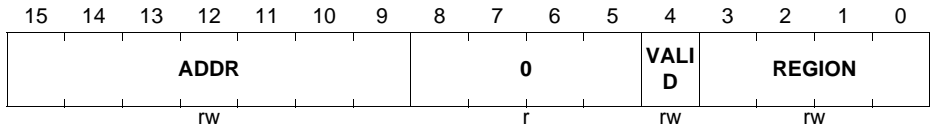
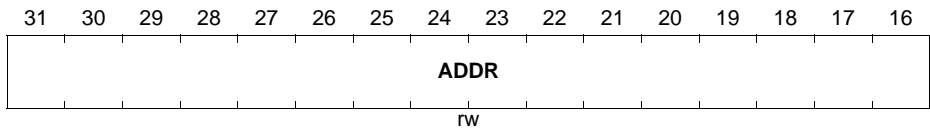
Reset Value: 0000 0000_H

MPU_RBAR_A3

MPU Region Base Address Register A3

(E000 EDB4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
REGION	[3:0]	rw	MPU region field For the behavior on writes, see the description of the VALID field. On reads, returns the current region number, as specified by the RNR.

Central Processing Unit (CPU)

Field	Bits	Type	Description
VALID	4	rw	<p>MPU Region Number valid bit</p> <p>Write:</p> <p>0_B MPU_RNR not changed, and the processor:</p> <ul style="list-style-type: none"> - updates the base address for the region specified in the MPU_RNR - ignores the value of the REGION field <p>1_B the processor:</p> <ul style="list-style-type: none"> - updates the value of the MPU_RNR to the value of the REGION field - updates the base address for the region specified in the REGION field. <p>Always reads as zero.</p>
ADDR	[31:9]	rw	<p>Region base address field</p> <p>The value of N (N = 9 for bit definition) depends on the region size. For more information see The ADDR field.</p>
0	[8:5]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

The ADDR field

The ADDR field is bits[31:N] of the MPU_RBAR. The region size, as specified by the SIZE field in the MPU_RASR, defines the value of N:

$$N = \text{Log}_2(\text{Region size in bytes}),$$

If the region size is configured to 4GB, in the MPU_RASR, there is no valid ADDR field. In this case, the region occupies the complete memory map, and the base address is 0x00000000.

The base address is aligned to the size of the region. For example, a 64KB region must be aligned on a multiple of 64KB, for example, at 0x00010000 or 0x00020000.

MPU Region Attribute and Size Register

The MPU_RASR defines the region size and memory attributes of the MPU region specified by the MPU_RNR, and enables that region and any subregions.

MPU_RASR is accessible using word or halfword accesses:

- the most significant halfword holds the region attributes
- the least significant halfword holds the region size and the region and subregion enable bits.

Central Processing Unit (CPU)

MPU_RASR

MPU Region Attribute and Size Register

(E000 EDA0_H)

Reset Value: 0000 0000_H

MPU_RASR_A1

MPU Region Attribute and Size Register A1

(E000 EDA8_H)

Reset Value: 0000 0000_H

MPU_RASR_A2

MPU Region Attribute and Size Register A2

(E000 EDB0_H)

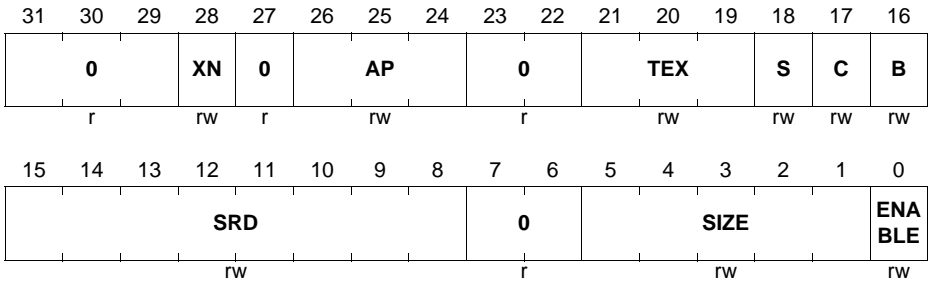
Reset Value: 0000 0000_H

MPU_RASR_A3

MPU Region Attribute and Size Register A3

(E000 EDB8_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ENABLE	0	rw	Region enable bit.
SIZE	[5:1]	rw	MPU protection region size The minimum permitted value is 3 (0b00010), see See SIZE field values for more information.
SRD	[15:8]	rw	Subregion disable bits For each bit in this field: 0 _B corresponding sub-region is enabled 1 _B corresponding sub-region is disabled See Subregions on Page 2-52 for more information. Region sizes of 128 bytes and less do not support subregions. When writing the attributes for such a region, write the SRD field as 0x00.
B	16	rw	Memory access attribute see Table 2-15 on Page 2-48 .

Central Processing Unit (CPU)

Field	Bits	Type	Description
C	17	rw	Memory access attribute see Table 2-15 on Page 2-48 .
S	18	rw	Shareable bit see Table 2-15 on Page 2-48 .
TEX	[21:19]	rw	Memory access attribute see Table 2-15 on Page 2-48 .
AP	[26:24]	rw	Access permission field see Table 2-18 on Page 2-49 .
XN	28	rw	Instruction access disable bit 0 _B instruction fetches enabled 1 _B instruction fetches disabled.
0	[31:29, 27, [23:22], [7:6]	r	Reserved Read as 0; should be written with 0.

For information about access permission, see [MPU Access Permission Attributes](#) on [Page 2-48](#).

SIZE field values

The SIZE field defines the size of the MPU memory region specified by the RNR. as follows:

$$(\text{Region size in bytes}) = 2(\text{SIZE}+1)$$

The smallest permitted region size is 32B, corresponding to a SIZE value of 4. [Table 2-22](#) gives example SIZE values, with the corresponding region size and value of N in the MPU_RBAR.

Table 2-22 Example SIZE field values

SIZE value	Region size	Value of N ¹⁾	Note
0b00100 (4)	32B	5	Minimum permitted size
0b01001 (9)	1KB	10	-
0b10011 (19)	1MB	20	-
0b11101 (29)	1GB	30	-
0b11111 (31)	4GB	32	Maximum possible size

1) In the MPU_RBAR, see MPU Region Base Address Register on [Page 2-96](#).

2.9.5 FPU Registers

Coprocessor Access Control Register

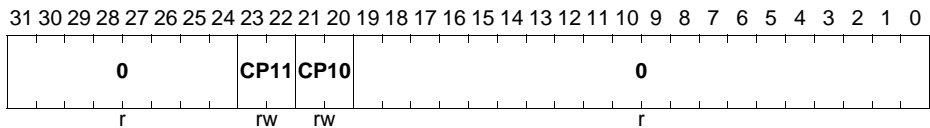
The CPACR register specifies the access privileges for coprocessors.

CPACR

Coprocessor Access Control Register

(E000 ED88_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CP10	[21:20]	rw	Access privileges for coprocessor 10 The possible values of each field are: 00 _B Access denied. Any attempted access generates a NOCP UsageFault. 01 _B Privileged access only. An unprivileged access generates a NOCP fault. 10 _B Reserved. The result of any access is Unpredictable. 11 _B Full access.
CP11	[23:22]	rw	Access privileges for coprocessor 11 The possible values of each field are: 00 _B Access denied. Any attempted access generates a NOCP UsageFault. 01 _B Privileged access only. An unprivileged access generates a NOCP fault. 10 _B Reserved. The result of any access is Unpredictable. 11 _B Full access.
0	[31:24], [19:0]	r	Reserved Read as 0; should be written with 0.

Floating-point Context Control Register

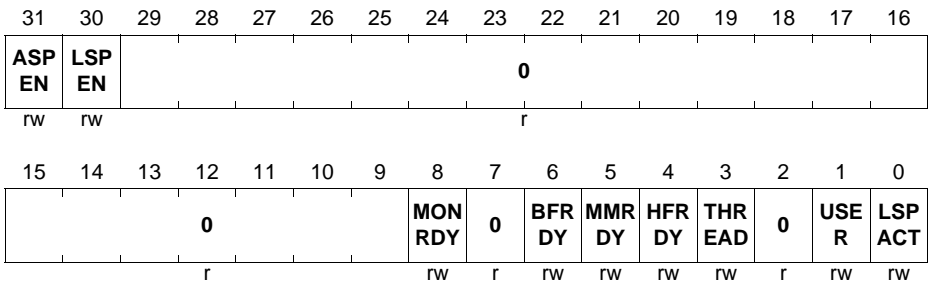
The FPCCR register sets or returns FPU control data.

FPCCR

Floating-point Context Control Register

(E000 EF34_H)

Reset Value: C000 0000_H



Field	Bits	Type	Description
LSPACT	0	rw	<p>Lazy State Preservation Active</p> <p>0_B Lazy state preservation is not active.</p> <p>1_B Lazy state preservation is active. floating-point stack frame has been allocated but saving state to it has been deferred.</p>
USER	1	rw	<p>User allocated Stack Frame</p> <p>0_B Privilege level was not user when the floating-point stack frame was allocated.</p> <p>1_B Privilege level was user when the floating-point stack frame was allocated.</p>
THREAD	3	rw	<p>Thread Mode allocated Stack Frame</p> <p>0_B Mode was not Thread Mode when the floating-point stack frame was allocated.</p> <p>1_B Mode was Thread Mode when the floating-point stack frame was allocated.</p>
HFRDY	4	rw	<p>HardFault Ready</p> <p>0_B Priority did not permit setting the HardFault handler to the pending state when the floating-point stack frame was allocated.</p> <p>1_B Priority permitted setting the HardFault handler to the pending state when the floating-point stack frame was allocated.</p>

Central Processing Unit (CPU)

Field	Bits	Type	Description
MMRDY	5	rw	<p>MemManage Ready</p> <p>0_B MemManage is disabled or priority did not permit setting the MemManage handler to the pending state when the floating-point stack frame was allocated.</p> <p>1_B MemManage is enabled and priority permitted setting the MemManage handler to the pending state when the floating-point stack frame was allocated.</p>
BFRDY	6	rw	<p>BusFault Ready</p> <p>0_B BusFault is disabled or priority did not permit setting the BusFault handler to the pending state when the floating-point stack frame was allocated.</p> <p>1_B BusFault is enabled and priority permitted setting the BusFault handler to the pending state when the floating-point stack frame was allocated.</p>
MONRDY	8	rw	<p>Monitor Ready</p> <p>0_B Debug Monitor is disabled or priority did not permit setting MON_PEND when the floating-point stack frame was allocated.</p> <p>1_B Debug Monitor is enabled and priority permits setting MON_PEND when the floating-point stack frame was allocated.</p>
LSPEN	30	rw	<p>Lazy State Preservation Enabled</p> <p>0_B Disable automatic lazy state preservation for floating-point context.</p> <p>1_B Enable automatic lazy state preservation for floating-point context.</p>
ASPEN	31	rw	<p>Automatic State Preservation</p> <p>Enables CONTROL setting on execution of a floating-point instruction. This results in automatic hardware state preservation and restoration, for floating-point context, on exception entry and exit.</p> <p>0_B Disable CONTROL setting on execution of a floating-point instruction.</p> <p>1_B Enable CONTROL setting on execution of a floating-point instruction.</p>

Central Processing Unit (CPU)

Field	Bits	Type	Description
0	[29:9], 7, 2	r	Reserved Read as 0; should be written with 0.

Floating-point Context Address Register

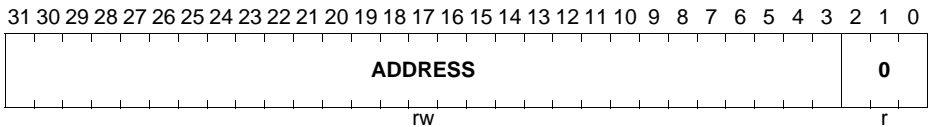
The FPCAR register holds the location of the unpopulated floating-point register space allocated on an exception stack frame.

FPCAR

Floating-point Context Address Register

(E000 EF38_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ADDRESS	[31:3]	rw	Address The location of the unpopulated floating-point register space allocated on an exception stack frame.
0	[2:0]	r	Reserved Read as 0; should be written with 0.

Floating-point Status Control Register

The FPSCR register provides all necessary User level control of the floating-point system.

FPSCR

Floating-point Status Control Register

Reset Value: XXXX XXXX_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
N	Z	C	V	0	AHP	DN	FZ	RMode			0				
rw	rw	rw	rw	r	rw	rw	rw	rw			r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								IDC	0	IXC	UFC	OFC	DZC	IOC	
r								rw	r	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
IOC	0	rw	Invalid Operation cumulative exception bit IOC set to 1 indicates that the Invalid Operation cumulative exception has occurred since 0 was last written to IOC.
DZC	1	rw	Division by Zero cumulative exception bit DZC set to 1 indicates that the Division by Zero cumulative exception has occurred since 0 was last written to DZC.
OFC	2	rw	Overflow cumulative exception bit OFC set to 1 indicates that the Overflow cumulative exception has occurred since 0 was last written to OFC.
UFC	3	rw	Underflow cumulative exception bit UFC set to 1 indicates that the Underflow cumulative exception has occurred since 0 was last written to UFC.
IXC	4	rw	Inexact cumulative exception bit IXC set to 1 indicates that the Inexact cumulative exception has occurred since 0 was last written to IXC.
IDC	7	rw	Input Denormal cumulative exception bit see bits [4:0].

Central Processing Unit (CPU)

Field	Bits	Type	Description
RMode	[23:22]	rw	Rounding Mode control field 00 _B Round to Nearest (RN) mode 01 _B Round towards Plus Infinity (RP) mode 10 _B Round towards Minus Infinity (RM) mode 11 _B Round towards Zero (RZ) mode. The specified rounding mode is used by almost all floating-point instructions.
FZ	24	rw	Flush-to-zero mode control bit 0 _B Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. 1 _B Flush-to-zero mode enabled.
DN	25	rw	Default NaN mode control bit 0 _B NaN operands propagate through to the output of a floating-point operation. 1 _B Any operation involving one or more NaNs returns the Default NaN.
AHP	26	rw	Alternative half-precision control bit 0 _B IEEE half-precision format selected. 1 _B Alternative half-precision format selected.
V	28	rw	Overflow condition code flag Floating-point comparison operations update this flag.
C	29	rw	Carry condition code flag Floating-point comparison operations update this flag.
Z	30	rw	Zero condition code flag Floating-point comparison operations update this flag.
N	31	rw	Negative condition code flag Floating-point comparison operations update this flag.
0	27, [21:8], [6:5]	r	Reserved Read as 0; should be written with 0.

Central Processing Unit (CPU)

Floating-point Default Status Control Register

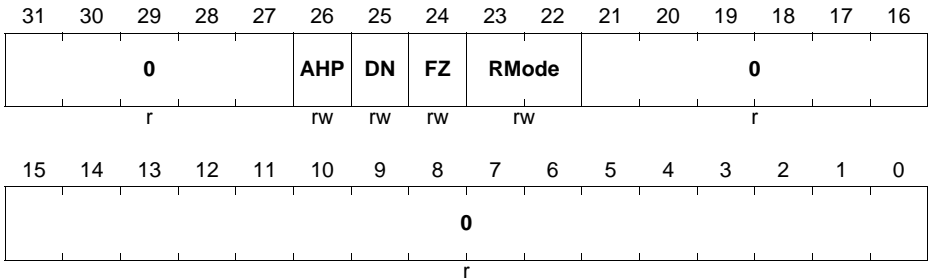
The FPDSCR register holds the default values for the floating-point status control data.

FPDSCR

Floating-point Default Status Control Register

(E000 EF3C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RMode	[23:22]	rw	Default value for FPSCR.RMode
FZ	24	rw	Default value for FPSCR.FZ
DN	25	rw	Default value for FPSCR.DN
AHP	26	rw	Default value for FPSCR.AHP
0	[31:27], [21:0]	r	Reserved Read as 0; should be written with 0.

Media and FP Feature Register 0

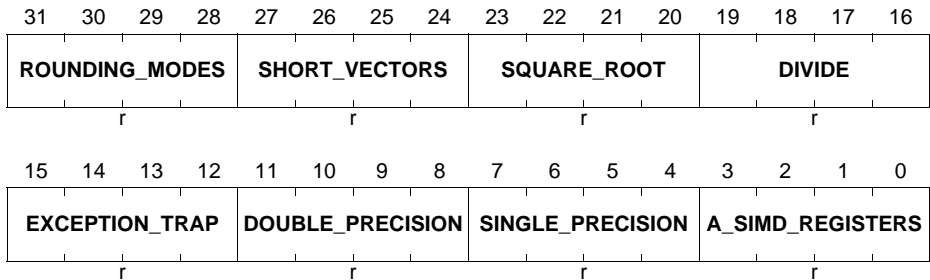
Describes the features provided by the Floating-point extension. Must be interpreted with [MVFR1](#).

MVFR0

Media and FP Feature Register 0

(E000 EF40_H)

Reset Value: 1011 0021_H



Field	Bits	Type	Description
ROUNDING_MODES	[31:28]	r	Indicates the rounding modes supported by the FP floating-point hardware. 1 _H : All rounding modes supported.
SHORT_VECTORS	[27:24]	r	Indicates the hardware support for FP short vectors. 0 _H : Not supported.
SQUARE_ROOT	[23:20]	r	Indicates the hardware support for FP square root operations. 1 _H : Supported.
DIVIDE	[19:16]	r	Indicates the hardware support for FP divide operations. 1 _H : Supported.
EXCEPTION_TRAP	[15:12]	r	Indicates whether the FP hardware implementation supports exception trapping. 0 _H : Not supported.
DOUBLE_PRECISION	[11:8]	r	Indicates the hardware support for FP double-precision operations. 0 _H : Not supported.

Central Processing Unit (CPU)

Field	Bits	Type	Description
SINGLE_PRECISION	[7:4]	r	Indicates the hardware support for FP single-precision operations. 2 _H : Supported (Instruction to load a single-precision floating-point constant, and conversions between single-precision and fixed-point values is available).
A_SIMD_REGISTERS	[3:0]	r	Indicates the size of the FP register bank. 1 _H : Supported, 16 x 64-bit registers.

Media and FP Feature Register 1

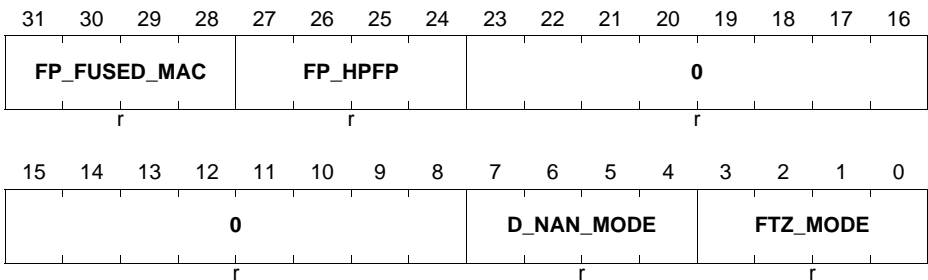
Describes the features provided by the Floating-point extension. Must be interpreted with **MVFR0**.

MVFR1

Media and FP Feature Register 1

(E000 EF44_H)

Reset Value: 1100 0011_H



Field	Bits	Type	Description
FP_FUSED_MAC	[31:28]	r	Indicates whether the FP supports fused multiply accumulate operations. 1 _H : Supported.
FP_HPFP	[27:24]	r	Indicates whether the FP supports half-precision floating-point conversion operations. 1 _H : Supported.

Central Processing Unit (CPU)

Field	Bits	Type	Description
D_NAN_MODE	[7:4]	r	Indicates whether the FP hardware implementation supports only the Default NaN mode. 1 _H : Hardware supports propagation of NaN values.
FTZ_MODE	[3:0]	r	Indicates whether the FP hardware implementation supports only the Flush-to-Zero mode of operation. 1 _H : Hardware supports full denormalized number arithmetic.
0	[23:8]	r	Reserved Read as 0; should be written with 0.

3 Bus System

The XMC4300 is targeted for use in embedded systems. Therefore the key features are timing determinism and low latency on real time events. Bus bandwidth is required particularly for communication peripherals.

The bus system will therefore provide:

- Timing Determinism
- Low Latency
- Performance
- Throughput

3.1 Bus Interfaces

This chapter describes the features for the two kinds of interfaces.

- **Memory Interface**
- **Peripheral Interface**

All on-chip peripherals and memories are attached to the Bus Matrix, in some cases via peripheral bridges. All on-chip modules implement Little Endian data organization. The following types of transfer are supported:

- Locked Transfers
- Burst Operation
- Protection Control

Pipelining is also supported for bandwidth critical transfers.

Memory Interface

The on-chip memories capable to accept a transfer request with each bus clock cycle.

The memory interface data bus width is 32-bit. Each memory slave support 32-bit, 16-bit and 8-bit access types.

Peripheral Interface

Each slave supports 32-bit accesses. Some slaves also support 8-bit and/or 16-bit accesses.

3.2 Bus Matrix

The central part of the bus system is built up around a multilayer AHB-lite compliant matrix. By means of this technique the bus masters and bus slaves can be connected in a flexible way while maintaining high bus performance.

The Bus Matrix depicted in [Figure 3-1](#) implements an optimized topology enabling zero wait state data accesses between the Masters and Slaves connected to it. Dedicated

arbitration scheme enables optimal access conflicts resolution resulting in improved system stability and real time behavior.

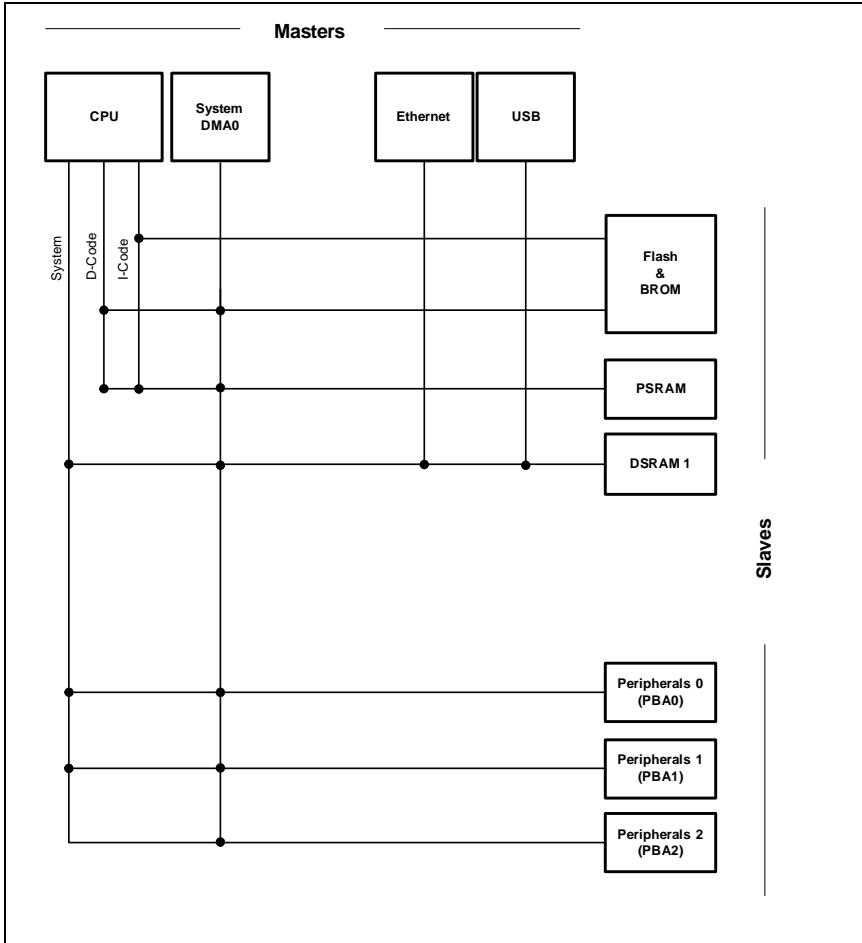


Figure 3-1 Multilayer Bus Matrix

Arbitration Priorities

In case of concurring access to the same slave the master with the highest priority is granted the bus.

Table 3-1 Access Priorities per Slave¹⁾

	CPU	GPDMA0	ETH	USB
PMU/FLASH	1	2	-	-
PSRAM	1	2	-	-
DSRAM1	1	2	3	4
PBA0	1	2	-	-
PBA1	1	2	-	-
PBA2	1	2	-	-

1) Lower number means higher priority

The DSRAM priorities are chosen to support the application dependence of the data memories:

- DSRAM1: general purpose data storage

4 Service Request Processing

A hardware pulse or level change is called Service Request (SR) in an XMC4300 system. Service Requests are the fastest way to send trigger “messages” between connected on-chip resources.

An SR can generate any of the following requests

- Interrupt
- DMA
- Peripheral action

This chapter describes the available Service Requests and the different ways to select and process them.

Notes

1. *The CPU exception model and interrupt processing (by NVIC unit) are described in the CPU chapter.*
2. *General Purpose DMA request processing is described in the GPDMA chapter*

Table 4-1 Abbreviations

DLR	DMA Line Router
ERU	Event Request Unit
NVIC	Nested Vectored Interrupt Controller
SR	Service Request

4.1 Overview

Efficient Service Request Processing is based on the interconnect between the request sources and the request processing units. XMC4300 provides both fixed and programmable interconnect.

4.1.1 Features

The following features are provided for Service Request processing:

- Connectivity matrix between Service Requests and request processing units
 - Fixed connections
 - Programmable connections using ERU
- Event Request Unit (ERU)
 - Flexible processing of external and internal service requests
 - Programmable for edge and/or level triggering
 - Multiple inputs per channel
 - Triggers combinable from multiple inputs
 - Input and output gating

- DMA Line Router (DLR)
 - Routing and processing of DMA requests

4.1.2 Block Diagram

The shaded components shown in **Figure 4-1** are described in this chapter.

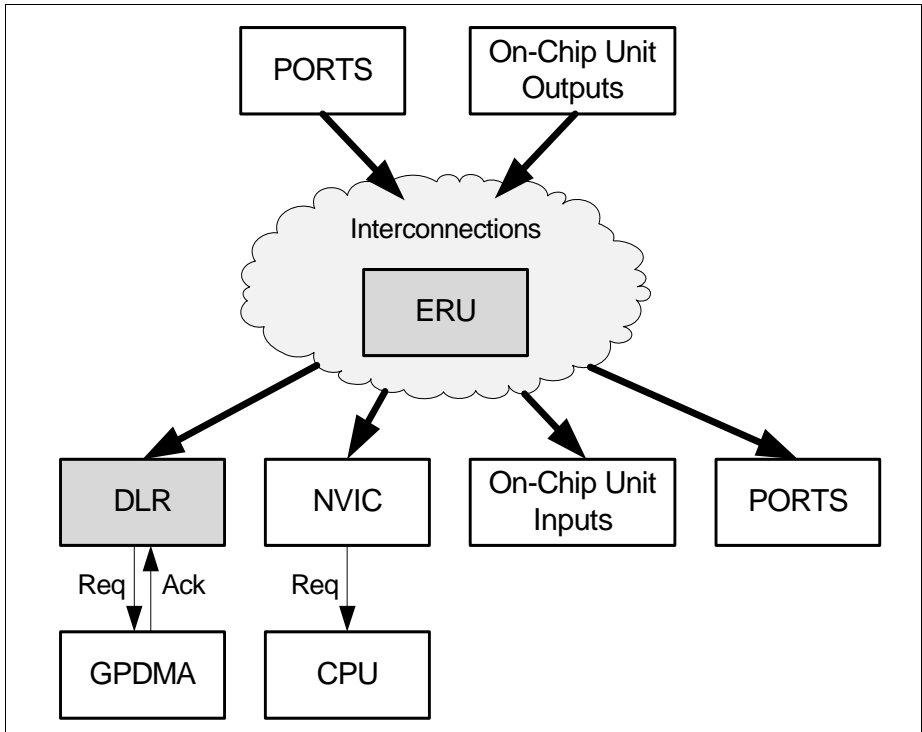


Figure 4-1 Block Diagram on Service Request Processing

4.2 Service Request Distribution

The following figure shown an example of how a service request can be distributed concurrently. To support the concurrent distribution to multiple receivers, the receiving modules are capable to enable/disable incoming requests.

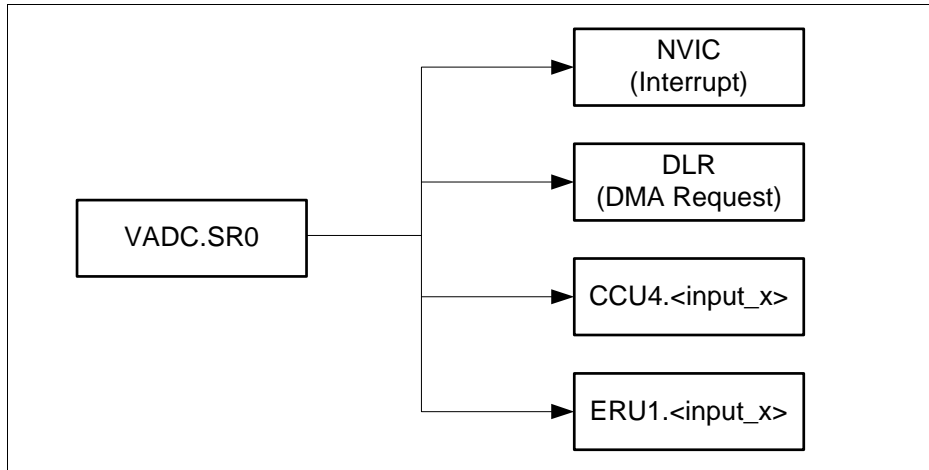


Figure 4-2 Example for Service Request Distribution

The units involved in Service Request distribution can be subdivided into

- Embedded real time services
- Interrupt and DMA services

Embedded real time services

Connectivity between On-Chip Units and PORTS is real time application and also chip package dependant. Related connectivity and availability of pins can be looked up in the

- “Interconnects” Section of the respective module(s) chapters
- “Parallel Ports” chapter and Data Sheet for PORTS
- Event Request Unit ([Section 4.5](#))

Interrupt and DMA services

The following table gives an overview on the number of service requests per module and how the service requests are assigned to NVIC Interrupt and DLR/GPDMA service providers.

Service Requests can be of type “Level” or “Pulse”. The DLR/GPDMA can only process “Pulse” type of requests while the NVIC can process both. The type of Service Requests generated is listed in column “Type” in [Table 4-2](#).

Table 4-2 Interrupt and DMA services per Module

Modules	Request Sources	NVIC	DLR/GPDMA	Type
VADC	20	20	20	Pulse
DAC	2	2	2	Pulse
CCU40-1	8	8	4	Pulse
CCU80	4	4	2	Pulse
CAN	8	8	4	Pulse
USIC0-1	12	12	4	Pulse
LEDTs0	1	1	-	Pulse
FCE	1	1	-	Pulse
PMU0/Flash	1	1	-	Pulse
GPDMA0	1	1	-	Level
SCU	1	1	-	Level
ERU0-1	8	8	4	Pulse
SDMMC	1	1	-	Level
USB0	1	1	-	Level
ETH0	1	1	-	Level
ECAT0	1	1	-	Level
Totals	71	71	40	-

4.3 Interrupt Service Requests

The NVIC is an integral part of the Cortex M4 processor unit. Due to a tight coupling with the CPU it allows to achieve lowest interrupt latency and efficient processing of late arriving interrupts.

NVIC Features

- 112 interrupt nodes
- Programmable priority level of 0-63 for each interrupt node. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority
- Request source can be level or edge signal type
- Dynamic reprioritization of interrupts.
- Grouping of priority values into group priority and subpriority fields.
- Interrupt tail-chaining.
- One external Non-maskable interrupt (NMI)

- Relocatable vector table
- Software interrupt generation

Level-sensitive and pulse interrupts

The NVIC is capable to capture both level-sensitive and pulse interrupts.

- A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Deassertion is typically triggered by the interrupt service routine (ISR). It is
 - used for less frequent requests and
 - the ISR is often more complex and longer.
- A pulse interrupt is asserted and after a fixed period of time automatically deasserted. The period of time depends on the peripheral, please refer to the “Service Request Generation” section of the respective peripheral. It is
 - used for more frequent requests and
 - the ISR is often more simple and shorter.

The way to process both types of requests differs and is described in section “Level-sensitive and pulse interrupts” in the CPU chapter.

Service Request to IRQ Number Assignment

Table 4-3 lists the service request sources per on-chip unit and their assignment to NVIC IRQ numbers. The resulting exception number is calculated by adding 16 to the IRQ Number. The first 16 exception numbers are used by the Cortex M4 CPU. For calculation of the resulting exception routine address please refer to the CPU chapter.

Table 4-3 Interrupt Node assignment

Service Request	IRQ Number	Description
SCU.SR0	0	System Control
ERU0.SR0 ERU0.SR3	1...4	External Request Unit 0
ERU1.SR0 ERU1.SR3	5...8	External Request Unit 1
NC	9, 10, 11	Reserved
PMU0.SR0	12	Program Management Unit
NC	13	Reserved
VADC.C0SR0 - VADC.C0SR3	14...17	Analog to Digital Converter Common Block 0
VADC.G0SR0 - VADC.G0SR3	18...21	Analog to Digital Converter Group 0

Service Request Processing
Table 4-3 Interrupt Node assignment (cont'd)

Service Request	IRQ Number	Description
VADC.G1SR0 - VADC.G1SR3	22...25	Analog to Digital Converter Group 1
VADC.G2SR0 - VADC.G2SR3	26...29	Analog to Digital Converter Group 2
VADC.G3SR0 - VADC.G3SR3	30...33	Analog to Digital Converter Group 3
NC	34...41	Reserved
DAC.SR0 - DAC.SR1	42, 43	Digital to Analog Converter
CCU40.SR0 - CCU40.SR3	44...47	Capture Compare Unit 4 (Module 0)
CCU41.SR0 - CCU41.SR3	48...51	Capture Compare Unit 4 (Module 1)
NC	52...59	Reserved
CCU80.SR0 - CCU80.SR3	60...63	Capture Compare Unit 8 (Module 0)
NC	64...67	Reserved
POSIF0.SR0 - POSIF0.SR1	68...69	Position Interface (Module 0)
NC	70...71	Reserved
NC	72...75	Reserved
CAN.SR0 - CAN.SR7	76...83	MultiCAN
USIC0.SR0 - USIC0.SR5	84...89	Universal Serial Interface Channel (Module 0)
USIC1.SR0 - USIC1.SR5	90...95	Universal Serial Interface Channel (Module 1)
NC	96...101	Reserved
LEDTS0.SR0	102	LED and Touch Sense Control Unit (Module 0)
NC	103	Reserved
FCE.SR0	104	Flexible CRC Engine
GPDMA0.SR0	105	General Purpose DMA unit 0
SDMMC.SR0	106	Multi Media Card Interface

Table 4-3 Interrupt Node assignment (cont'd)

Service Request	IRQ Number	Description
USB0.SR0	107	Universal Serial Bus
ETH0.SR0	108	Ethernet (Module 0)
ECAT0.SR0	109	EtherCAT (Module 0)
NC	110...111	Reserved

4.4 DMA Line Router (DLR)

The DMA line router provides the following functionality:

- Selection of DMA request sources
- Handling of the DMA request and acknowledge handshake
- Detection of service request overruns

4.4.1 Functional Description

This unit enables the user to select 8 DMA service requests out of the set of DMA capable service request sources. It handles the Request and Acknowledge handshake to the GPDMA unit. Furthermore it detects service request overruns.

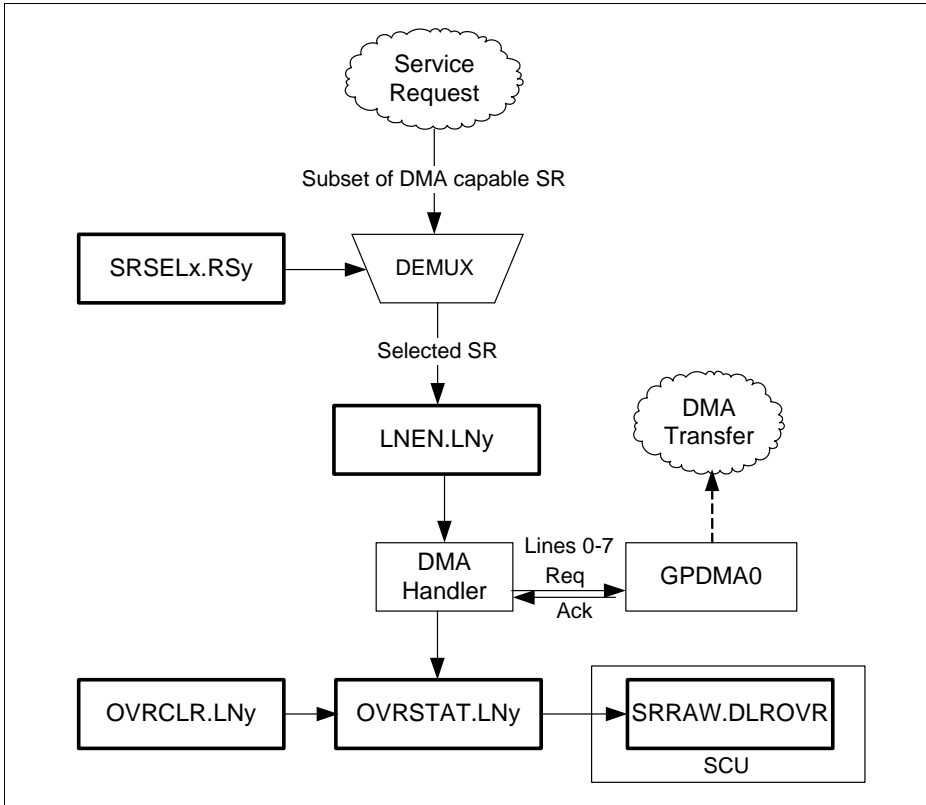


Figure 4-3 DMA Line Handler

For each DMA line the user can assign one service request source from the subset of DMA capable XMC4300 service request sources. The assignment is done by programming the SRSx bit field of register **DLR_SRSELO**. The DLR lines 0-7 are connected to GPDMA0.

If the selected service request pulse occurs and if the according line is enabled by the **DLR_LNEN** register, then the DMA handler forwards the request and stores it until the GPDMA responds with an acknowledge. A request pulse occurring while another transfer is ongoing is ignored and the according overrun status bit is set in the **DLR_OVRSTAT** register.

Once the overrun condition is entered the user can clear the overrun status bits by writing to the **DLR_OVRCLR** register. Additionally the pending request must be reset by successively disabling and enabling the respective line.

Service Request Processing

If any bit within the **DLR_OVRSTAT** register is set, a service request is flagged by setting the SCU_SRRAW.DLROVR bit.

The DLR unit has the following inputs:

Table 4-4 DMA Handler Service Request inputs

Service Request	# of Inputs	Description
ERU0.SR1 - ERU0.SR4	4	ERU0 (System Control) requests
VADC.C0SR0 - VADC.C0SR3	4	Analog to Digital Converter Common Block 0
VADC.G0SR0 - VADC.G0SR3	4	Analog to Digital Converter Group 0
VADC.G1SR0 - VADC.G1SR3	4	Analog to Digital Converter Group 1
DAC.SR0 - DAC.SR1	2	Digital to Analog Converter
CCU40.SR0 - CCU40.SR1	2	Capture Compare Unit 4 (Module 0)
CCU41.SR0 - CCU41.SR1	2	Capture Compare Unit 4 (Module 1)
CCU80.SR0 - CCU80.SR1	2	Capture Compare Unit 8 (Module 0)
CAN.SR0 - CAN.SR3	4	MultiCAN
USIC0.SR0 - USIC0.SR1	2	Universal Serial Interface Channel (Module 0)
USIC1.SR0 - USIC1.SR1	2	Universal Serial Interface Channel (Module 1)

4.4.2 DMA Service Request Source Selection

The selection of the request sources is done according to the following table by programming the **DLR_SRSELO** register. Please note that each service request source can be assigned to 2 different lines to provide maximum flexibility. For example VADC.SR0 can be assigned to line 0 and 4.

Table 4-5 DMA Request Source Selection

DMA Line	DMA Request Line	Selected by DLR_SRSEL bit field
0	ERU0.SR0	RS0 = 0000 _B
	VADC.C0SR0	RS0 = 0001 _B
	VADC.G0SR3	RS0 = 0010 _B
	VADC.G2SR0	RS0 = 0011 _B
	VADC.G2SR3	RS0 = 0100 _B
	Reserved	RS0 = 0101 _B
	CCU40.SR0	RS0 = 0110 _B
	CCU80.SR0	RS0 = 0111 _B
	Reserved	RS0 = 1000 _B
	CAN.SR0	RS0 = 1001 _B
	USIC0.SR0	RS0 = 1010 _B
	USIC1.SR0	RS0 = 1011 _B
	Reserved	RS0 = 1100 _B
	VADC.G3SR3	RS0 = 1101 _B
	Reserved	RS0 = 1110 _B
	Reserved	RS0 = 1111 _B
1	ERU0.SR3	RS1 = 0000 _B
	VADC.C0SR1	RS1 = 0001 _B
	VADC.G0SR2	RS1 = 0010 _B
	VADC.G1SR0	RS1 = 0011 _B
	VADC.G2SR2	RS1 = 0100 _B
	DAC.SR0	RS1 = 0101 _B
	CCU40.SR0	RS1 = 0110 _B
	CCU80.SR0	RS1 = 0111 _B
	Reserved	RS1 = 1000 _B
	CAN.SR0	RS1 = 1001 _B
	USIC0.SR0	RS1 = 1010 _B
	USIC1.SR0	RS1 = 1011 _B
	Reserved	RS1 = 1100 _B
VADC.G3SR0	RS1 = 1101 _B	

Service Request Processing

Table 4-5 DMA Request Source Selection (cont'd)

DMA Line	DMA Request Line	Selected by DLR_SRSEL bit field
	Reserved	RS1 = 1110 _B
	Reserved	RS1 = 1111 _B
2	ERU0.SR1	RS2 = 0000 _B
	VADC.C0SR2	RS2 = 0001 _B
	VADC.C0SR3	RS2 = 0010 _B
	VADC.G1SR3	RS2 = 0011 _B
	VADC.G2SR1	RS2 = 0100 _B
	Reserved	RS2 = 0101 _B
	Reserved	RS2 = 0110 _B
	CCU40.SR1	RS2 = 0111 _B
	CCU80.SR1	RS2 = 1000 _B
	Reserved	RS2 = 1001 _B
	CAN.SR1	RS2 = 1010 _B
	USIC0.SR1	RS2 = 1011 _B
	USIC1.SR1	RS2 = 1100 _B
	VADC.G3SR2	RS2 = 1101 _B
	Reserved	RS2 = 1110 _B
	Reserved	RS2 = 1111 _B
3	ERU0.SR2	RS3 = 0000 _B
	VADC.C0SR2	RS3 = 0001 _B
	VADC.C0SR3	RS3 = 0010 _B
	VADC.G1SR1	RS3 = 0011 _B
	VADC.G1SR2	RS3 = 0100 _B
	Reserved	RS3 = 0101 _B
	DAC.SR1	RS3 = 0110 _B
	CCU40.SR1	RS3 = 0111 _B
	CCU80.SR1	RS3 = 1000 _B
	Reserved	RS3 = 1001 _B
	CAN.SR1	RS3 = 1010 _B
	USIC0.SR1	RS3 = 1011 _B

Service Request Processing

Table 4-5 DMA Request Source Selection (cont'd)

DMA Line	DMA Request Line	Selected by DLR_SRSEL bit field
	USIC1.SR1	RS3 = 1100 _B
	VADC.G3SR1	RS3 = 1101 _B
	Reserved	RS3 = 1110 _B
	Reserved	RS3 = 1111 _B
4	ERU0.SR2	RS4 = 0000 _B
	VADC.G0SR0	RS4 = 0001 _B
	VADC.G0SR1	RS4 = 0010 _B
	VADC.G2SR1	RS4 = 0011 _B
	VADC.G2SR2	RS4 = 0100 _B
	Reserved	RS4 = 0101 _B
	DAC.SR1	RS4 = 0110 _B
	CCU41.SR0	RS4 = 0111 _B
	Reserved	RS4 = 1000 _B
	Reserved	RS4 = 1001 _B
	CAN.SR2	RS4 = 1010 _B
	USIC0.SR0	RS4 = 1011 _B
	USIC1.SR0	RS4 = 1100 _B
	VADC.G3SR1	RS4 = 1101 _B
	Reserved	RS4 = 1110 _B
	Reserved	RS4 = 1111 _B
5	ERU0.SR1	RS5 = 0000 _B
	VADC.G0SR0	RS5 = 0001 _B
	VADC.G0SR1	RS5 = 0010 _B
	VADC.G1SR2	RS5 = 0011 _B
	VADC.G2SR0	RS5 = 0100 _B
	DAC.SR0	RS5 = 0101 _B
	CCU41.SR0	RS5 = 0110 _B
	Reserved	RS5 = 0111 _B
	Reserved	RS5 = 1000 _B
	CAN.SR2	RS5 = 1001 _B

Service Request Processing

Table 4-5 DMA Request Source Selection (cont'd)

DMA Line	DMA Request Line	Selected by DLR_SRSEL bit field
	USIC0.SR0	RS5 = 1010 _B
	USIC1.SR0	RS5 = 1011 _B
	Reserved	RS5 = 1100 _B
	VADC.G3SR2	RS5 = 1101 _B
	Reserved	RS5 = 1110 _B
	Reserved	RS5 = 1111 _B
6	ERU0.SR3	RS6 = 0000 _B
	VADC.C0SR1	RS6 = 0001 _B
	VADC.G0SR2	RS6 = 0010 _B
	VADC.G1SR1	RS6 = 0011 _B
	VADC.G2SR3	RS6 = 0100 _B
	Reserved	RS6 = 0101 _B
	Reserved	RS6 = 0110 _B
	CCU41.SR1	RS6 = 0111 _B
	Reserved	RS6 = 1000 _B
	Reserved	RS6 = 1001 _B
	CAN.SR3	RS6 = 1010 _B
	USIC0.SR1	RS6 = 1011 _B
	USIC1.SR1	RS6 = 1100 _B
	VADC.G3SR0	RS6 = 1101 _B
	Reserved	RS6 = 1110 _B
	Reserved	RS6 = 1111 _B
7	ERU0.SR0	RS7 = 0000 _B
	VADC.C0SR0	RS7 = 0001 _B
	VADC.G0SR3	RS7 = 0010 _B
	VADC.G1SR0	RS7 = 0011 _B
	VADC.G1SR3	RS7 = 0100 _B
	Reserved	RS7 = 0101 _B
	CCU41.SR1	RS7 = 0110 _B
	Reserved	RS7 = 0111 _B

Table 4-5 DMA Request Source Selection (cont'd)

DMA Line	DMA Request Line	Selected by DLR_SRSEL bit field
	Reserved	RS7 = 1000 _B
	CAN.SR3	RS7 = 1001 _B
	USIC0.SR1	RS7 = 1010 _B
	USIC1.SR1	RS7 = 1011 _B
	Reserved	RS7 = 1100 _B
	VADC.G3SR3	RS7 = 1101 _B
	Reserved	RS7 = 1110 _B
	Reserved	RS7 = 1111 _B

4.5 Event Request Unit (ERU)

The Event Request Unit (ERU) is a versatile multiple input event detection and processing unit. The XMC4300 provides two units - ERU0 and ERU1.

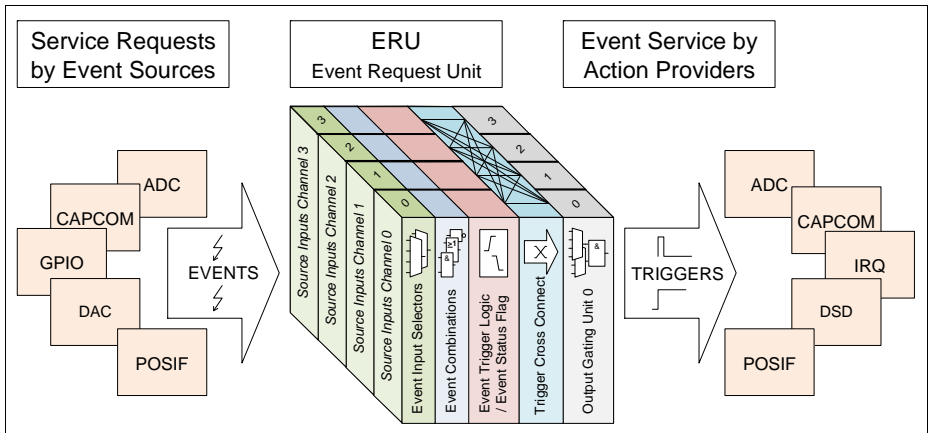


Figure 4-4 Event Request Unit Overview

Each ERU unit consists of the following blocks:

- An **Event Request Select (ERS)** unit.
 - Event Input Selectors allow the selection of one out of two inputs. For each of these two inputs, an vector of 4 possible signals is available.
 - Event Combinations allow a logical combination of two input signals to a common trigger.
- An **Event Trigger Logic (ETL)** per Input Channel allows the definition of the transition (edge selection, or by software) that lead to a trigger event and can also

Service Request Processing

store this status. Here, the input levels of the selected signals are translated into events.

- The Trigger **Cross Connect Matrix** distributes the events and status flags to the Output Channels. Additionally, trigger signals from other modules are made available and can be combined with the local triggers.
- An **Output Gating Unit (OGU)** combines the trigger events and status information and gates the Output depending on a gating signal.

Note: An event of one Input can lead to reactions on several Outputs, or also events on several Inputs can be combined to a reaction on one Output.

4.5.1 Event Request Select Unit (ERS)

For each Input Channel x (x = 0-3), an ERSx unit handles the input selection for the associated ETLx unit. Each ERSx performs a logical combination of two signals (Ax, Bx) to provide one combined output signal ERSxO to the associated ETLx. Input Ax can be selected from 4 options of the input vector ERU_xA[3:0] and can be optionally inverted. A similar structure exists for input Bx (selection from ERU_xB[3:0]).

In addition to the direct choice of either input Ax or Bx or their inverted values, the possible logical combinations for two selected inputs are a logical AND or a logical OR.

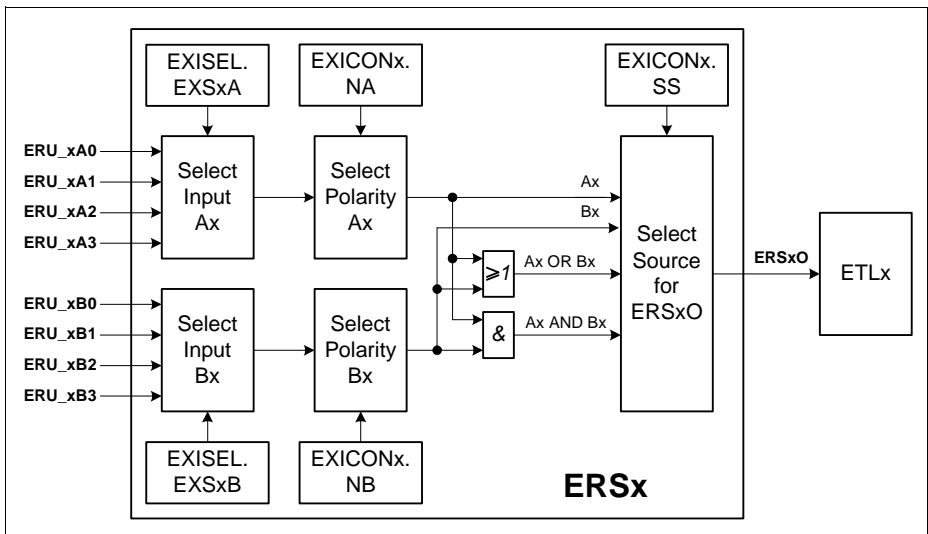


Figure 4-5 Event Request Select Unit Overview

The ERS units are controlled via register **ERU0_EXISEL** (one register for all four ERSx units) and registers EXICONx (one register for each ERSx and associated ETLx unit, e.g. **ERU0_EXICONx (x=0-3)** for Input Channel 0).

4.5.2 Event Trigger Logic (ETLx)

For each Input Channel x (x = 0-3), an event trigger logic ETLx derives a trigger event and related status information from the input ERSxO. Each ETLx is based on an edge detection block, where the detection of a rising or a falling edge can be individually enabled. Both edges lead to a trigger event if both enable bits are set (e.g. to handle a toggling input).

Each of the four ETLx units has an associated EXICONx register, that controls all options of an ETLx (the register also holds control bits for the associated ERSx unit, e.g. ERU0_EXICONx (x=0-3) to control ERS0 and ETL0).

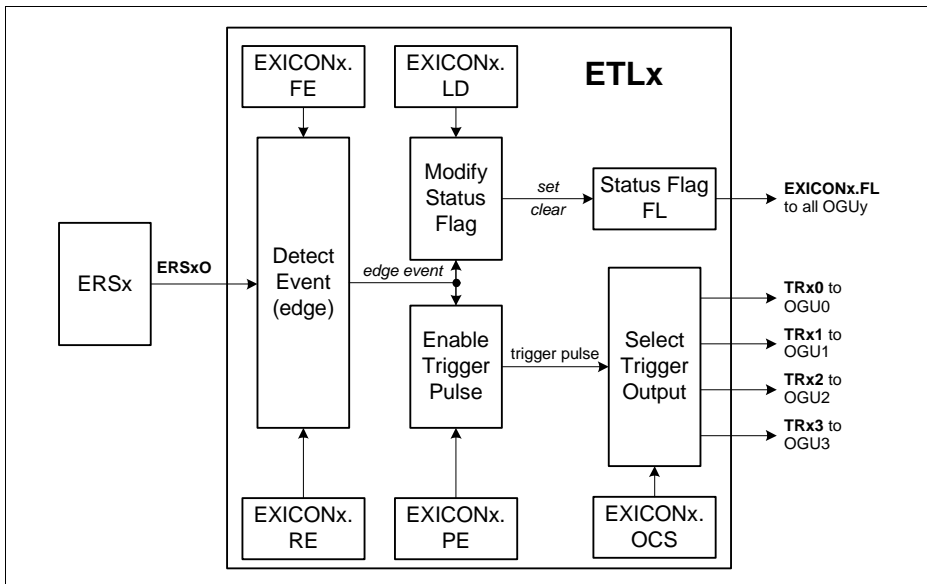


Figure 4-6 Event Trigger Logic Overview

When the selected event (edge) is detected, the status flag EXICONx.FL becomes set. This flag can also be modified by software. Two different operating modes are supported by this status flag.

It can be used as “sticky” flag, which is set by hardware when the desired event has been detected and has to be cleared by software. In this operating mode, it indicates that the event has taken place, but without indicating the actual status of the input.

In the second operating mode, it is cleared automatically if the “opposite” event is detected. For example, if only the falling edge detection is enabled to set the status flag, it is cleared when the rising edge is detected. In this mode, it can be used for pattern

Service Request Processing

detection where the actual status of the input is important (enabling both edge detections is not useful in this mode).

The output of the status flag is connected to all following Output Gating Units (OGUy) in parallel (see [Figure 4-7](#)) to provide **pattern detection capability of all OGUy** units based on different or the same status flags.

In addition to the modification of the status flag, a trigger pulse output TRxy of ETLx can be enabled (by bit EXICONx.PE) and selected to **trigger actions in one of the OGUy** units. The target OGUy for the trigger is selected by bit field EXICONx.OCS.

The trigger becomes active when the selected edge event is detected, independently from the status flag EXICONx.FL.

4.5.3 Cross Connect Matrix

The matrix shown in [Figure 4-7](#) distributes the trigger signals (TRxy) and status signals (EXICONx.FL) from the different ETLx units between the OGUy units. In addition, it receives peripheral trigger signals that can be OR-combined with the ETLx trigger signals in the OGUy units.

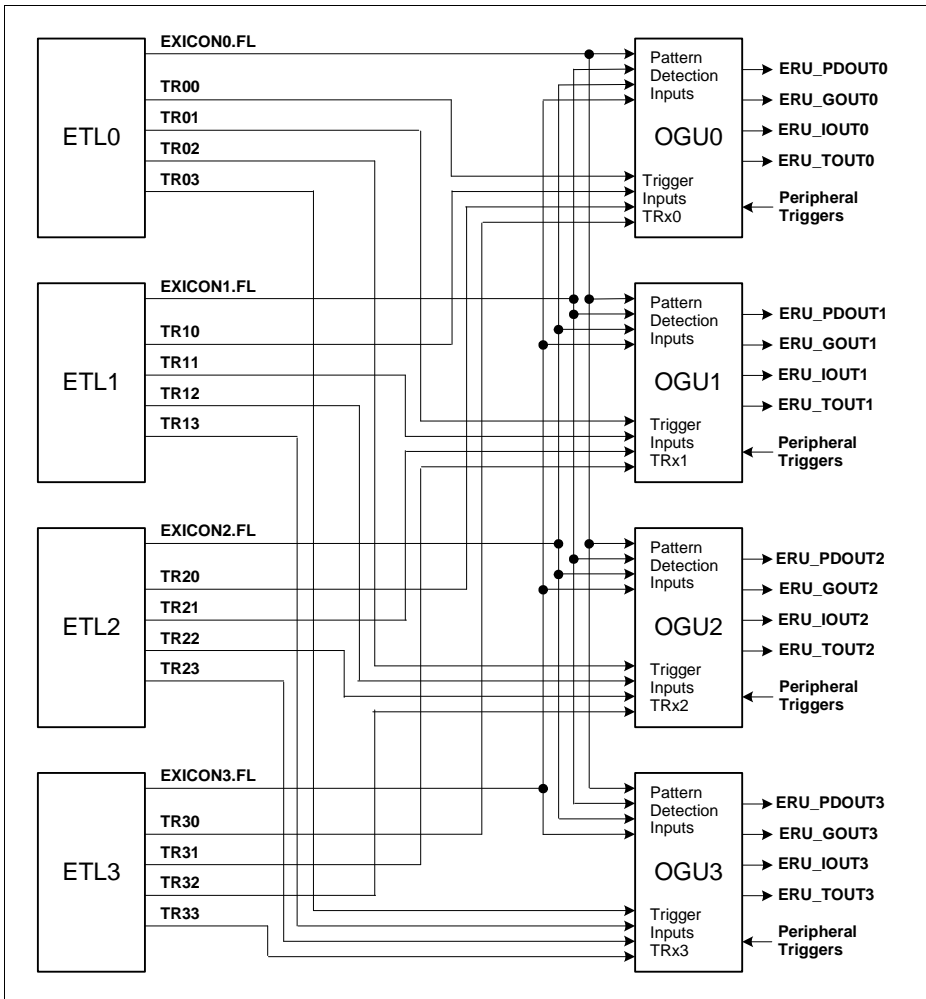


Figure 4-7 ERU Cross Connect Matrix

4.5.4 Output Gating Unit (OGUy)

Each OGUy (y = 0-3) unit combines the available trigger events and status flags from the Input Channels and distributes the results to the system. [Figure 4-8](#) illustrates the logic blocks within an OGUy unit. All functions of an OGUy unit are controlled by its associated

Service Request Processing

EXOCONy register, e.g. **ERU0_EXOCONx (x=0-3)** for OGU0. The function of an OGUy unit can be split into two parts:

- **Trigger Combination:**
All trigger signals TRxy from the Input Channels that are enabled and directed to OGUy, a selected peripheral-related trigger event, and a pattern change event (if enabled) are logically OR-combined.
- **Pattern Detection:**
The status flags EXICONx.FL of the Input Channels can be enabled to take part in the pattern detection. A pattern match is detected while all enabled status flags are set.

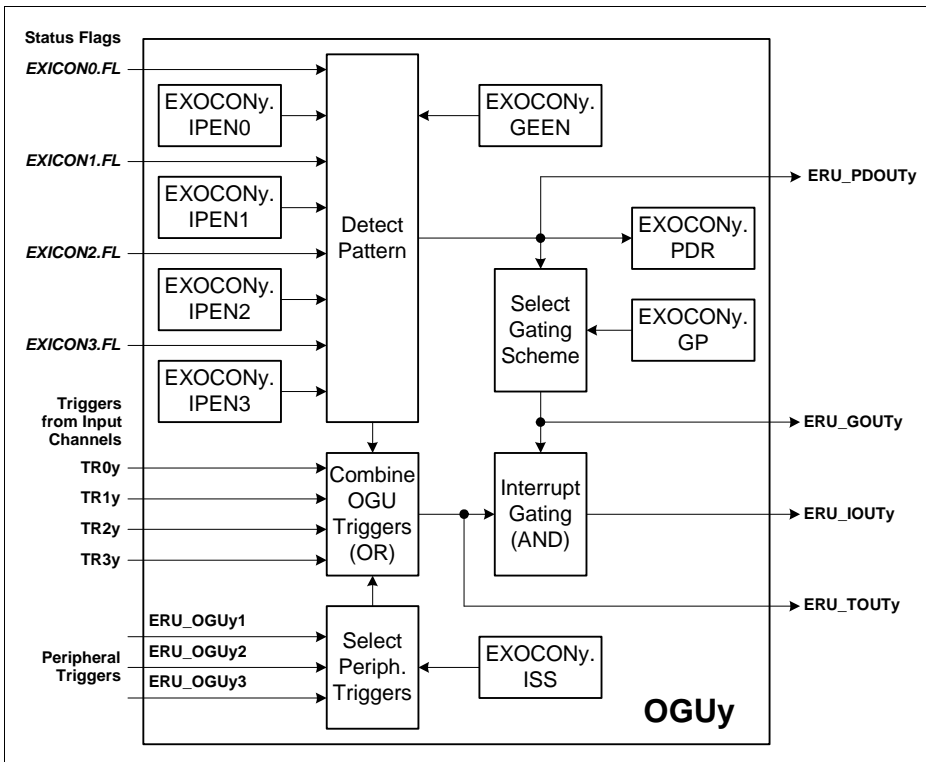


Figure 4-8 Output Gating Unit for Output Channel y

Each OGUy unit generates 4 output signals that are distributed to the system (not all of them are necessarily used):

- **ERU_PDOUTy** to directly output the pattern match information for gating purposes in other modules (pattern match = 1).

Service Request Processing

- **ERU_GOUTy** to output the pattern match or pattern miss information (inverted pattern match), or a permanent 0 or 1 under software control for gating purposes in other modules.
- **ERU_TOUTy** as combination of a peripheral trigger, a pattern detection result change event, or the ETLx trigger outputs TRxy to trigger actions in other modules.
- **ERU_IOUTy** as gated trigger output (ERU_GOUTy logical AND-combined with ERU_TOUTy) to trigger service requests (e.g. the service request generation can be gated to allow service request activation during a certain time window).

Trigger Combination

The trigger combination logically OR-combines different trigger inputs to form a common trigger ERU_TOUTy. Possible trigger inputs are:

- In each ETLx unit of the **Input Channels**, the trigger output TRxy can be enabled and the trigger event can be directed to one of the OGUy units.
- One out of three **peripheral trigger** signals per OGUy can be selected as additional trigger source. These peripheral triggers are generated by on-chip peripheral modules, such as capture/compare or timer units. The selection is done by bit field EXOCONy.ISS.
- In the case that at least one **pattern detection** input is enabled (EXOCONy.IPENx) and a change of the pattern detection result from pattern match to pattern miss (or vice-versa) is detected, a trigger event is generated to indicate a pattern detection result event (if enabled by ECOCONy.GEEN).

The trigger combination offers the possibility to program different trigger criteria for several input signals (independently for each Input Channel) or peripheral signals, and to combine their effects to a single output, e.g. to generate an service request or to start an ADC conversion. This combination capability allows the generation of a service request per OGU that can be triggered by several inputs (multitude of request sources results in one reaction).

The selection is defined by the bit fields ISS in registers **ERU0_EXOCONx (x=0-3)** (for ERU0.OGUx) and **ERU1_EXOCONy (y=0-3)** (for ERU1.OGUy).

Pattern Detection

The pattern detection logic allows the combination of the status flags of all ETLx units. Each status flag can be individually included or excluded from the pattern detection for each OGUy, via control bits EXOCONy.IPENx. The pattern detection block outputs the following pattern detection results:

- **Pattern match** (EXOCONy.PDR = 1 and ERU_PDOUTy = 1):
A pattern match is indicated while all status flags FL that are included in the pattern detection are 1.

Service Request Processing

- **Pattern miss** (EXOCONy.PDR = 0 and ERU_PDOUTy = 0):
A pattern miss is indicated while at least one of the status flags FL that are included in the pattern detection is 0.

In addition, the pattern detection can deliver a trigger event if the pattern detection result changes from match to miss or vice-versa (if enabled by EXOCONy.GEEN = 1). The pattern result change event is logically OR-combined with the other enabled trigger events to support service request generation or to trigger other module functions (e.g. in the ADC). The event is indicated when the pattern detection result changes and EXOCONy.PDR becomes updated.

The service request generation in the OGUy is based on the trigger ERU_TOUTy that can be gated (masked) with the pattern detection result ERU_PDOUTy. This allows an automatic and reproducible generation of service requests during a certain time window, where the request event is elaborated by the trigger combination block and the time window information (gating) is given by the pattern detection. For example, service requests can be issued on a regular time base (peripheral trigger input from capture/compare unit is selected) while a combination of input signals occurs (pattern detection based on ETLx status bits).

A programmable gating scheme introduces flexibility to adapt to application requirements and allows the generation of service requests ERU_IOUTy under different conditions:

- **Pattern match** (EXOCONy.GP = 10_B):
A service request is issued when a trigger event occurs while the pattern detection shows a pattern match.
- **Pattern miss** (EXOCONy.GP = 11_B):
A service request is issued when the trigger event occurs while the pattern detection shows a pattern miss.
- **Independent** of pattern detection (EXOCONy.GP = 01_B):
In this mode, each occurring trigger event leads to a service request. The pattern detection output can be used independently from the trigger combination for gating purposes of other peripherals (independent use of ERU_TOUTy and ERU_PDOUTy with service requests on trigger events).
- **No service requests** (EXOCONy.GP = 00_B, default setting)
In this mode, an occurring trigger event does not lead to a service request. The pattern detection output can be used independently from the trigger combination for gating purposes of other peripherals (independent use of ERU_TOUTy and ERU_PDOUTy without service requests on trigger events).

4.6 Service Request Generation

If any bit within the DLR.DLR_OVRSTAT register is set, a service request is flagged by setting the SCU_SRRAW.DLROVR bit.

Service Request Processing

To clear a request user must program the corresponding bit in the DLR.**DLR_OVRCLR** register. Then also clear SCU_SRRAW.DLROVR bit as described in the SCU chapter. It is possible to mask out promotion of the service request by setting SCU_SRMSK.DLROVR bit.

4.7 Debug Behavior

Service request processing behavior is unchanged in debug mode.

4.8 Power, Reset and Clock

Service request processing is

- consuming power in all operating modes.
- running on f_{CPU} .
- asynchronously initialized by the system reset.

4.9 Initialization and System Dependencies

Service Requests must always be enabled at the source and at the destination. Additionally it must be checked whether it is necessary to program the ERU process and route a request.

Enabling Peripheral SRx Outputs

- Peripherals SRx outputs must be selectively enabled. This procedure depends on the individual peripheral. Please look up the section “Service Request Generation” within a peripherals chapter for details.
- Optionally ERUx must be programmed to process and route the request

Enabling External Requests

- Selected PORTS must be programmed for input
- ERUx must be programmed to process and route the external request

Note: The number of external service request inputs may be limited by the package used.

Enabling NVIC and GPDMA

Interrupt and DMA service request processing must be enabled. Please refer to the CPU and GPDMA chapters for details.

4.10 Registers

Registers Overview

The absolute register address is calculated by adding:
Module Base Address + Offset Address

Table 4-6 Registers Address Space

Module	Base Address	End Address	Note
DLR	5000 4900 _H	5000 49FF _H	
ERU0	5000 4800 _H	5000 48FF _H	
ERU1	4004 4000 _H	4004 7FFF _H	

Table 4-7

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	

DLR Registers

OVRSTAT	Status of DMA Service Request Overruns	000 _H	U, PV	PV	Page 4-24
OVRCLR	Clear Status of DMA Service Request Overruns	004 _H	U, PV	PV	Page 4-25
SRSEL0	DLR Service Request Selection 0	008 _H	U, PV	PV	Page 4-26
LNEN	Enable DLR Line	010 _H	U, PV	PV	Page 4-25

ERU Registers

EXISEL	ERU External Input Control Selection	0000 _H	U, PV	PV	Page 4-27
EXICON0	ERU External Input Control Selection	0010 _H	U, PV	PV	Page 4-29
EXICON1	ERU External Input Control Selection	0014 _H	U, PV	PV	Page 4-29
EXICON2	ERU External Input Control Selection	0018 _H	U, PV	PV	Page 4-29
EXICON3	ERU External Input Control Selection	001C _H	U, PV	PV	Page 4-29

Table 4-7 (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
EXOCON0	ERU Output Control Register	0020 _H	U, PV	PV	Page 4-31
EXOCON1	ERU Output Control Register	0024 _H	U, PV	PV	Page 4-31
EXOCON2	ERU Output Control Register	0028 _H	U, PV	PV	Page 4-31
EXOCON3	ERU Output Control Register	002C _H	U, PV	PV	Page 4-31

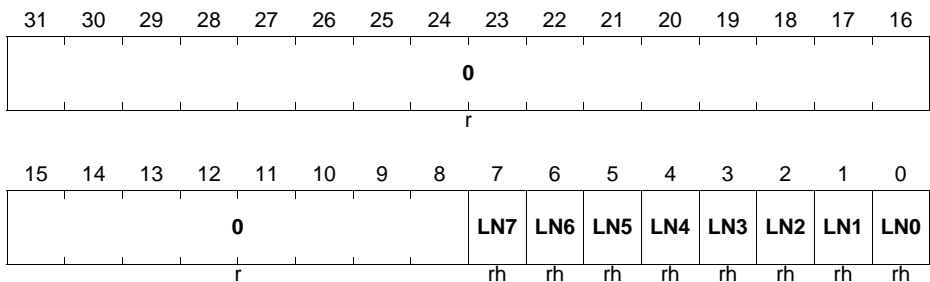
4.10.1 DLR Registers

DLR_OVRSTAT

The DLR_OVRSTAT register is used to track status of GPDMA service request overruns. Upon overrun detection, additionally a service request flag is set in the SCU_SRRAW.DLROVR bit.

DLR_OVRSTAT

Overrun Status (00_H) **Reset Value: 0000 0000_H**



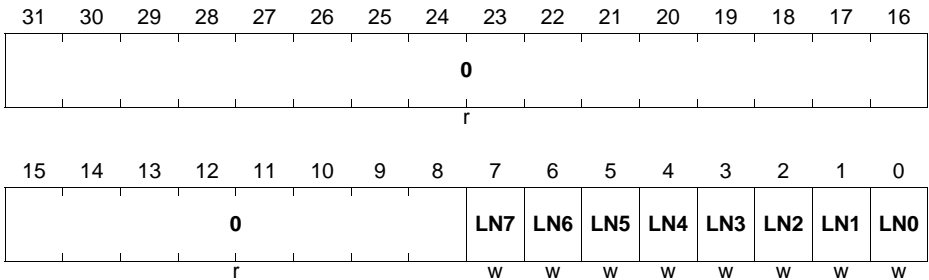
Field	Bits	Type	Description
LNx (x = 0-7)	x	rh	Line x Overrun Status Set if an overrun occurred on this line.
0	[31:8]	r	Reserved Read as 0; should be written with 0.

DLR_OVRCLR

The DLR_OVRCLR register is used to clear the DLR_OVRSTAT register bits.

DLR_OVRCLR

Overrun Clear (04_H) **Reset Value: 0000 0000_H**



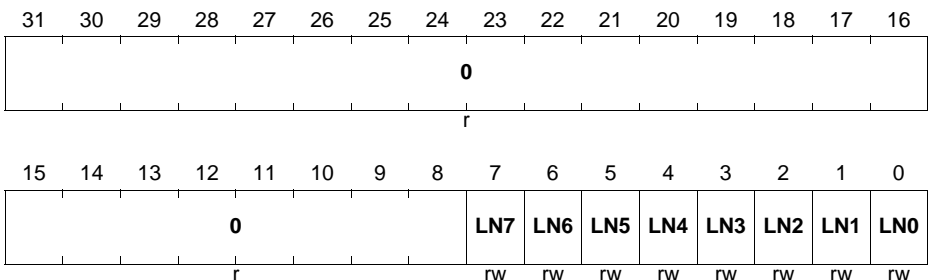
Field	Bits	Type	Description
LN_x (x = 0-7)	x	w	Line x Overrun Status Clear Clears the corresponding bit in the DLR_OVRSTAT register when set to 1.
0	[31:8]	r	Reserved Read as 0; should be written with 0.

DLR_LNEN

The DLR_LNEN register is used to enable each individual DLR line and to reset a previously stored and pending service request.

DLR_LNEN

Line Enable (10_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
LNx (x = 0-7)	x	rw	Line x Enable 0 _B Disables the line 1 _B Enables the line and resets a pending request
0	[31:8]	r	Reserved Read as 0; should be written with 0.

DLR_SRSELO

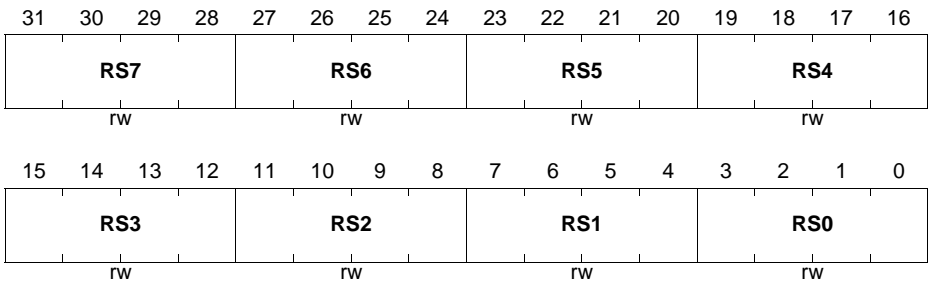
The DLR_SRSELO register is used to select the service request source used to trigger a DMA transfer.

DLR_SRSELO

Service Request Selection 0

(08_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RSx (x = 0-7)	[x*4+3: x*4]	rw	Request Source for Line x The request source according to Table 4-5 is selected for DMA line x. These lines are connected to GPDMA0

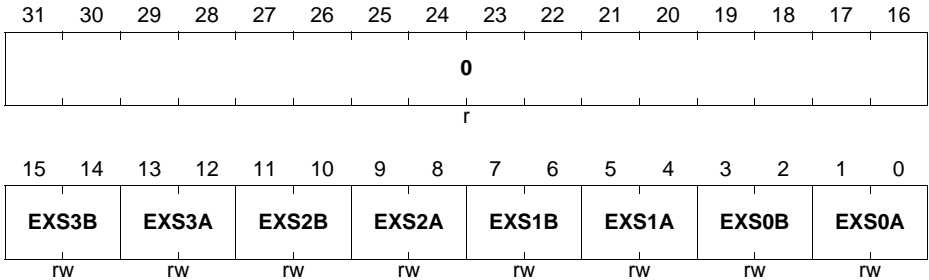
4.10.2 ERU Registers

ERU0_EXISEL

Event Input Select (00_H) Reset Value: 0000 0000_H

ERU1_EXISEL

Event Input Select (0000_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
EXS0A	[1:0]	rw	Event Source Select for A0 (ERS0) This bit field defines which input is selected for A0. 00 _B Input ERU_0A0 is selected 01 _B Input ERU_0A1 is selected 10 _B Input ERU_0A2 is selected 11 _B Input ERU_0A3 is selected
EXS0B	[3:2]	rw	Event Source Select for B0 (ERS0) This bit field defines which input is selected for B0. 00 _B Input ERU_0B0 is selected 01 _B Input ERU_0B1 is selected 10 _B Input ERU_0B2 is selected 11 _B Input ERU_0B3 is selected
EXS1A	[5:4]	rw	Event Source Select for A1 (ERS1) This bit field defines which input is selected for A1. 00 _B Input ERU_1A0 is selected 01 _B Input ERU_1A1 is selected 10 _B Input ERU_1A2 is selected 11 _B Input ERU_1A3 is selected

Service Request Processing

Field	Bits	Type	Description
EXS1B	[7:6]	rw	Event Source Select for B1 (ERS1) This bit field defines which input is selected for B1. 00 _B Input ERU_1B0 is selected 01 _B Input ERU_1B1 is selected 10 _B Input ERU_1B2 is selected 11 _B Input ERU_1B3 is selected
EXS2A	[9:8]	rw	Event Source Select for A2 (ERS2) This bit field defines which input is selected for A2. 00 _B Input ERU_2A0 is selected 01 _B Input ERU_2A1 is selected 10 _B Input ERU_2A2 is selected 11 _B Input ERU_2A3 is selected
EXS2B	[11:10]	rw	Event Source Select for B2 (ERS2) This bit field defines which input is selected for B2. 00 _B Input ERU_2B0 is selected 01 _B Input ERU_2B1 is selected 10 _B Input ERU_2B2 is selected 11 _B Input ERU_2B3 is selected
EXS3A	[13:12]	rw	Event Source Select for A3 (ERS3) This bit field defines which input is selected for A3. 00 _B Input ERU_3A0 is selected 01 _B Input ERU_3A1 is selected 10 _B Input ERU_3A2 is selected 11 _B Input ERU_3A3 is selected
EXS3B	[15:14]	rw	Event Source Select for B3 (ERS3) This bit field defines which input is selected for B3. 00 _B Input ERU_3B0 is selected 01 _B Input ERU_3B1 is selected 10 _B Input ERU_3B2 is selected 11 _B Input ERU_3B3 is selected
0	[31:16]	r	Reserved Read as 0; should be written with 0.

ERU0_EXICONx (x=0-3)

Event Input Control x

(10_H + 4*x)

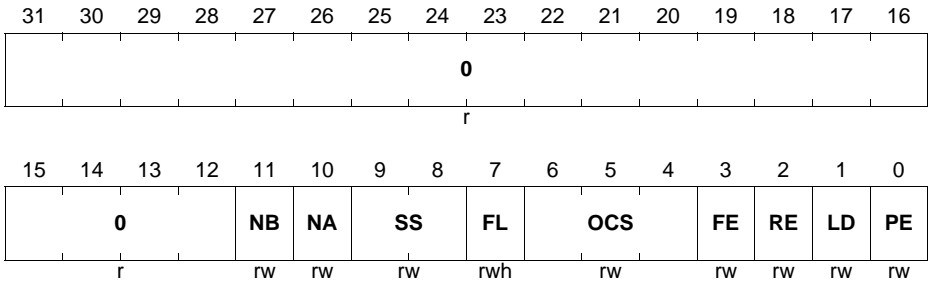
Reset Value: 0000 0000_H

ERU1_EXICONy (y=0-3)

Event Input Control y

(0010_H + 4*y)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PE	0	rw	<p>Output Trigger Pulse Enable for ETLx</p> <p>This bit enables the generation of an output trigger pulse at TRxy when the selected edge is detected (set condition for the status flag FL).</p> <p>0_B The trigger pulse generation is disabled 1_B The trigger pulse generation is enabled</p>
LD	1	rw	<p>Rebuild Level Detection for Status Flag for ETLx</p> <p>This bit selects if the status flag FL is used as “sticky” bit or if it rebuilds the result of a level detection.</p> <p>0_B The status flag FL is not cleared by hardware and is used as “sticky” bit. Once set, it is not influenced by any edge until it becomes cleared by software. 1_B The status flag FL rebuilds a level detection of the desired event. It becomes automatically set with a rising edge if RE = 1 or with a falling edge if FE = 1. It becomes automatically cleared with a rising edge if RE = 0 or with a falling edge if FE = 0.</p>

Service Request Processing

Field	Bits	Type	Description
RE	2	rw	<p>Rising Edge Detection Enable ETLx This bit enables/disables the rising edge event as edge event as set condition for the status flag FL or as possible trigger pulse for TRxy.</p> <p>0_B A rising edge is not considered as edge event 1_B A rising edge is considered as edge event</p>
FE	3	rw	<p>Falling Edge Detection Enable ETLx This bit enables/disables the falling edge event as edge event as set condition for the status flag FL or as possible trigger pulse for TRxy.</p> <p>0_B A falling edge is not considered as edge event 1_B A falling edge is considered as edge event</p>
OCS	[6:4]	rw	<p>Output Channel Select for ETLx Output Trigger Pulse This bit field defines which Output Channel OGUy is targeted by an enabled trigger pulse TRxy.</p> <p>000_B Trigger pulses are sent to OGU0 001_B Trigger pulses are sent to OGU1 010_B Trigger pulses are sent to OGU2 011_B Trigger pulses are sent to OGU3 Others: Reserved, do not use this combination</p>
FL	7	rwh	<p>Status Flag for ETLx This bit represents the status flag that becomes set or cleared by the edge detection.</p> <p>0_B The enabled edge event has not been detected 1_B The enabled edge event has been detected</p>
SS	[9:8]	rw	<p>Input Source Select for ERSx This bit field defines which logical combination is taken into account as ERSxO.</p> <p>00_B Input A without additional combination 01_B Input B without additional combination 10_B Input A OR input B 11_B Input A AND input B</p>
NA	10	rw	<p>Input A Negation Select for ERSx This bit selects the polarity for the input A.</p> <p>0_B Input A is used directly 1_B Input A is inverted</p>

Service Request Processing

Field	Bits	Type	Description
NB	11	rw	Input B Negation Select for ERSx This bit selects the polarity for the input B. 0 _B Input B is used directly 1 _B Input B is inverted
0	[31:12]	r	Reserved Read as 0; should be written with 0.

ERU0_EXOCONx (x=0-3)

Event Output Trigger Control x

$$(20_H + 4^*x)$$

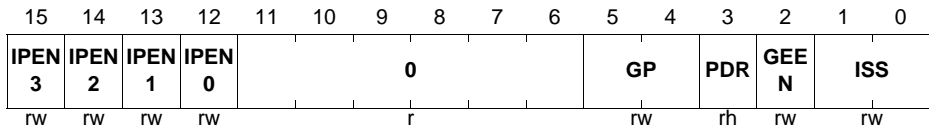
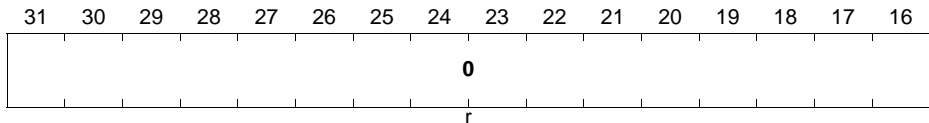
Reset Value: 0000 0008_H

ERU1_EXOCONy (y=0-3)

Event Output Trigger Control y

$$(0020_H + 4^*y)$$

Reset Value: 0000 0008_H



Field	Bits	Type	Description
ISS	[1:0]	rw	Internal Trigger Source Selection This bit field defines which input is selected as peripheral trigger input for OGUy. 00 _B The peripheral trigger function is disabled 01 _B Input ERU_OGUy1 is selected 10 _B Input ERU_OGUy2 is selected 11 _B Input ERU_OGUy3 is selected
GEEN	2	rw	Gating Event Enable Bit GEEN enables the generation of a trigger event when the result of the pattern detection changes from match to miss or vice-versa. 0 _B The event detection is disabled 1 _B The event detection is enabled

Service Request Processing

Field	Bits	Type	Description
PDR	3	rh	<p>Pattern Detection Result Flag</p> <p>This bit represents the pattern detection result.</p> <p>0_B A pattern miss is detected</p> <p>1_B A pattern match is detected</p>
GP	[5:4]	rw	<p>Gating Selection for Pattern Detection Result</p> <p>This bit field defines the gating scheme for the service request generation (relation between the OGU output ERU_PDOUTy and ERU_GOUTy).</p> <p>00_B ERU_GOUTy is always disabled and ERU_IOUTy can not be activated</p> <p>01_B ERU_GOUTy is always enabled and ERU_IOUTy becomes activated with each activation of ERU_TOUTy</p> <p>10_B ERU_GOUTy is equal to ERU_PDOUTy and ERU_IOUTy becomes activated with an activation of ERU_TOUTy while the desired pattern is detected (pattern match PDR = 1)</p> <p>11_B ERU_GOUTy is inverted to ERU_PDOUTy and ERU_IOUTy becomes activated with an activation of ERU_TOUTy while the desired pattern is not detected (pattern miss PDR = 0)</p>
IPENx (x = 0-3)	12+x	rw	<p>Pattern Detection Enable for ETLx</p> <p>Bit IPENx defines whether the trigger event status flag EXICONx.FL of ETLx takes part in the pattern detection of OGUy.</p> <p>0_B Flag EXICONx.FL is excluded from the pattern detection</p> <p>1_B Flag EXICONx.FL is included in the pattern detection</p>
0	[31:16] , [11:6]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

4.11 Interconnects

This section describes how the ERU0 and ERU1 modules are connected within the XMC4300 system.

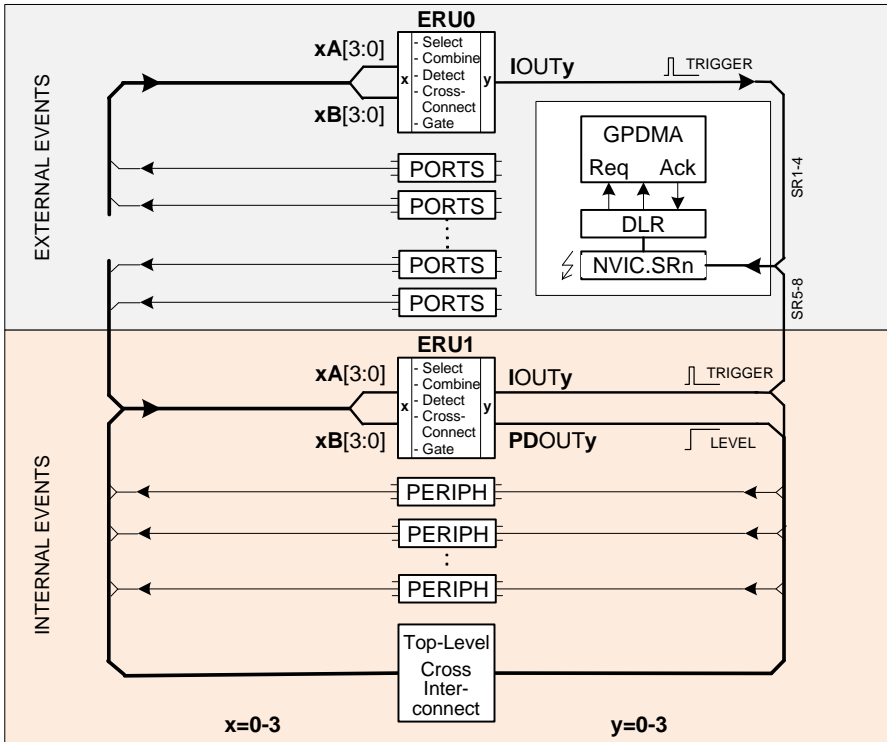


Figure 4-9 ERU Interconnects Overview

4.11.1 ERU0 Connections

The following table shows the ERU0 connections. Please refer to the ports chapter for details about PORTS connections.

Table 4-8 ERU0 Pin Connections

Global Inputs/Outputs	I/O	Connected To	Description
ERU0.0A0	I	PORTS	
ERU0.0A1	I	PORTS	
ERU0.0A2	I	PORTS	
ERU0.0A3	I	SCU.GOORCOUT6	

Table 4-8 ERU0 Pin Connections (cont'd)

Global Inputs/Outputs	I/O	Connected To	Description
ERU0.0B0	I	PORTS	
ERU0.0B1	I	PORTS	
ERU0.0B2	I	PORTS	
ERU0.0B3	I	PORTS	
ERU0.1A0	I	PORTS	
ERU0.1A1	I	SCU.HIB_SR0	
ERU0.1A2	I	PORTS	
ERU0.1A3	I	SCU.G0ORCOUT7	
ERU0.1B0	I	PORTS	
ERU0.1B1	I	SCU.HIB_SR1	
ERU0.1B2	I	PORTS	
ERU0.1B3	I	PORTS	
ERU0.2A0	I	PORTS	
ERU0.2A1	I	PORTS	
ERU0.2A2	I	PORTS	
ERU0.2A3	I	SCU.G1ORCOUT6	
ERU0.2B0	I	PORTS	
ERU0.2B1	I	PORTS	
ERU0.2B2	I	PORTS	
ERU0.2B3	I	PORTS	
ERU0.3A0	I	PORTS	
ERU0.3A1	I	PORTS	
ERU0.3A2	I	PORTS	
ERU0.3A3	I	SCU.G1ORCOUT7	
ERU0.3B0	I	PORTS	
ERU0.3B1	I	PORTS	
ERU0.3B2	I	PORTS	
ERU0.3B3	I	PORTS	
ERU0.0GU01	I	0	
ERU0.0GU02	I	0	

Table 4-8 ERU0 Pin Connections (cont'd)

Global Inputs/Outputs	I/O	Connected To	Description
ERU0.OGU03	I	1	
ERU0.OGU11	I	0	
ERU0.OGU12	I	0	
ERU0.OGU13	I	1	
ERU0.OGU21	I	0	
ERU0.OGU22	I	0	
ERU0.OGU23	I	1	
ERU0.OGU31	I	0	
ERU0.OGU32	I	0	
ERU0.OGU33	I	1	
ERU0.PDOUT0	O	not connected	
ERU0.GOUT0	O	not connected	
ERU0.TOUT0	O	not connected	
ERU0.IOUT0	O	NVIC.ERU0.SR0 DLR	
ERU0.PDOUT1	O	not connected	
ERU0.GOUT1	O	not connected	
ERU0.TOUT1	O	not connected	
ERU0.IOUT1	O	NVIC.ERU0.SR1 DLR	
ERU0.PDOUT2	O	not connected	
ERU0.GOUT2	O	not connected	
ERU0.TOUT2	O	not connected	
ERU0.IOUT2	O	NVIC.ERU0.SR2 DLR	
ERU0.PDOUT3	O	not connected	
ERU0.GOUT3	O	not connected	
ERU0.TOUT3	O	not connected	
ERU0.IOUT3	O	NVIC.ERU0.SR3 DLR	

4.11.2 ERU1 Connections

The following table shows the ERU1 connections. Please refer to the ports chapter for details about PORTS connections.

Table 4-9 ERU1 Pin Connections

Global Inputs/Outputs	I/O	Connected To	Description
ERU1.0A0	I	PORTS	
ERU1.0A1	I	not connected	
ERU1.0A2	I	CCU40.ST0	
ERU1.0A3	I	DAC.SIGN_0	
ERU1.0B0	I	PORTS	
ERU1.0B1	I	CCU80.ST0	
ERU1.0B2	I	VADC.G0BFL3	
ERU1.0B3	I	ERU1.IOUT3	
ERU1.1A0	I	PORTS	
ERU1.1A1	I	not connected	
ERU1.1A2	I	CCU40.ST1	
ERU1.1A3	I	ERU1.IOUT2	
ERU1.1B0	I	PORTS	
ERU1.1B1	I	CCU80.ST1	
ERU1.1B2	I	VADC.G1BFL3	
ERU1.1B3	I	ERU1.IOUT2	
ERU1.2A0	I	PORTS	
ERU1.2A1	I	not connected	
ERU1.2A2	I	CCU40.ST2	
ERU1.2A3	I	DAC.SIGN_1	
ERU1.2B0	I	PORTS	
ERU1.2B1	I	CCU80.ST2	
ERU1.2B2	I	VADC.G0BFL3	
ERU1.2B3	I	not connected	
ERU1.3A0	I	PORTS	
ERU1.3A1	I	not connected	

Table 4-9 ERU1 Pin Connections (cont'd)

Global Inputs/Outputs	I/O	Connected To	Description
ERU1.3A2	I	CCU40.ST3	
ERU1.3A3	I	not connected	
ERU1.3B0	I	PORTS	
ERU1.3B1	I	CCU80.ST3	
ERU1.3B2	I	VADC.G1BFL3	
ERU1.3B3	I	not connected	
ERU1.OGU01	I	VADC.C0SR0	
ERU1.OGU02	I	CCU40.ST0	
ERU1.OGU03	I	1	
ERU1.OGU11	I	VADC.C0SR1	
ERU1.OGU12	I	CCU41.ST0	
ERU1.OGU13	I	1	
ERU1.OGU21	I	VADC.C0SR2	
ERU1.OGU22	I	not connected	
ERU1.OGU23	I	1	
ERU1.OGU31	I	VADC.C0SR3	
ERU1.OGU32	I	not connected	
ERU1.OGU33	I	1	

Table 4-9 ERU1 Pin Connections (cont'd)

Global Inputs/Outputs	I/O	Connected To	Description
ERU1.PDOUT0	O	CCU40.IN0J CCU41.IN0J CCU40.IN1D CCU40.IN2D CCU40.IN3D CCU41.IN1D CCU41.IN2D CCU41.IN3D CCU80.IN0J CCU80.IN1J CCU80.IN2J CCU80.IN3J VADC.G0REQGTO VADC.G1REQGTO VADC.BGREQGTO PORTS	
ERU1.GOUT0	O	not connected	
ERU1.TOUT0	O	not connected	
ERU1.IOUT0	O	CCU40.IN0K CCU41.IN0K CCU80.IN0G VADC.G0REQTRM VADC.G1REQTRM VADC.BGREQTRM CCU40.MCLKA CCU41.MCLKA CCU80.MCLKA NVIC.ERU1.SR0	
ERU1.PDOUT1	O	CCU40.IN1J CCU41.IN1J CCU40.IN0D CCU41.IN0D VADC.G0REQGTP VADC.G1REQGTP VADC.BGREQGTP PORTS	
ERU1.GOUT1	O	not connected	

Table 4-9 ERU1 Pin Connections (cont'd)

Global Inputs/Outputs	I/O	Connected To	Description
ERU1.TOUT1	O	not connected	
ERU1.IOUT1	O	CCU40.IN1K CCU41.IN1K CCU80.IN1G VADC.G0REQTRN VADC.G1REQTRN VADC.BGREQTRN CCU40.MCLKB CCU41.MCLKB CCU80.MCLKB NVIC.ERU1.SR1	
ERU1.PDOUT2	O	CCU40.IN2J CCU41.IN2J CCU80.IN2F PORTS	
ERU1.GOUT2	O	not connected	
ERU1.TOUT2	O	not connected	
ERU1.IOUT2	O	CCU40.IN2K CCU41.IN2K CCU80.IN2G ERU1.1B3 NVIC.ERU1.SR2	
ERU1.PDOUT3	O	CCU40.IN3J CCU41.IN3J CCU80.IN3F PORTS	
ERU1.GOUT3	O	not connected	
ERU1.TOUT3	O	not connected	
ERU1.IOUT3	O	CCU40.IN3K CCU41.IN3K CCU80.IN3G ERU1.0B3 NVIC.ERU1.SR3	

5 General Purpose DMA (GPDMA)

The GPDMA is a highly configurable DMA controller, that allows high-speed data transfers between peripherals and memories. Complex data transfers can be done with minimal intervention of the processor, keeping this way the CPU resources free for other operations.

Extensive support for the microcontroller peripherals, like A/D and D/A converters, Timers, Communication Interfaces (USIC) via the GPDMA, unload the CPU and increase the efficiency and parallelism, for a high arrangement of real-time applications.

Table 5-1 Abbreviations table

GPDMA _x	General Purpose DMA instance x
SCU	System Control Unit
DLR	DMA Line Router
f_{DMA}	GPDMA clock frequency

5.1 Overview

The GPDMA module enables hardware or software controlled data transfers between all microcontroller modules with the exclusion of those modules which provide built-in DMA functionality (USB and Ethernet).

Each GPDMA module contains a dedicated set of highly programmable channels, that can accommodate several type of peripheral-to-peripheral, peripheral-to-memory and memory-to-memory transfers.

The link between a highly programmable channel allocation and channel priority, gives a high benefit for applications that need high efficiency and parallelism.

The built-in fast DMA request handling together with the flexible peripheral configuration, enables the implementation of very demanding application software loops.

5.1.1 Features

The GPDMA component includes the following features.

General

- Bus interfaces
 - 1 Bus master interface per DMA unit
 - 1 Bus slave interface per DMA unit
- Channels
 - One GPDMA0 unit with 8 channels
 - Programmable channel priority
- Transfers

General Purpose DMA (GPDMA)

- Support for memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral DMA transfers

Channels

All channels can be programmed for the following transfer modes

- DMA triggered by software or selectable from hardware service request sources
- Programmable source and destination addresses
- Address increment, decrement, or no change

Channels 0 and 1 of GPDMA0 can be programmed for the following transfer modes

- Multi-block transfers achieved through:
 - Linked Lists (block chaining)
 - Auto-reloading of channel registers
 - Contiguous address between blocks
- Independent source and destination selection of multi-block transfer type
- Scatter/Gather - source and destination areas do not need to be in a contiguous memory space

The GPDMA0 channels 0 and 1 provide a FIFO of 32 Bytes (eight 32-bit entries). These channels can be used to execute burst transfers up to a fixed length burst size of 8. The remaining channels FIFO size is 8 Bytes.

Channel Control

- Programmable source and destination for each channel
- Programmable burst transaction size for each channel
- Programmable enable and disable of DMA channel
- Support for disabling channel without data loss
- Support for suspension of DMA operation
- Support for ERROR response
- Bus locking - programmable over transaction, block, or DMA transfer level
- Channel locking - programmable over transaction, block, or DMA transfer level
- Optional writeback of the Channel Control register at the end of every block transfer

Interrupts

- Combined and separate interrupt service requests
- Request generation on:
 - DMA transfer completion
 - Block transfer completion
 - Single and burst transaction completion
 - Error condition
- Support of interrupt enabling and masking

5.1.2 Block Diagram

Figure 5-1 shows the following functional groupings of the main interfaces to the GPDMA block:

- DMA hardware request interface (DLR)
- Up to eight channels
- Arbiter
- Bus Master and Slave interfaces

One channel of the GPDMA is required for each source/destination pair. The master interface reads the data from a source peripheral and writes it to a destination peripheral. Two physical transfers are therefore required for each DMA transaction.

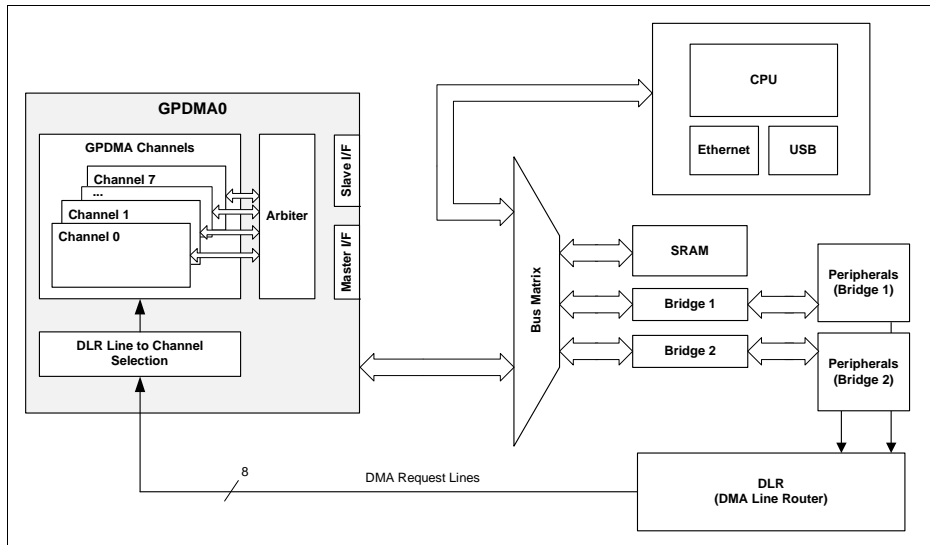


Figure 5-1 GPDMA Block Diagram

5.2 Functional Description

This chapter describes the functional details of the GPDMA.

5.2.1 Terminology

The following terms are concise definitions of the DMA concepts are used throughout this chapter:

Service Partner Terms

- **Source peripheral** - Device from which the GPDMA reads data; the GPDMA then stores the data in the channel FIFO. The source peripheral teams up with a destination peripheral to form a channel.
- **Destination peripheral** - Device to which the GPDMA writes the stored data from the FIFO (previously read from the source peripheral).
- **Memory** - Source or destination that is always "ready" for a DMA transfer and does not require a handshaking interface to interact with the GPDMA. Note that
- **Channel** - Read/write data path between a source peripheral and a destination peripheral, that occurs through the channel FIFO. If the source peripheral is not memory, then a source handshaking interface is assigned to the channel. If the destination peripheral is not memory, then a destination handshaking interface is assigned to the channel. Source and destination handshaking interfaces can be assigned dynamically by programming the channel registers.

Interface Terms

- **Master interface** - GPDMA is a master on the AHB, reading data from the source and writing it to the destination over the bus. Each channel has to arbitrate for the master interface.
- **Slave interface** - The AHB interface over which the GPDMA is programmed.
- **Handshaking interface** - A set of signals or software registers that conform to a protocol and handshake between the GPDMA and source or destination peripheral in order to control transferring a single or burst transaction between them. This interface is used to request, acknowledge, and control a GPDMA transaction. A channel can receive a request through one of two types of handshaking interface: software, or peripheral trigger.
 - **Software handshaking interface**- Software uses registers to control transferring a single or burst transaction between the GPDMA and the source or destination peripheral. This mode is useful if the total block size is unknown at the beginning of a transfer. For more information about this interface, refer to [Section 5.2.4.2](#).
 - **Peripheral trigger interface** - In this mode, a DLR service request line is used to trigger single or burst transactions. For using this mode, the total block size must be known at the beginning of a transfer. For more information about this interface, refer to [Section 5.2.4](#).

Flow Control Terms

- **Flow controller** - Device that determines the length of a DMA block transfer and terminates it.
 - If you know the length of a block before enabling the channel, then GPDMA should be programmed as the flow controller.
 - If the length of a block is not known prior to enabling the channel, the source or destination peripheral needs to control and terminate a transfer. In this mode, the peripheral, using the software handshaking interface, is the flow controller.
- **Flow control mode (CFG.FCMODE)** - Special mode that only applies when the destination peripheral is the flow controller. It controls the data pre-fetching from the source peripheral.

Transfer Terms

- **Transfer hierarchy** - **Figure 5-2** illustrates the hierarchy between GPDMA transfers, block transfers, transactions (single or burst), and AHB transfers (single or burst) for peripherals. **Figure 5-3** shows the transfer hierarchy for memory.

Note: For memory type transfers, there is no “Transaction Level”.

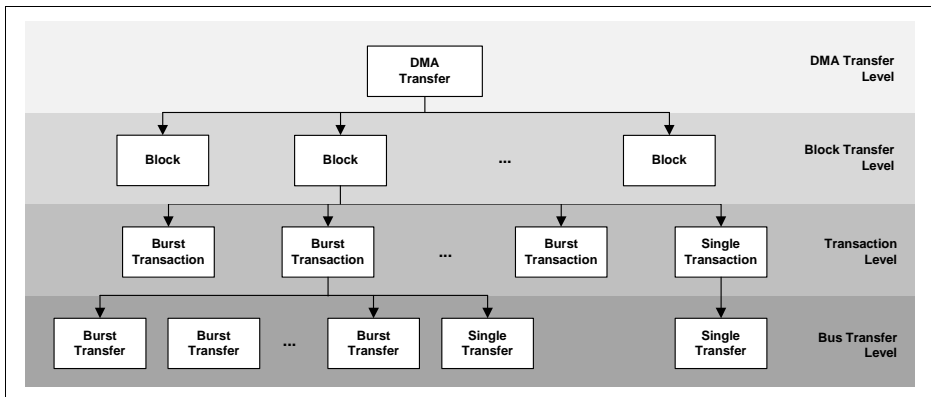


Figure 5-2 Transfer Hierarchy for Peripherals

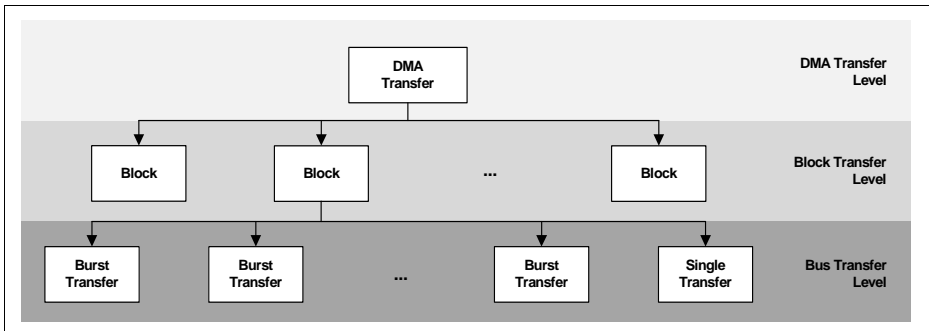


Figure 5-3 Transfer Hierarchy for Memory

- **DMA transfer** - Can be programmed to single or multiple blocks (depends on channel features). Once a DMA transfer has finished, the hardware within the GPDMA disables the channel and can generate an interrupt to signal the DMA transfer completion. You can then reprogram the channel for a new DMA transfer.
 - **Single-block DMA transfer** - Consists of a single block.
 - **Multi-block DMA transfer** - DMA transfer may consist of multiple GPDMA blocks. Multi-block DMA transfers are supported through **Linked lists (block chaining)**, **Auto-reloading** and **Contiguous blocks**. The source and destination can independently select which method to use.
- **Block** - Block of GPDMA data, the amount of which is the block length and is determined by the flow controller. For transfers between the GPDMA and memory, a block is broken directly into a sequence of bursts and single transfers. For transfers between the GPDMA and a peripheral, a block is broken into a sequence of GPDMA transactions (single and bursts). These are in turn broken into a sequence of AHB transfers.
- **Transaction** - Basic unit of a GPDMA transfer, as determined by either the hardware or software handshaking interface. A transaction is relevant only for transfers between the GPDMA and a source or destination peripheral. There are two types of transactions:
 - **Single transaction** - is always converted to a single AHB transfer.
 - **Burst transaction** - Length of a burst transaction is programmed into the GPDMA. The burst transaction is converted into a sequence of AHB fixed length bursts and AHB single transfers. GPDMA executes each burst transfer by performing incremental bursts that are no longer than the maximum burst size set; the only type of burst in this kind of transaction is incremental. The burst transaction length is under program control and normally bears some relationship to the FIFO sizes in the GPDMA and in the source and destination peripherals.

Specific Transfer Mode Terms

- **Scatter** - Relevant to destination transfers within a block. The destination address is incremented or decremented by a programmed amount when a scatter boundary is reached. The number of AHB transfers between successive scatter boundaries is under software control.
- **Gather** - Relevant to source transfers within a block. The source address is incremented or decremented by a programmed amount when a gather boundary is reached. The number of AHB transfers between successive gather boundaries is under software control.
- **Channel locking** - Software can program a channel to keep the AHB master interface by locking arbitration of the master AHB interface for the duration of a DMA transfer, block, or transaction (single or burst).
- **Bus locking** - Software can program a channel to maintain control of the AHB bus for the duration of a DMA transfer, block, or transaction (single or burst). At minimum, channel locking is asserted during bus locking.
- **FIFO mode** - Special mode to improve bandwidth. When enabled, the channel waits until the FIFO is less than half full to fetch the data from the source peripheral, and waits until the FIFO is greater than or equal to half full in order to send data to the destination peripheral. Because of this, the channel can transfer the data using bursts, which eliminates the need to arbitrate for the AHB master interface in each single AHB transfer. When this mode is not enabled, the channel waits only until the FIFO can transmit or accept a single AHB transfer before it requests the master bus interface.

5.2.2 Variable Definitions

The following variable definitions are used in this chapter:

Source single transaction size in bytes

```
src_single_size_bytes = CTLL.SRC_TR_WIDTH/8
```

Source burst transaction size in bytes

```
src_burst_size_bytes = CTLL.SRC_MSIZ * src_single_size_bytes
```

Destination single transaction size in bytes

```
dst_single_size_bytes = CTLL.DST_TR_WIDTH/8
```

Destination burst transaction size in bytes

```
dst_burst_size_bytes = CTLL.DEST_MSIZ * dst_single_size_bytes
```

Block size in bytes

- GPDMA is the flow controller:
With the GPDMA as the flow controller, the processor programs the GPDMA with the number of data items (block size) of source transfer width (**CTL.SRC_TR_WIDTH**) to be transferred by the GPDMA in a block transfer; this is programmed into the **CTL.BLOCK_TS** field. Therefore, the total number of bytes to be transferred in a block is defined by:

$$\text{blk_size_bytes_dma} = \text{CTL.BLOCK_TS} * \text{src_single_size_bytes}$$

- Source peripheral is block flow controller:
$$\text{blk_size_bytes_src} = (\text{Number of source burst transactions in block} * \text{src_burst_size_bytes}) + (\text{Number of source single transactions in block} * \text{src_single_size_bytes})$$
- Destination peripheral is block flow controller:
$$\text{blk_size_bytes_dst} = (\text{Number of destination burst transactions in block} * \text{dst_burst_size_bytes}) + (\text{Number of destination single transactions in block} * \text{dst_single_size_bytes})$$

*Note: In the above equations, references to **CTL.SRC_MSIZ**, **CTL.DEST_MSIZ**, **CTL.SRC_TR_WIDTH**, and **CTL.DST_TR_WIDTH** refer to the decoded values of the parameters; for example, **CTL.SRC_MSIZ** = 001_B decodes to 4, and **CTL.SRC_TR_WIDTH** = 010_B decodes to 32 bits.*

5.2.3 Flow Controller and Transfer Type

The device that controls the length of a block is known as the flow controller. Either the GPDMA, the source peripheral, or the destination peripheral must be assigned as the flow controller.

- If the block size is known prior to when the channel is enabled, then the GPDMA must be programmed as the flow controller. The block size is programmed into the **CTL.BLOCK_TS** field.
- If the block size is unknown when the GPDMA channel is enabled, either the source or destination peripheral must be the flow controller.

Attention: If a peripheral is assigned as the flow controller then hardware handshaking is not supported.

The **CTL.TT_FC** field indicates the transfer type and flow controller for that channel.

Table 5-2 lists valid transfer types and flow controller combinations.

Table 5-2 Transfer Type, Flow Control and Handshake Combinations

Transfer Type	Flow Controller	Handshaking
Memory to Memory	GPDMA	-
Memory to Peripheral	GPDMA	Hardware or Software
Peripheral to Memory	GPDMA	Hardware or Software
Peripheral to Peripheral	GPDMA	Hardware or Software
Peripheral to Memory	Peripheral	Software
Peripheral to Peripheral	Source Peripheral	Software
Memory to Peripheral	Peripheral	Software
Peripheral to Peripheral	Destination Peripheral	Software

5.2.4 Handshaking Interface

Handshaking interfaces are used at the transaction level to control the flow of single or burst transactions. The operation of the handshaking interface depends on whether the peripheral or the GPDMA is the flow controller.

The peripheral uses the handshaking interface to indicate to the GPDMA that it is ready to transfer data over the AHB bus.

A peripheral can request a DMA transaction through the GPDMA using one of two types of handshaking interfaces:

- Hardware
- Software

The user selects between the hardware or software handshaking interface on a per-channel basis. Software handshaking is accomplished through memory-mapped registers, while hardware handshaking is accomplished using a dedicated handshaking interface.

Notes

1. Throughout the remainder of this chapter, references to both source and destination hardware handshaking interfaces assume an active-high interface (refer to **CFG.SRC(DST)_HS_POL** bits in the Channel Configuration register, **CFG**). When active-low handshaking interfaces are used, then the active level and edge are reversed from that of an active-high interface.
2. Source and destination peripherals can independently select the handshaking interface type; that is, hardware or software handshaking. For more information, refer to the **CFG.HS_SEL_SRC** and **CFG.HS_SEL_DST** parameters in the **CFG** register.

5.2.4.1 Hardware Handshaking

Before the transfer can begin the GPDMA and DLR units must be set up according to the user requirements (shown as step 1 in [Figure 5-4](#)).

Once the peripheral (source or destination) is ready for a transaction it sends a service request. This request is taken by the DLR which in turn forwards it to the GPDMA (step 2). The GPDMA finally executes the transaction (step 3).

Steps 2 and 3 repeat until the programmed transfer is complete.

Note: Optionally interrupts can be generated after block or transaction completion as described in [Section 5.5](#)

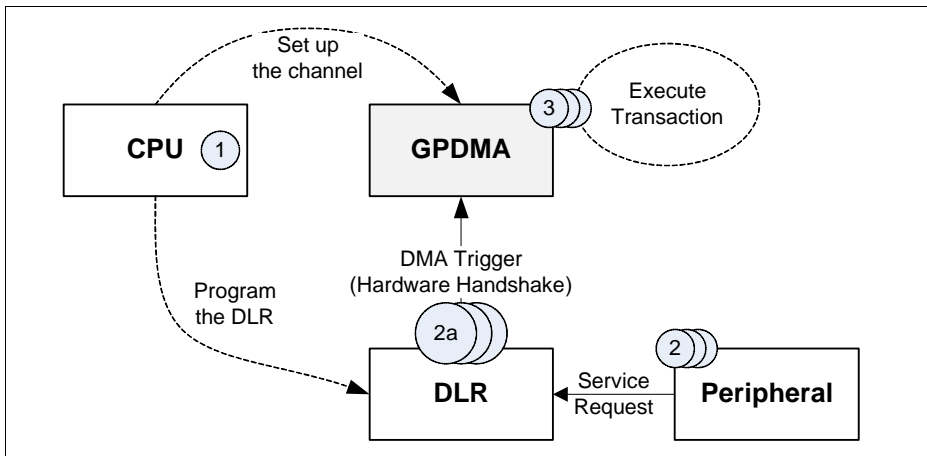


Figure 5-4 Hardware Handshaking Interface

5.2.4.2 Software Handshaking

Before the transfer can begin the GPDMA and NVIC units must be set up according to the user requirements (shown as step 1 in [Figure 5-5](#)).

Once the peripheral (source or destination) is ready for a transaction it sends a service request to the CPU. The interrupt service routine then uses the software registers, detailed in [Section 5.8.4](#), to initiate and control a DMA transaction (step 2). The GPDMA finally executes the transaction (step 3).

Steps 2 and 3 repeat until the programmed transfer is complete.

Note: Optionally interrupts can be generated after block or transaction completion as described in [Section 5.5](#)

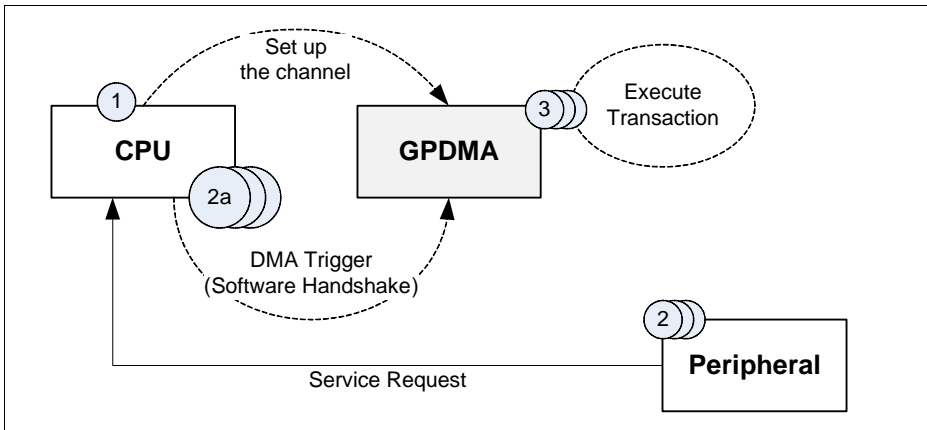


Figure 5-5 Software Handshaking Interface

5.2.4.3 Handshaking with GPDMA as Flow Controller

The GPDMA tries to efficiently transfer the data using as little of the bus bandwidth as possible. Generally, the GPDMA tries to transfer the data using burst transactions and, where possible, fill or empty the channel FIFO in single bursts - provided that the software has not limited the burst length.

The GPDMA can also lock the arbitration for the master bus interface so that a channel is permanently granted the master bus interface. Additionally, the GPDMA can assert the lock signal to lock the system arbiter. For more information, refer to [Section 5.2.6](#).

Before describing the handshaking interface operation, the following sections define the terms "Single Transaction Region" and "Early-Terminated Burst Transaction".

Single Transaction Region

The single transaction region is the time interval where the GPDMA can no longer use full burst transactions to complete the block transfer.

There are cases where a DMA block transfer cannot be completed using only burst transactions. Typically this occurs when the block size is not a multiple of the burst transaction length. In these cases, the block transfer uses burst transactions up to the point where the amount of data left to complete the block is less than the amount of data in a burst transaction. At this point, the GPDMA completes the block transfer using single or early-terminated burst transactions.

Early-Terminated Burst Transaction

When a source or destination peripheral is in the Single Transaction Region, a burst transaction can still be requested when using Software Handshaking. In this case, the burst transaction is started and "early-terminated" at block completion without transferring the programmed amount of data, that is, `src_burst_size_bytes` or `dst_burst_size_bytes`, but only the amount required to complete the block transfer.

Hardware Handshaking

Works as described above in [Chapter 5.2.4.1](#).

Software Handshaking

When the GPDMA is the flow controller, then the last transaction registers - **LSTSRCREG** and **LSTDSTREG** - are not used, and the values in these registers are ignored.

- Operation - Peripheral Not In Single Transaction Region

Writing a 1 to the **REQSRCREG[x]**, **REQDSTREG[x]** bit fields is always interpreted as a burst transaction request, where x is the channel number. However, in order for a burst transaction request to start, software must write a 1 to the **SGLREQSRCREG[x]**, **SGLREQDSTREG[x]** register.

You can write a 1 to the **SGLREQSRCREG[x]**, **SGLREQDSTREG[x]** and **REQSRCREG[x]**, **REQDSTREG[x]** registers in any order, but both registers must be asserted in order to initiate a burst transaction. Upon completion of the burst transaction, the hardware clears the **SGLREQSRCREG[x]**, **SGLREQDSTREG[x]** and **REQSRCREG[x]**, **REQDSTREG[x]** registers.

- Operation - Peripheral In Single Transaction Region

Writing a 1 to the **SGLREQSRCREG**, **SGLREQDSTREG** initiates a single transaction. Upon completion of the single transaction, both the **SGLREQSRCREG**, **SGLREQDSTREG** and **REQSRCREG**, **REQDSTREG** bits are cleared by hardware. Therefore, writing a 1 to the **REQSRCREG**, **REQDSTREG** is ignored while a single transaction has been initiated, and the requested burst transaction is not serviced.

Again, writing a 1 to the **REQSRCREG**, **REQDSTREG** register is always a burst transaction request. However, in order for a burst transaction request to start, the corresponding channel bit in the **SGLREQSRCREG**, **SGLREQDSTREG** must be asserted. Therefore, to ensure that a burst transaction is serviced in this region, you must write a 1 to the **REQSRCREG**, **REQDSTREG** before writing a 1 to the **SGLREQSRCREG**, **SGLREQDSTREG** register. If the programming order is reversed, a single transaction is started instead of a burst transaction. The hardware clears both the **REQSRCREG**, **REQDSTREG** and the **SGLREQSRCREG**, **SGLREQDSTREG** registers after the burst transaction request completes. When a burst transaction is initiated in the

General Purpose DMA (GPDMA)

Single Transaction Region, then the block completes using an Early-Terminated Burst Transaction.

Software can poll the relevant channel bit in the SGLREQSRCREG, SGLREQDSTREG and REQSRCREG, REQDSTREG registers. When both are 0, then either the requested burst or single transaction has completed. Alternatively, the IntSrcTran or IntDstTran interrupts can be enabled and unmasked in order to generate an interrupt when the requested source or destination transaction has completed.

Note: The transaction-complete interrupts are triggered when both single and burst transactions are complete. The same transaction-complete interrupt is used for both single and burst transactions.

5.2.4.4 Handshaking with Peripheral as Flow Controller

When the peripheral is the flow controller, it controls the length of the block and must communicate to the GPDMA when the block transfer is complete. The peripheral does this by telling the GPDMA that the current transaction - burst or single - is the last transaction in the block. When the peripheral is the flow controller and the block size is not a multiple of the **CTL.SRC_MSIZ**, **CTL.DEST_MSIZ**, then the peripheral must use single transactions to complete a block transfer.

Note: Since the peripheral can terminate the block on a single transaction, there is no notion of a Single Transaction Region such as there is when the GPDMA is the flow controller.

When the peripheral is the flow controller, it indicates to the GPDMA which type of transaction - single or burst - to perform by using **Software Handshaking**. Where possible, the GPDMA uses the maximum possible burst length. It can also lock the arbitration for the master bus so that a channel is permanently granted the master bus interface. The GPDMA can also assert the hlock signal to lock the system arbiter. For more information, refer to **Section 5.2.6**.

Hardware Handshaking

This mode is not supported in the XMC4300.

Software Handshaking

Writing a 1 to the Source/Destination Software Transaction Request initiates a transaction (refer to **REQSRCREG** and **REQDSTREG**, respectively). The type of transaction - single or burst - depends on the state of the corresponding channel bit in the Single Source/Destination Transaction Request register (refer to **SGLREQSRCREG** or **SGLREQDSTREG**, respectively)

If **SGLREQSRCREG[n]**, **SGLREQDSTREG[n]** = 1 when a 1 is written to the **REQSRCREG[n]**, **REQDSTREG[n]** register, this means that software is requesting a single transaction on channel n, or a burst transaction otherwise.

General Purpose DMA (GPDMA)

The request is the last in the block if the corresponding channel bit in the Last Source/Destination Request register is asserted; refer to **LSTSRCREG** and **LSTDSTREG**, respectively.

If **LSTSRCREG[n]**, **LSTDSTREG[n]** = 1 when a 1 is written to the **REQSRCREG[n]**, **REQDSTREG[n]** register, this means that software is requesting that this transaction is the last transaction in the block. The **SGLREQSRCREG**, **SGLREQDSTREG** and **LSTSRCREG**, **LSTDSTREG** registers must be written to before the **REQSRCREG**, **REQDSTREG** registers.

On completion of the transaction - single or burst - the relevant channel bit in the **REQSRCREG**, **REQDSTREG** register is cleared by hardware. Software can therefore poll this bit in order to determine when the requested transaction has completed. Alternatively, the **IntSrcTran** or **IntDstTran** interrupts can be enabled and unmasked in order to generate an interrupt when the requested transaction - single or burst - has completed.

When the peripheral is the flow controller and the block size is not a multiple of the **CTL.SRC_MSIZ**, **CTL.DEST_MSIZ**, then software must use single transactions to complete the block transfer.

5.2.5 FIFO Usage

Each channel has a source state machine and destination state machine running in parallel. These state machines generate the request inputs to the arbiter, which arbitrates for the master bus interface (one arbiter per master bus interface).

When the source/destination state machine is granted control of the master bus interface, then AHB transfers between the peripheral and the GPDMA (on behalf of the granted state machine) can take place.

AHB transfers from the source peripheral or to the destination peripheral cannot proceed until the channel FIFO is ready. For burst transaction requests and for transfers involving memory peripherals, the criterion for "FIFO readiness" is controlled by the **FIFO_MODE** field of the **CFG** register.

The definition of FIFO readiness is the same for:

- Single transactions
- Burst transactions, where **CFG.FIFO_MODE** = 0
- Transfers involving memory peripherals, where **CFG.FIFO_MODE** = 0

The channel FIFO is deemed ready when the space/data available is sufficient to complete a single AHB transfer of the specified transfer width. FIFO readiness for source transfers occurs when the channel FIFO contains enough room to accept at least a single transfer of **CTL.SRC_TR_WIDTH** width. FIFO readiness for destination transfers occurs when the channel FIFO contains data to form at least a single transfer of **CTL.DST_TR_WIDTH** width.

General Purpose DMA (GPDMA)

*Note: An exception to FIFO readiness for destination transfers occurs in "FIFO flush mode". In this mode, FIFO readiness for destination transfers occurs when the channel FIFO contains data to form at least a single transfer of **CTL.SRC_TR_WIDTH** width (and not **CTL.DST_TR_WIDTH** width, as is the normal case).*

When **CFG.FIFO_MODE** = 1, then the criteria for FIFO readiness for burst transaction requests and transfers involving memory peripherals are as follows:

- A FIFO is ready for a source burst transfer when the FIFO is less than half empty.
- A FIFO is ready for a destination burst transfer when the FIFO is greater than or equal to half full.

Exceptions to this "readiness" occur. During these exceptions, a value of **CTL.FIFO_MODE** = 0 is assumed. The following are the exceptions:

- Near the end of a burst transaction or block transfer - The channel source state machine does not wait for the channel FIFO to be less than half empty if the number of source data items left to complete the source burst transaction or source block transfer is less than **FIFO DEPTH/2**. Similarly, the channel destination state machine does not wait for the channel FIFO to be greater than or equal to half full, if the number of destination data items left to complete the destination burst transaction or destination block transfer is less than **FIFO DEPTH/2**.
- In FIFO flush mode
- When a channel is suspended - The destination state machine does not wait for the FIFO to become half empty to flush the FIFO, regardless of the value of the **FIFO_MODE** field.

When the source/destination peripheral is not memory, the source/destination state machine waits for a single/burst transaction request. Upon receipt of a transaction request and only if the channel FIFO is "ready" for source/destination AHB transfers, a request for the master bus interface is made by the source/destination state machine.

*Note: There is one exception to this, which occurs when the destination peripheral is the flow controller and **CFG.FCMODE** = 1 (data pre-fetching is disabled). Then the source state machine does not generate a request for the master bus interface (even if the FIFO is "ready" for source transfers and has received a source transaction request) until the destination requests new data.*

When the source/destination peripheral is memory, the source/destination state machine must wait until the channel FIFO is "ready". A request is then made for the master bus interface. There is no handshaking mechanism employed between a memory peripheral and the GPDMA.

5.2.6 Bus and Channel Locking

It is possible to program the GPDMA for:

- Bus locking

General Purpose DMA (GPDMA)

- Channel locking - Locks the arbitration for the AHB master interface, which grants ownership of the master bus interface to one of the requesting channel state machines (source or destination).

Bus and channel locking can proceed for the duration of a DMA transfer, a block transfer, or a single or burst transaction.

Bus Locking

If the LOCK_B bit in the channel configuration register (**CFG**) is set, then the AHB bus is locked for the duration specified in the LOCK_B_L field.

Channel Locking

If the LOCK_CH field is set, then the arbitration for the master bus interface is exclusively reserved for the source and destination peripherals of that channel for the duration specified in the LOCK_CH_L field.

If bus locking is activated for a certain duration, then it follows that the channel is also automatically locked for that duration. Three cases arise:

- **CFG.LOCK_B = 0** - Programmed values of **CFG.LOCK_CH** and **CFG.LOCK_CH_L** are used.
- **CFG.LOCK_B = 1** and **CFG.LOCK_CH = 0** - DMA transfer proceeds as if **CFG.LOCK_CH = 1** and **CFG.LOCK_CH_L = CFG.LOCK_B_L**. The programmed values of **CFG.LOCK_CH** and **CFG.LOCK_CH_L** are ignored.
- **CFG.LOCK_B = 1** and **CFG.LOCK_CH = 1** - Two cases arise:
 - **CFG.LOCK_B_L <= CFG.LOCK_CH_L** - In this case, the DMA transfer proceeds as if **CFG.LOCK_CH_L = CFG.LOCK_B_L** and the programmed value of **CFG.LOCK_CH_L** is ignored. Thus, if bus locking is enabled over the DMA transfer level, then channel locking is enabled over the DMA transfer level, regardless of the programmed value of **CFG.LOCK_CH_L**.
 - **LOCK_B_L > CFG.LOCK_CH_L** - The programmed value of **CFG.LOCK_CH_L** is used. Thus, if bus locking is enabled over the DMA block transfer level and channel locking is enabled over the DMA transfer level, then channel locking is performed over the DMA transfer level.

Locking Levels

If locking is enabled for a channel, then locking of the AHB master bus interface at a programmed locking transfer level is activated when the channel is first granted the AHB master bus interface at the start of that locking transfer level. It continues until the locking transfer level has completed; that is, if channel 0 has enabled channel level locking at the block transfer level, then this channel locks the master bus interface when it is first granted the master bus interface at the start of the block transfer, and continues to lock the master bus interface until the block transfer has completed.

General Purpose DMA (GPDMA)

Source and destination block transfers occur successively in time, and a new source block cannot commence until the previous destination block has completed.

Block and DMA transfer level locking are both terminated on completion of the block or DMA transfer to the destination.

Transaction-level locking is different due to the fact that source and destination transactions occur independently in time, and the number of source and destination transactions in a DMA block or DMA transfer do not have to match. Transaction-level locking is cleared at the end of a source or destination transaction only if the opposing peripheral is not currently in the middle of a transaction.

If channel-level or bus-level locking is enabled for a channel at the transaction level, and either the source or destination of the channel is a memory device, then the locking is ignored and the channel proceeds as if locking (bus or channel) is disabled.

Note: Since there is no notion of a transaction level for a memory peripheral, then transaction-level locking is not allowed when either source or destination is memory.

5.2.7 Scatter/Gather

Scatter is relevant to a destination transfer. The destination address is incremented or decremented by a programmed amount - the scatter increment - when a scatter boundary is reached. [Figure 5-6](#) shows an example destination scatter transfer. The destination address is incremented or decremented by the value stored in the destination scatter increment (DSRx.DSI) field (refer to [DSR](#)), multiplied by the number of bytes in a single AHB transfer to the destination s (decoded value of [CTL.DST_TR_WIDTH](#))/8 - when a scatter boundary is reached. The number of destination transfers between successive scatter boundaries is programmed into the Destination Scatter Count (DSC) field of the DSR register.

Scatter is enabled by writing a 1 to the [CTL.DST_SCATTER_EN](#) field. The [CTL.DINC](#) field determines if the address is incremented, decremented, or remains fixed when a scatter boundary is reached. If the [CTL.DINC](#) field indicates a fixed-address control throughout a DMA transfer, then the [CTL.DST_SCATTER_EN](#) field is ignored, and the scatter feature is automatically disabled.

Gather is relevant to a source transfer. The source address is incremented or decremented by a programmed amount when a gather boundary is reached. The number of source transfers between successive gather boundaries is programmed into the Source Gather Count (SGRx.SGC) field. The source address is incremented or decremented by the value stored in the source gather increment (SGRx.SGI) field (refer to [SGR](#)), multiplied by the number of bytes in a single AHB transfer from the source - (decoded value of [CTL.SRC_TR_WIDTH](#))/8 - when a gather boundary is reached.

Gather is enabled by writing a 1 to the [CTL.SRC_GATHER_EN](#) field. The [CTL.SINC](#) field determines if the address is incremented, decremented, or remains fixed when a

General Purpose DMA (GPDMA)

gather boundary is reached. If the **CTL.SINC** field indicates a fixed-address control throughout a DMA transfer, then the **CTL.SRC_GATHER_EN** field is ignored, and the gather feature is automatically disabled.

Note: For multi-block transfers, the counters that keep track of the number of transfers left to reach a gather/scatter boundary are re-initialized to the source gather count (SGRx.SGC) and destination scatter count (DSC), respectively, at the start of each block transfer.

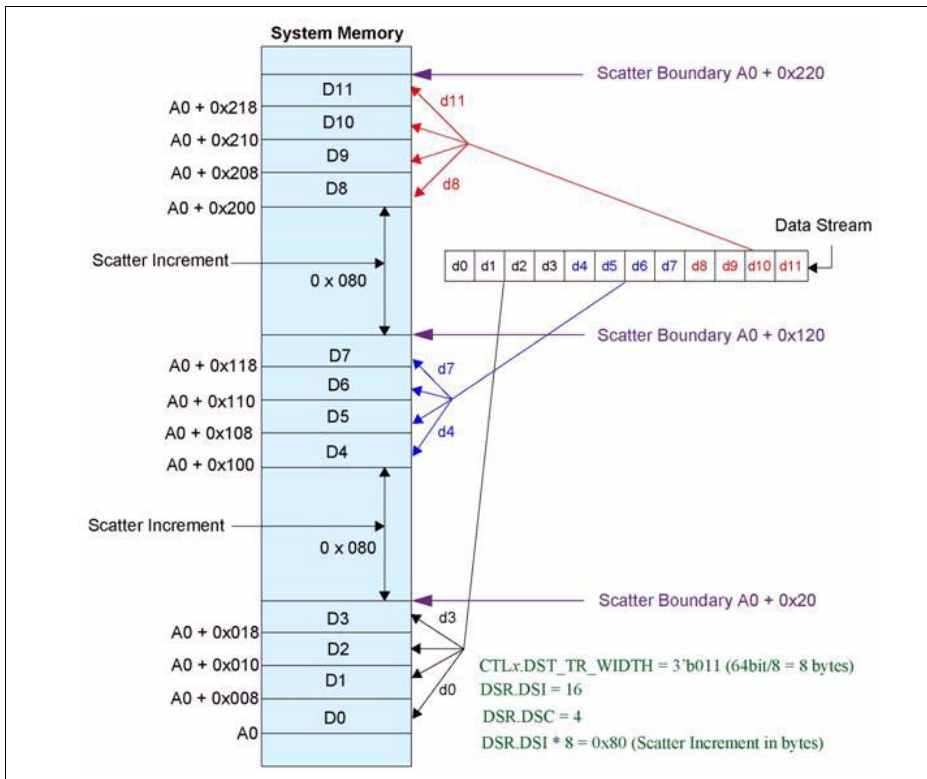


Figure 5-6 Example of Destination Scatter Transfer

As an example of gather increment, consider the following:

```
SRC_TR_WIDTH = 3'b010 (32 bits)
SGR.SGC = 0x04 (source gather count)
CTL.SRC_GATHER_EN = 1 (source gather enabled)
SAR = A0 (starting source address)
```

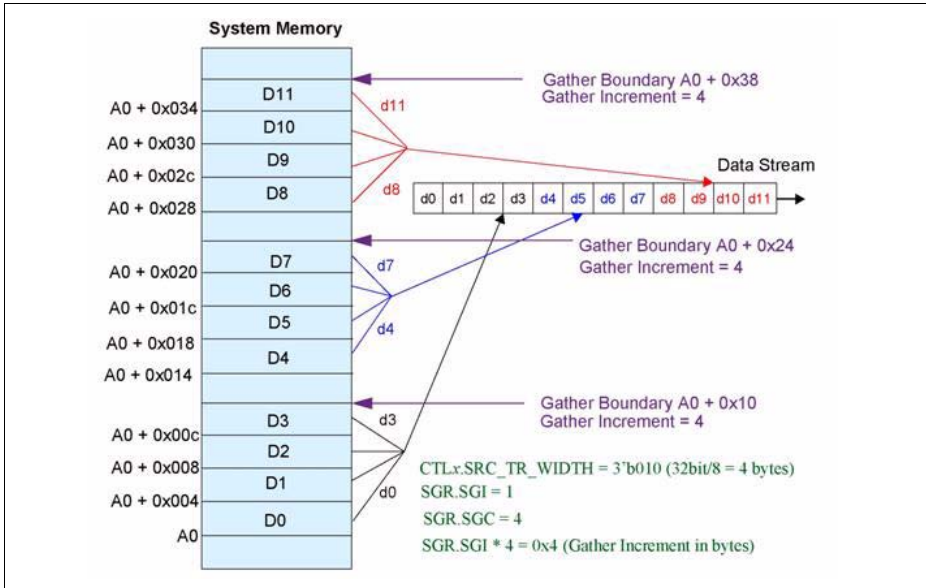


Figure 5-7 Source Gather when SGR.SGI = 0x1

In general, if the starting address is A0 and **CTL.SINC** = 00_B (increment source address control), then the transfer will be:

$$A0, A0 + TWB, A0 + 2*TWB \quad (A0 + (SGR.SGC-1)*TWB)$$

$$\leftarrow \text{scatter_increment} \rightarrow (A0 + (SGR.SGC*TWB) + (SGR.SGI * TWB))$$

where TWB is the transfer width in bytes, decoded value of **CTL.SRC_TR_WIDTH**/8 = src_single_size_bytes.

5.2.8 Abnormal Transfer Termination

A GPDMA DMA transfer may be terminated abruptly by software by clearing the channel enable bit, **CHENREG.CH_EN** or by clearing the global enable bit in the GPDMA Configuration Register (**DMACFGREG[0]**).

If a transfer is in progress while a channel is disabled, abnormal transfer termination and data corruption occurs. Also the transfer acknowledge may be lost. Therefore this must be avoided.

Attention: *Disabling a channel via software prior to completing a transfer is not supported.*

5.3 Basic Transfers

From a users perspective DMA transfers can be grouped into

- software triggered transfers and
- hardware triggered transfers.

The setup procedure for both kinds of transfers is identical to a large extent and is described in more detail later in this section after highlighting the differences of the trigger types.

Software triggered transfers

are set up as memory-to-memory types and start automatically when the channel is enabled. After transfer completion the channel is disabled. There is no way to trigger the transactions by on-chip hardware.

Hardware triggered transfers

are set up as peripheral-to-memory, memory-to-peripheral or peripheral-to-peripheral types. Additionally the trigger source, signal routing and trigger generation must be programmed. Details on trigger generation are found in the “Service Request Generation” section of each peripherals chapter. Signal routing options (ERU) and trigger generation (DLR) are described in the “Service Request Processing” chapter.

GPDMA set up

Transfers are set up by programming fields of the **CTL** and **CFG** registers for that channel. As shown in **Figure 5-2**, a single block is made up of numerous transactions - single and burst - which are in turn composed of AHB transfers.

*Note: There are references to software parameters throughout this chapter. The software parameters are the field names in each register description table and are prefixed by the register name; for example, the Block Transfer Size field in the Control Register is designated as "**CTL.BLOCK_TS**."*

Table 5-3 lists the parameters that are investigated in the following examples. The effects of these parameters on the flow of the block transfer are highlighted.

Table 5-3 Parameters Used in Transfer Examples

Parameter	Description
CTL.TT_FC	Transfer type and flow control
CTL.BLOCK_TS	Block transfer size
CTL.SRC_TR_WIDTH	Source transfer width
CTL.DST_TR_WIDTH	Destination transfer width

General Purpose DMA (GPDMA)

Table 5-3 Parameters Used in Transfer Examples (cont'd)

Parameter	Description
CTL.SRC_MSIZ E	Source burst transaction length
CTL.DEST_MSIZ E	Destination burst transaction length
CFG.MAX_ABRST	Maximum AMBA burst length
CFG.FIFO_MODE	FIFO mode select
CFG.FCMODE	Flow-control mode

The GPDMA is programmed with the number of data items that are to be transferred for each burst transaction request, **CTL.SRC_MSIZ**E and **CTL.DEST_MSIZ**E. Similarly, the width of each data item in the transaction is set by the **CTL.SRC_TR_WIDTH** and **CTL.DST_TR_WIDTH** fields.

5.3.1 Block transfer with GPDMA as the flow controller

Table 5-4 lists the DMA parameters for this example (the FIFO depth is taken as 16 bytes).

Table 5-4 Parameters in Transfer Operation

Parameter	Description
CTL.TT_FC = 011 _B	Peripheral-to-peripheral transfer with GPDMA as flow controller
CTL.BLOCK_TS = 12	-
CTL.SRC_TR_WIDTH = 010 _B	32 bits
CTL.DST_TR_WIDTH = 010 _B	32 bits
CTL.SRC_MSIZ E = 001 _B	Source burst transaction length = 4
CTL.DEST_MSIZ E = 001 _B	Destination burst transaction length = 4
CFG.MAX_ABRST = 0 _B	No limit on maximum AMBA burst length

A total of 48 bytes are transferred in the block (that is `blk_size_bytes_dma = 48`). As shown in **Figure 5-8**, this block transfer consists of three bursts of length 4 from the source, interleaved with three bursts, again of length 4, to the destination.

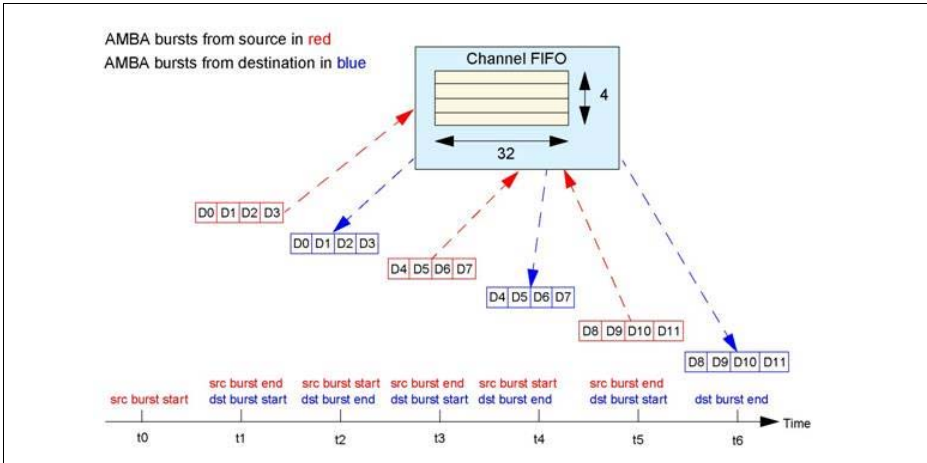


Figure 5-8 Breakdown of Block Transfer

The channel FIFO is alternatively filled by a burst from the source and emptied by a burst to the destination until the block transfer has completed, as shown in [Figure 5-9](#).

D3	Empty	D7	Empty	D11	Empty
D2	Empty	D6	Empty	D10	Empty
D1	Empty	D5	Empty	D9	Empty
D0	Empty	D4	Empty	D8	Empty
Time t1	Time t2	Time t3	Time t4	Time t5	Time t6

Figure 5-9 Channel FIFO Contents

Burst transactions are completed in one burst. Additionally neither the source or destination peripherals enter their Single Transaction Region at any stage throughout the DMA transfer, and the block transfer from the source and to the destination consists of burst transactions only.

5.3.2 Effect of maximum AMBA burst length on a block transfer

If the `CFG.MAX_ABRST = 2` parameter and all other parameters are left unchanged from previous example, then the block transfer would look like that shown in [Figure 5-10](#).

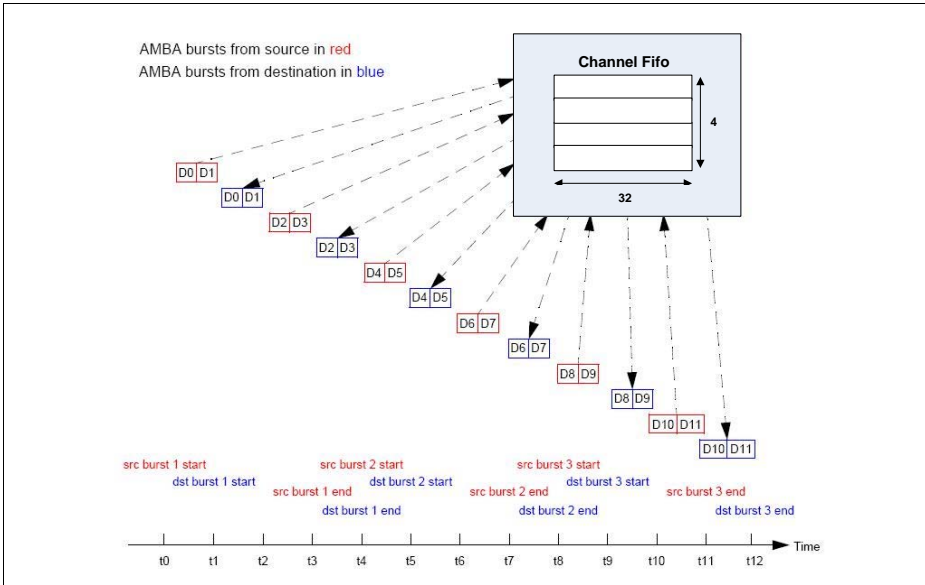


Figure 5-10 Breakdown of Block Transfer where max_abrst = 2, Case 1

The channel FIFO is alternatively half filled by a burst from the source, and then emptied by a burst to the destination until the block transfer has completed; this is illustrated in [Figure 5-11](#).

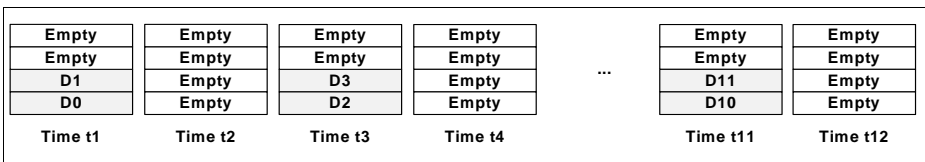


Figure 5-11 Channel FIFO Contents

In this example block transfer, each source or destination burst transaction is made up of two bursts, each of length 2. As [Figure 5-11](#) illustrates, the top two channel FIFO locations are redundant for this block transfer. However, this is not the general case. The block transfer could proceed as indicated in [Figure 5-12](#).

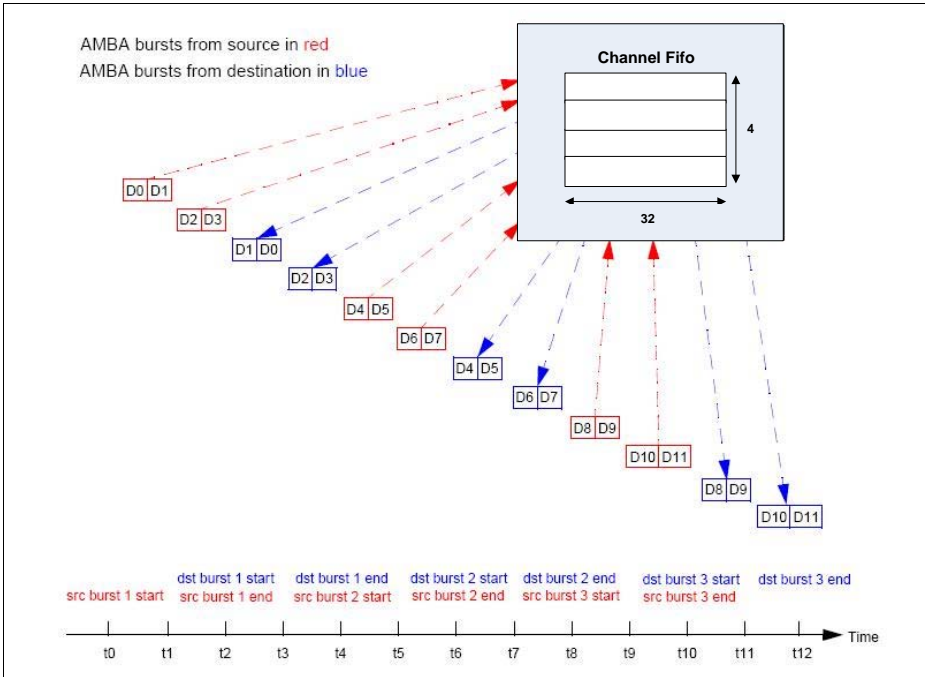


Figure 5-12 Breakdown of Block Transfer where max_abrst = 2, Case 2

This depends on the timing of the source and destination transaction requests, relative to each other. **Figure 5-13** illustrates the channel FIFO status for **Figure 5-12**.

D3	Empty	D7	Empty	D11	Empty
D2	Empty	D6	Empty	D10	Empty
D1	Empty	D5	Empty	D9	Empty
D0	Empty	D4	Empty	D8	Empty
Time t2	Time t4	Time t6	Time t8	Time t10	Time t12

Figure 5-13 Channel FIFO Contents

Recommendation

To allow a burst transaction to complete in a single burst, the following should be true:

$$CFGL.MAX_ABRST \geq \max(\text{src_burst_size_bytes}, \text{dst_burst_size_bytes})$$

Adhering to the above recommendation results in a reduced number of bursts per block, which in turn results in improved bus utilization and lower latency for block transfers.

General Purpose DMA (GPDMA)

Limiting a burst to a maximum length prevents the GPDMA from saturating the AHB bus when the system arbiter is configured to only allow changing of the grant signals to bus masters at the end of an undefined length burst. It also prevents a channel from saturating a GPDMA master bus interface.

5.4 Multi Block Transfers

A DMA transfer may consist of

- single block transfer, supported by all channels.
- multi-block transfers, supported by channels 0 and 1 of GPDMA0.

On successive blocks of a multi-block transfer, the **SAR**, **DAR** register in the GPDMA is reprogrammed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading
- Contiguous address between blocks

On successive blocks of a multi-block transfer, the **CTL** register in the GPDMA is reprogrammed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading

When block chaining, using Linked Lists is the multi-block method of choice. On successive blocks, the **LLP** register in the GPDMA is reprogrammed using block chaining with linked lists.

A block descriptor consists of six registers: **SAR**, **DAR**, **LLP**, **CTL**, **SSTAT** and **DSTAT**. The first four registers, along with the **CFG** register, are used by the GPDMA to set up and describe the block transfer.

Note: The term Link List Item (LLI) and block descriptor are synonymous.

5.4.1 Block Chaining Using Linked Lists

In this case, the GPDMA reprograms the channel registers prior to the start of each block by fetching the block descriptor for that block from system memory. This is known as an LLI update.

GPDMA block chaining uses a Linked List Pointer register (**LLP**) that stores the address in memory of the next linked list item. Each LLI contains the corresponding block descriptors:

1. **SAR**
2. **DAR**
3. **LLP**
4. **CTL**
5. **SSTAT**
6. **DSTAT**

To set up block chaining, you program a sequence of Linked Lists in memory.

LLI accesses are always 32-bit accesses aligned to 32-bit boundaries and cannot be changed or programmed to anything other than 32-bit, even if the AHB master interface of the LLI supports more than a 32-bit data width.

General Purpose DMA (GPDMA)

The **SAR**, **DAR**, **LLP**, and **CTL** registers are fetched from system memory on an LLI update. The updated contents of the **CTL**, **SSTAT**, and **DSTAT** registers are optionally written back to memory on block completion. **Figure 5-14** and **Figure 5-15** show how you use chained linked lists in memory to define multi-block transfers using block chaining.

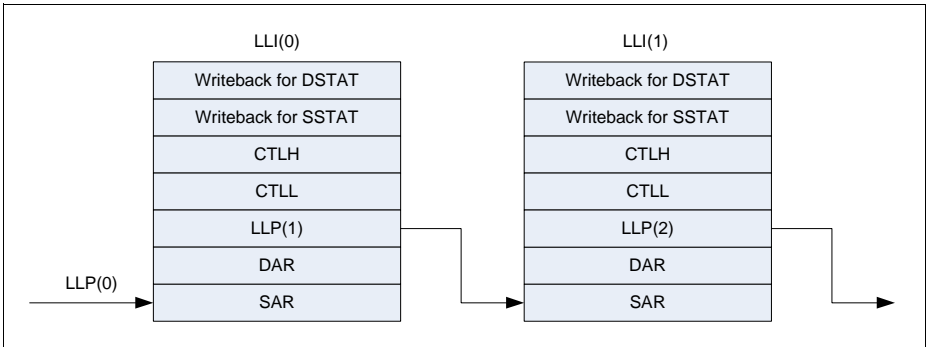


Figure 5-14 Multi-Block Transfer Using Linked Lists When `CFG.SS_UPD_EN` is set to '1'

It is assumed that no allocation is made in system memory for the source status when the parameter `CFG.SS_UPD_EN` is set to '0'. In this case, then the order of a Linked List item is as follows:

1. **SAR**
2. **DAR**
3. **LLP**
4. **CTL**
5. **DSTAT**

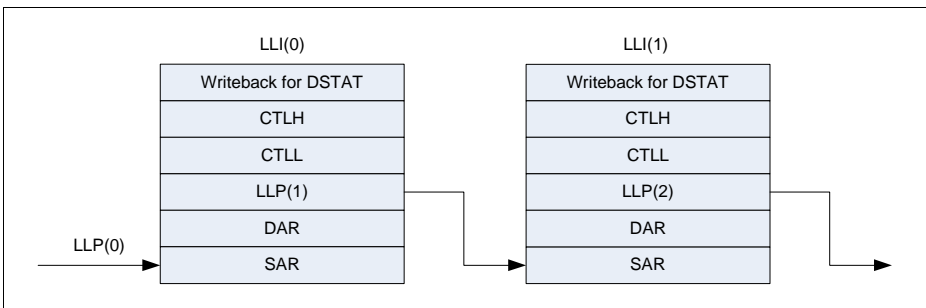


Figure 5-15 Multi-Block Transfer Using Linked Lists When `CFG.SS_UPD_EN` is set to '0'

General Purpose DMA (GPDMA)

Note: In order to not confuse the **SAR**, **DAR**, **LLP**, **CTL**, **SSTAT** and **DSTAT** register locations of the LLI with the corresponding GPDMA memory mapped register locations, the LLI register locations are prefixed with LLI; that is, LLI.SAR, LLI.DAR, LLI.LLP, LLI.CTLH/L, LLI.SSTATx, and LLI.DSTATx.

Figure 5-14 and Figure 5-15 show the mapping of a Linked List Item stored in memory to the channel registers block descriptor.

Rows 6 through 10 of Table 5-5 show the required values of **LLP**, **CTL**, and **CFG** for multi-block DMA transfers using block chaining.

Note: For rows 6 through 10 of Table 5-5, the LLI.CTLH/L, LLI.LLP, LLI.SAR, and LLI.DAR register locations of the LLI are always affected at the start of every block transfer. The LLI.LLP and LLI.CTLH/L locations are always used to reprogram the GPDMA **LLP** and **CTL** registers. However, depending on the Table 5-5 row number, the LLI.SAR, LLI.DAR address may or may not be used to reprogram the GPDMA **SAR**, **DAR** registers.

Table 5-5 Programming of Transfer Types and Channel Register Update Method

Transfer Type	LLP.LOC = 0	CTL.LLP_SRCEN	CFG.RELOAD_SRC	CTL.LLPDST_EN	CFG.RELOAD_DST	CTL, LLP UpdateMethod	SAR UpdateMethod	DAR UpdateMethod	Write Back
1. Single-block or last transfer of multi-block.	Yes	0	0	0	0	None, user reprograms	None (single)	None (single)	No
2. Auto-reload multi-block transfer with contiguous SAR	Yes	0	0	0	1	CTL , LLP are reloaded from initial values.	Contiguous	Auto-Reload	No
3. Auto-reload multi-block transfer with contiguous DAR .	Yes	0	1	0	0	CTL , LLP are reloaded from initial values	Auto-Reload	Contiguous	No
4. Auto-reload multi-block transfer	Yes	0	1	0	1	CTL , LLP are reloaded from initial values	Auto-reload	Auto-Reload	No

Table 5-5 Programming of Transfer Types and Channel Register Update Method (cont'd)

Transfer Type	LLP.LOC = 0	CTL.LLP_SRCEN	CFG.RELOAD_SRC	CTL.LLPDST_EN	CFG.RELOAD_DST	CTL, LLP UpdateMethod	SAR UpdateMethod	DAR UpdateMethod	Write Back
5. Single-block or last transfer of multi-block.	No	0	0	0	0	None, user reprograms	None (single)	None (single)	Yes
6. Linked list multi-block transfer with contiguous SAR	No	0	0	1	0	CTL, LLP loaded from next Linked List item.	Contiguous	Linked List	Yes
7. Linked list multi-block transfer with auto-reload SAR	No	0	1	1	0	CTL, LLP loaded from next Linked List item.	Auto-Reload	Linked List	Yes
8. Linked list multi-block transfer with contiguous DAR	No	1	0	0	0	CTL, LLP loaded from next Linked List item.	Linked List	Contiguous	Yes
9. Linked list multi-block transfer with auto-reload DAR	No	1	0	0	1	CTL, LLP loaded from next Linked List item.	Linked List	Auto-Reload	Yes
10. Linked list multi-block transfer	No	1	0	1	0	CTL, LLP loaded from next Linked List item.	Linked List	Linked List	Yes

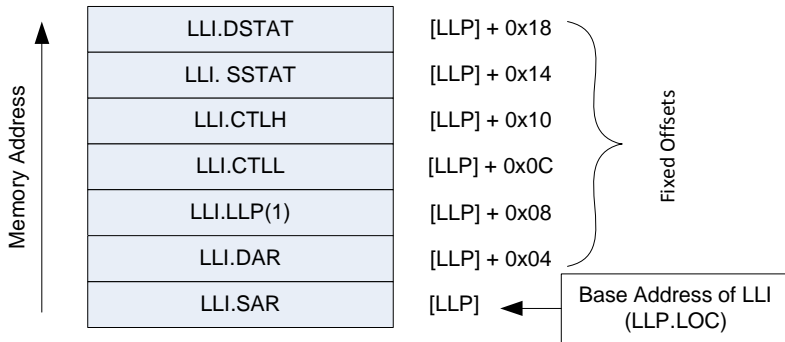


Figure 5-16 Mapping of Block Descriptor (LLI) in Memory to Channel Registers When CFG.SS_UPD_EN = 1

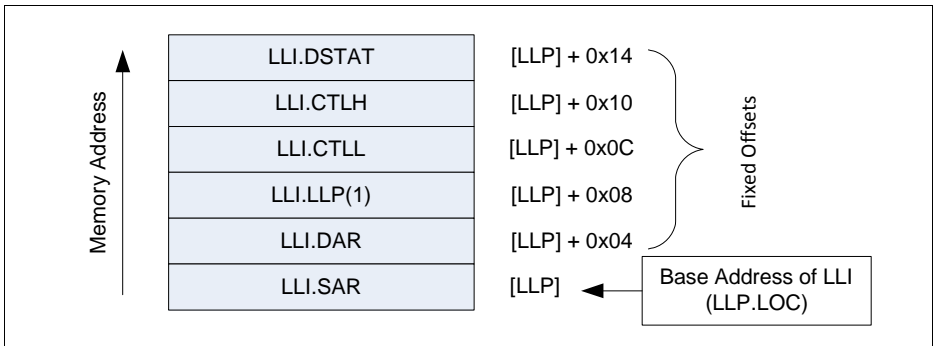


Figure 5-17 Mapping of Block Descriptor (LLI) in Memory to Channel Registers When CFG.SS_UPD_EN = 0

Notes

1. Throughout this chapter, there are descriptions about fetching the LLI.CTLH/L register from the location pointed to by the LLP register. This exact location is the LLI base address (stored in LLP register) plus the fixed offset. For example, in **Figure 5-16** the location of the LLI.CTLH/L register is LLP.LOC + 0xc.
2. Referring to **Table 5-5**, if the Write Back column entry is "Yes" and the channel is 0 or 1, then the CTLH register is always written to system memory (to LLI.CTLH) at the end of every block transfer.
3. The source status is fetched and written to system memory at the end of every block transfer if the Write Back column entry is "Yes" and CFG.SS_UPD_EN is enabled.

General Purpose DMA (GPDMA)

4. The destination status is fetched and written to system memory at the end of every block transfer if the Write Back column entry is "Yes" and **CFG.DS_UPD_EN** is enabled.

5.4.2 Auto-Reloading of Channel Registers

During auto-reloading, the channel registers are reloaded with their initial values at the completion of each block and the new values used for the new block. Depending on the row number in **Table 5-5**, some or all of the **SAR**, **DAR**, and **CTL** channel registers are reloaded from their initial value at the start of a block transfer.

5.4.3 Contiguous Address Between Blocks

In this case, the address between successive blocks is selected as a continuation from the end of the previous block.

Enabling the source or destination address to be contiguous between blocks is a function of the **CTL.LLP_SRC_EN**, **CFG.RELOAD_SRC**, **CTL.LLP_DST_EN**, and **CTL.RELOAD_DST** registers (see **Table 5-5**).

*Note: You cannot select both **SAR** and **DAR** updates to be contiguous. If you want this functionality, you should increase the size of the Block Transfer (**CTL.BLOCK_TS**), or if this is at the maximum value, use Row 10 of **Table 5-5** and set up the **LLI.SAR** address of the block descriptor to be equal to the end **SAR** address of the previous block. Similarly, set up the **LLI.DAR** address of the block descriptor to be equal to the end **DAR** address of the previous block.*

5.4.4 Suspension of Transfers Between Blocks

At the end of every block transfer, an end-of-block interrupt is asserted if:

1. Interrupts are enabled, **CTL.INT_EN** = 1, and
2. The channel block interrupt is unmasked, **MASKBLOCK[n]** = 1, where n is the channel number.

Note: The block-complete interrupt is generated at the completion of the block transfer to the destination.

For rows 6, 8, and 10 of **Table 5-5**, the DMA transfer does not stall between block transfers. For example, at the end-of-block N, the GPDMA automatically proceeds to block N + 1.

For rows 2, 3, 4, 7, and 9 of **Table 5-5** (**SAR** and/or **DAR** auto-reloaded between block transfers), the DMA transfer automatically stalls after the end-of-block interrupt is asserted, if the end-of-block interrupt is enabled and unmasked.

The GPDMA does not proceed to the next block transfer until a write to the **CLEARBLOCK[n]** block interrupt clear register, done by software to clear the channel block-complete interrupt, is detected by hardware.

General Purpose DMA (GPDMA)

For rows 2, 3, 4, 7, and 9 of **Table 5-5** (**SAR** and/or **DAR** auto-reloaded between block transfers), the DMA transfer does not stall if either:

- Interrupts are disabled, **CTL.INT_EN** = 0, or
- The channel block interrupt is masked, **MASKBLOCK[n]** = 0, where n is the channel number.

Channel suspension between blocks is used to ensure that the end-of-block ISR (interrupt service routine) of the next-to-last block is serviced before the start of the final block commences. This ensures that the ISR has cleared the **CFG.RELOAD_SRC** and/or **CFG.RELOAD_DST** bits before completion of the final block. The reload bits **CFG.RELOAD_SRC** and/or **CFG.RELOAD_DST** should be cleared in the end-of-block ISR for the next-to-last block transfer.

5.4.5 Ending Multi-Block Transfers

All multi-block transfers must end as shown in either Row 1 or Row 5 of **Table 5-5**. At the end of every block transfer, the GPDMA samples the row number, and if the GPDMA is in the Row 1 or Row 5 state, then the previous block transferred was the last block and the DMA transfer is terminated.

Note: Row 1 and Row 5 are used for single-block transfers or terminating multi-block transfers. Ending in the Row 5 state enables status fetch and write-back for the last block. Ending in the Row 1 state disables status fetch and write-back for the last block.

For rows 2, 3, and 4 of **Table 5-5**, (**LLP.LOC** = 0 and **CFG.RELOAD_SRC** and/or **CFG.RELOAD_DST** is set), multi-block DMA transfers continue until both the **CFG.RELOAD_SRC** and **CFG.RELOAD_DST** registers are cleared by software. They should be programmed to 0 in the end-of-block interrupt service routine that services the next-to-last block transfer; this puts the GPDMA into the Row 1 state.

For rows 6, 8, and 10 of **Table 5-5** (both **CFG.RELOAD_SRC** and **CFG.RELOAD_DST** cleared), the user must set up the last block descriptor in memory so that both **LLI.CTLH/L.LLP_SRC_EN** and **LLI.CTLH/L.LLP_DST_EN** are 0. If the **LLI.LLP** register of the last block descriptor in memory is non-zero, then the DMA transfer is terminated in Row 5. If the **LLI.LLP** register of the last block descriptor in memory is 0, then the DMA transfer is terminated in Row 1.

*Note: The only allowed transitions between the rows of **Table 5-5** are from any row into Row 1 or Row 5. As already stated, a transition into row 1 or row 5 is used to terminate the DMA transfer; all other transitions between rows are not allowed. Software must ensure that illegal transitions between rows do not occur between blocks of a multi-block transfer. For example, if block N is in row 10, then the only allowed rows for block N + 1 are rows 10, 5, or 1.*

5.4.6 Programming Examples

Three registers - **LLP**, **CTL**, and **CFG** - need to be programmed to determine whether single- or multi-block transfers occur, and which type of multi-block transfer is used. The different transfer types are shown in **Table 5-5**.

The GPDMA can be programmed to fetch the status from the source or destination peripheral; this status is stored in the **SSTAT** and **DSTAT** registers. When the GPDMA is programmed to fetch the status from the source or destination peripheral, it writes this status and the contents of the **CTL** register back to memory at the end of a block transfer. The Write Back column of **Table 5-5** shows when this occurs.

The "Update Method" columns indicate where the values of **SAR**, **DAR**, **CTL**, and **LLP** are obtained for the next block transfer when multi-block GPDMA transfers are enabled.

*Note: In **Table 5-5**, all other combinations of **LLP.LOC = 0**, **CTL.LLP_SRC_EN**, **CFG.RELOAD_SRC**, **CTL.LLP_DST_EN**, and **CFG.RELOAD_DST** are illegal, and will cause indeterminate or erroneous behavior.*

Generic Setup of Transfer Type and Characteristics

This generic sequence is referenced by the examples further below in this section.

1. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the TT_FC of the **CTL** register. **Table 5-10** lists the decoding for this field.
2. Set up the transfer characteristics, such as:
 - a) Transfer width for the source in the SRC_TR_WIDTH field. **Table 5-9** lists the decoding for this field.
 - b) Transfer width for the destination in the DST_TR_WIDTH field. **Table 5-9** lists the decoding for this field.
 - c) Incrementing/decrementing or fixed address for the source in the SINC field.
 - d) Incrementing/decrementing or fixed address for the destination in the DINC field.

5.4.6.1 Single-block Transfer

This section is an example for the transfer listed in row 1 in **Table 5-5**.

*Note: Row 5 in **Table 5-5** is also a single-block transfer with write-back of control and status information enabled at the end of the single-block transfer.*

1. Read the Channel Enable register to choose a free (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: **CLEARTRF**, **CLEARBLOCK**, **CLEARSRCTRAN**, **CLEARDSTTRAN**, and **CLEARERR**. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
3. Program the following channel registers:
 - a) Write the starting source address in the **SAR** register for channel x.

General Purpose DMA (GPDMA)

- b) Write the starting destination address in the **DAR** register for channel x.
 - c) Program **CTL** and **CFG** according to Row 1, as shown in **Table 5-5**. Program the **LLP** register with 0.
 - d) Write the control information for the DMA transfer in the **CTL** register for channel x.
 - e) Write the channel configuration information into the **CFG** register for channel x.
 1. Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory. This step requires programming the **CFG.HS_SEL_SRC** or **CFG.HS_SEL_DST** bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests. Writing a 1 activates the software handshaking interface to handle source and destination requests.
 2. If the hardware handshaking interface is activated for the source or destination peripheral, assign a handshaking interface to the source and destination peripheral; this requires programming the **CFG.SRC_PER** and **CFG.DEST_PER** bits, respectively.
 - f) If gather is enabled (**CTL.SRC_GATHER_EN** = 1), program the **SGR** register for channel x.
 - g) If scatter is enabled (**CTL.DST_SCATTER_EN** = 1), program the **DSR** register for channel x.
4. After the GPDMA-selected channel has been programmed, enable the channel by writing a 1 to the **GPDMA0_CHENREG.CH_EN** bit. Ensure that bit 0 of the **GPDMA0_DMACFGREG** register is enabled.
 5. Source and destination request single and burst DMA transactions in order to transfer the block of data (assuming non-memory peripherals). The GPDMA acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
 6. Once the transfer completes, hardware sets the interrupts and disables the channel. At this time, you can respond to either the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register (**RAWTFR[n]**, n = channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, the software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, **CLEARTFR[n]**, before the channel is enabled.

5.4.6.2 Multi-Block Transfer with Source Address Auto-Reloaded and Contiguous Destination Address

This section is an example for the transfer listed in row 3 in **Table 5-5**.

Note: This type of transfer is supported by GPDMA0 channels 0 and 1 only.

1. Read the Channel Enable register (see **GPDMA0_CHENREG**) to choose a free (disabled) channel.

General Purpose DMA (GPDMA)

2. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: **CLEARTRF**, **CLEARBLOCK**, **CLEARSRCTRAN**, **CLEARSTTRAN**, and **CLEARERR**. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
3. Program the following channel registers:
 - a) Write the starting source address in the **SAR** register for channel x.
 - b) Write the starting destination address in the **DAR** register for channel x.
 - c) Program **CTL** and **CFG** according to Row 3, shown in **Table 5-5**. Program the **LLP** register with 0.
 - d) Write the control information for the DMA transfer in the **CTL** register for channel x.
 - e) If gather is enabled (**CTL.SRC_GATHER_EN** = 1), program the **SGR** register for channel x.
 - f) If scatter is enabled (**CTL.DST_SCATTER_EN** = 1), program the **DSR** register for channel x.
 - g) Write the channel configuration information into the **CFG** register for channel x.
 1. Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
This step requires programming the **HS_SEL_SRC**, **HS_SEL_DST** bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.
 2. If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the **SRC_PER** and **DEST_PER** bits, respectively.
4. After the GPDMA channel has been programmed, enable the channel by writing a 1 to the **GPDMA0_CHENREG.CH_EN** bit. Ensure that bit 0 of the **GPDMA0_DMACFGREG** register is enabled.
5. Source and destination request single and burst GPDMA transactions to transfer the block of data (assuming non-memory peripherals). The GPDMA acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
6. When the block transfer has completed, the GPDMA reloads the **SAR** register; the **DAR** register remains unchanged. Hardware sets the block-complete interrupt. The GPDMA then samples the row number, as shown in **Table 5-5**. If the GPDMA is in Row 1, then the DMA transfer has completed. Hardware sets the transfer-complete interrupt and disables the channel. You can either respond to the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register (**RAWTRF[n]**, n = channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, **CLEARTRF[n]**, before the channel is enabled. If the GPDMA is not in Row 1, the next step is performed.

7. The DMA transfer proceeds as follows:
- If interrupts are enabled ($\text{CTL.INT_EN} = 1$) and the block-complete interrupt is unmasked ($\text{MASKBLOCK}[x] = 1_B$, where x is the channel number), hardware sets the block-complete interrupt when the block transfer has completed. It then stalls until the block-complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block-complete ISR (interrupt service routine) should clear the source reload bit, CFG.RELOAD_SRC . This puts the GPDMA into Row 1, as shown in [Table 5-5](#). If the next block is not the last block in the DMA transfer, then the source reload bit should remain enabled to keep the GPDMA in Row 3, as shown in [Table 5-5](#).
 - If interrupts are disabled ($\text{CTL.INT_EN} = 0$) or the block-complete interrupt is masked ($\text{MASKBLOCK}[x] = 0_B$, where x is the channel number), then hardware does not stall until it detects a write to the block-complete interrupt clear register; instead, it starts the next block transfer immediately. In this case, software must clear the source reload bit, CFG.RELOAD_SRC , to put the device into Row 1 of [Table 5-5](#) before the last block of the DMA transfer has completed.

The transfer is similar to that shown in [Figure 5-18](#).

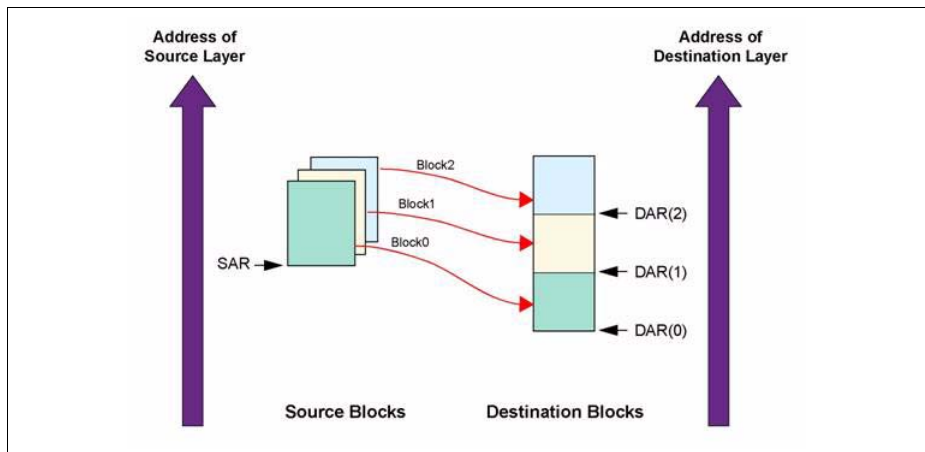


Figure 5-18 Multi-Block DMA Transfer with Source Address Auto-Reloaded and Contiguous Destination Address

The DMA transfer flow is shown in **Figure 5-19**.

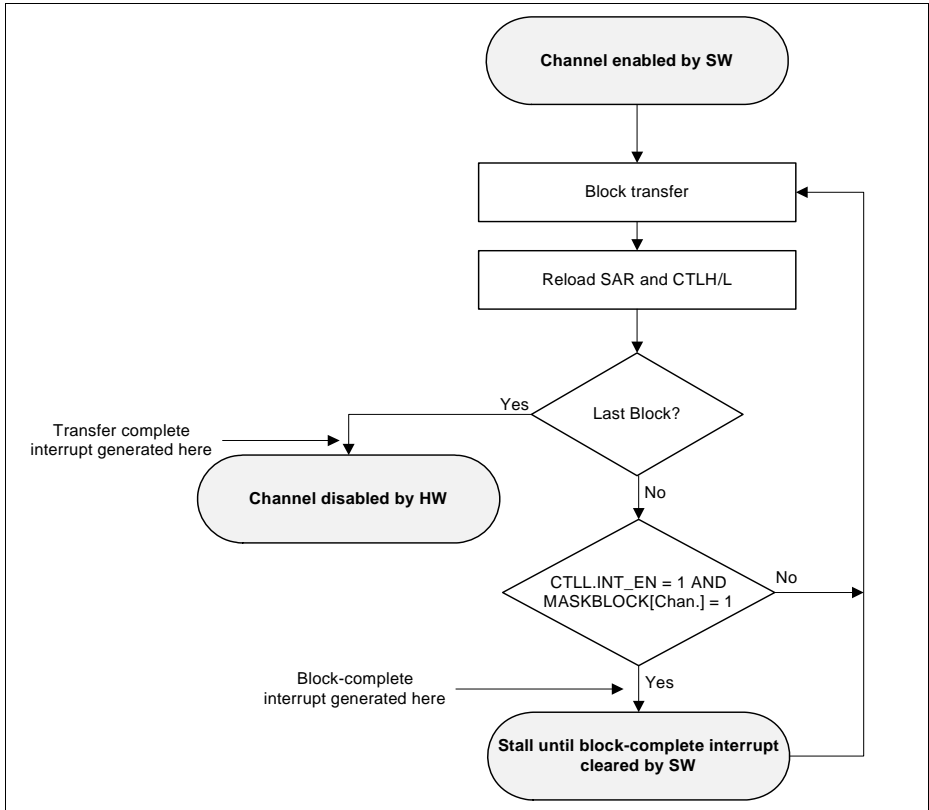


Figure 5-19 DMA Transfer Flow for Source Address Auto-Reloaded and Linked List Destination Address

5.4.6.3 Multi-Block Transfer with Source and Destination Address Auto-Reloaded

This section is an example for the transfer listed in row 4 in [Table 5-5](#).

Note: This type of transfer is supported by GPDMA0 channels 0 and 1 only.

1. Read the Channel Enable register (see [GPDMA0_CHENREG](#)) to choose an available (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: CLEARTRF, CLEARBLOCK, CLEARSRCTRAN, CLEARDSTTRAN, and CLEARERR. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
3. Program the following channel registers:
 - a) Write the starting source address in the [SAR](#) register for channel x.
 - b) Write the starting destination address in the [DAR](#) register for channel x.
 - c) Program [CTL](#) and [CFG](#) according to Row 4, as shown in [Table 5-5](#). Program the [LLP](#) register with 0.
 - d) Write the control information for the DMA transfer in the [CTL](#) register for channel x.
 - e) If gather is enabled ([CTL.SRC_GATHER_EN](#) = 1), program the [SGR](#) register for channel x.
 - f) If scatter is enabled ([CTL.DST_SCATTER_EN](#) = 1), program the [DSR](#) register for channel x.
 - g) Write the channel configuration information into the [CFG](#) register for channel x. Ensure that the reload bits, [CFG.RELOAD_SRC](#) and [CFG.RELOAD_DST](#), are enabled.
 1. Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory. This step requires programming the [HS_SEL_SRC](#), [HS_SEL_DST](#) bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.
 2. If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the [SRC_PER](#) and [DEST_PER](#) bits, respectively.
4. After the GPDMA selected channel has been programmed, enable the channel by writing a 1 to the [GPDMA0_CHENREG.CH_EN](#) bit. Ensure that bit 0 of the [GPDMA0_DMACFGREG](#) register is enabled.
5. Source and destination request single and burst GPDMA transactions to transfer the block of data (assuming non-memory peripherals). The GPDMA acknowledges on completion of each burst/single transaction and carries out the block transfer.
6. When the block transfer has completed, the GPDMA reloads the [SAR](#), [DAR](#), and [CTL](#) registers. Hardware sets the block-complete interrupt. The GPDMA then

General Purpose DMA (GPDMA)

samples the row number, as shown in [Table 5-5](#). If the GPDMA is in Row 1, then the DMA transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. You can either respond to the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register (RAWTFR[n], where n is the channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, CLEARTFR[n], before the channel is enabled. If the GPDMA is not in Row 1, the next step is performed.

7. The DMA transfer proceeds as follows:
 - a) If interrupts are enabled (**CTL.INT_EN** = 1) and the block-complete interrupt is unmasked (**MASKBLOCK[x]** = 1_B, where x is the channel number), hardware sets the block-complete interrupt when the block transfer has completed. It then stalls until the block-complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block-complete ISR (interrupt service routine) should clear the reload bits in the **CFG.RELOAD_SRC** and **CFG.RELOAD_DST** registers. This puts the GPDMA into Row 1, as shown in [Table 5-5](#). If the next block is not the last block in the DMA transfer, then the reload bits should remain enabled to keep the GPDMA in Row 4.
 - b) If interrupts are disabled (**CTL.INT_EN** = 0) or the block-complete interrupt is masked (**MASKBLOCK[x]** = 0_B, where x is the channel number), then hardware does not stall until it detects a write to the block-complete interrupt clear register; instead, it immediately starts the next block transfer. In this case, software must clear the reload bits in the **CFG.RELOAD_SRC** and **CFG.RELOAD_DST** registers to put the GPDMA into Row 1 of [Table 5-5](#) before the last block of the DMA transfer has completed.

The transfer is similar to that shown in [Figure 5-20](#).

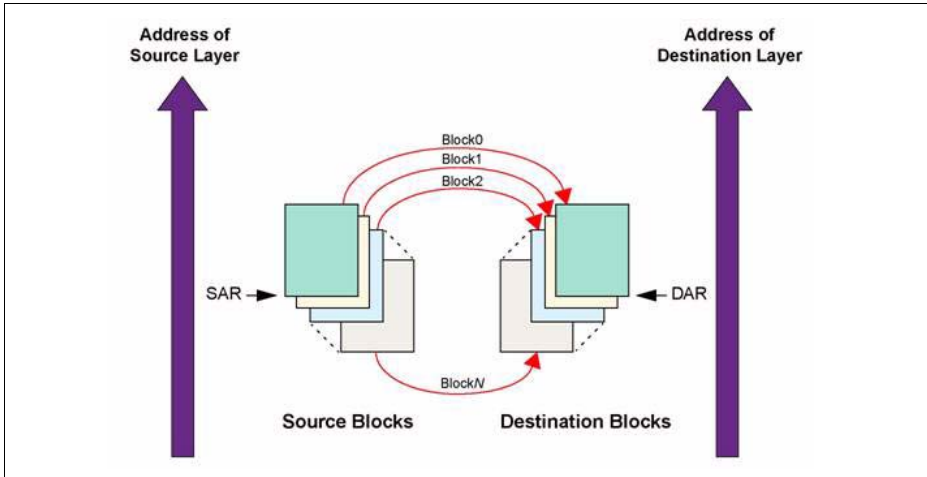


Figure 5-20 Multi-Block DMA Transfer with Source and Destination Address Auto-Reloaded

The DMA transfer flow is shown in **Figure 5-21**.

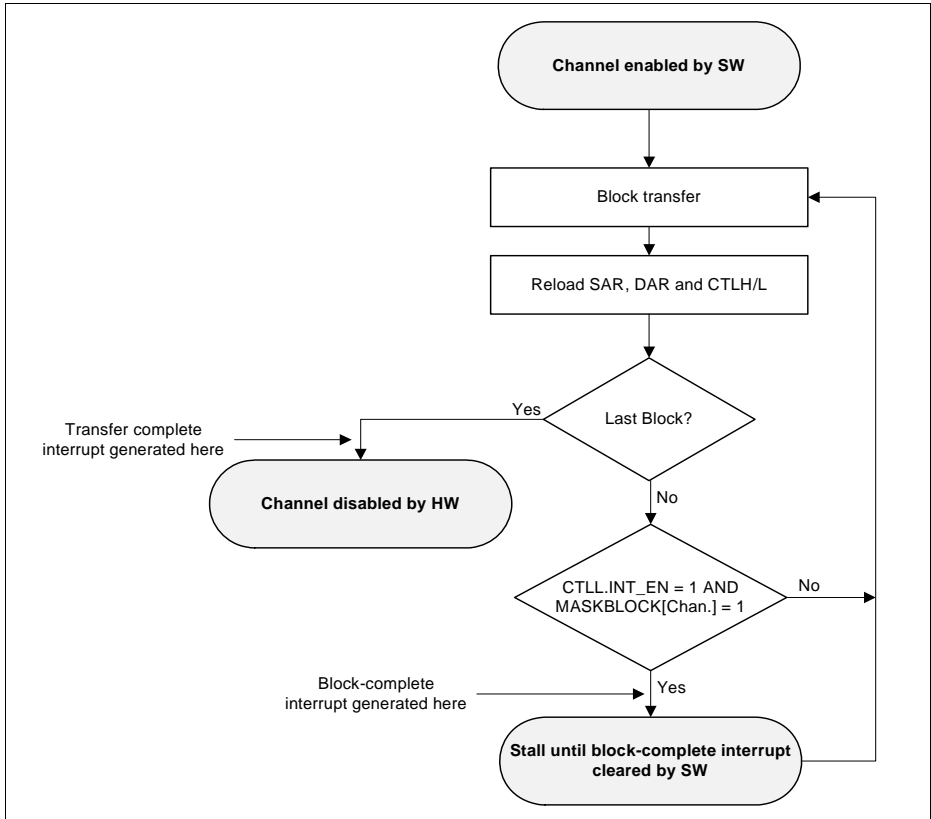


Figure 5-21 DMA Transfer Flow for Source and Destination Address Auto-Reloaded

5.4.6.4 Multi-Block Transfer with Source Address Auto-Reloaded and Linked List Destination Address

This section is an example for the transfer listed in row 7 in [Table 5-5](#).

Note: This type of transfer is supported by GPDMA0 channels 0 and 1 only.

1. Read the Channel Enable register (see [GPDMA0_CHENREG](#)) in order to choose a free (disabled) channel.
2. Set up the chain of linked list items (otherwise known as block descriptors) in memory. Write the control information in the LLI.CTL register location of the block descriptor for each LLI in memory (see [Figure 5-14](#)) for channel x.
3. Write the starting source address in the [SAR](#) register for channel x.
Note: *The values in the LLI.SAR register locations of each of the Linked List Items (LLIs) set up in memory, although fetched during an LLI fetch, are not used.*
4. Write the channel configuration information into the [CFG](#) register for channel x.
 - a) Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
This step requires programming the HS_SEL_SRC, HS_SEL_DST bits. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface source/destination requests.
 - b) If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral; this requires programming the SRC_PER and DEST_PER bits, respectively.
5. Make sure that the LLI.CTLH/L register locations of all LLIs in memory (except the last) are set as shown in Row 7 of [Table 5-5](#), while the LLI.CTLH/L register of the last Linked List item must be set as described in Row 1 or Row 5 of [Table 5-5](#). Figure 7-1 shows a Linked List example with two list items.
6. Ensure that the LLI.LLP register locations of all LLIs in memory (except the last) are non-zero and point to the next Linked List Item.
7. Ensure that the LLI.DAR register location of all LLIs in memory point to the start destination block address preceding that LLI fetch.
8. Ensure that the LLI.CTLH/L.DONE fields of the LLI.CTLH/L register locations of all LLIs in memory are cleared.
9. If source status fetching is enabled ([CFG.SS_UPD_EN](#) is enabled), program the [SSTATAR](#) register so that the source status information can be fetched from the location pointed to by the [SSTATAR](#). For conditions under which the source status information is fetched from system memory, refer to the Write Back column of [Table 5-5](#).
10. If destination status fetching is enabled ([CFG.DS_UPD_EN](#) is enabled), program the [DSTATAR](#) register so that the destination status information can be fetched from the location pointed to by the [DSTATAR](#) register. For conditions under which the

General Purpose DMA (GPDMA)

destination status information is fetched from system memory, refer to the Write Back column of **Table 5-5**.

11. If gather is enabled (**CTL.SRC_GATHER_EN** = 1), program the **SGR** register for channel x.
12. If scatter is enabled (**CTL.DST_SCATTER_EN** = 1), program the **DSR** register for channel x.
13. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: **CLEARTRF**, **CLEARBLOCK**, **CLEARSRCTRAN**, **CLEAR DSTTRAN**, and **CLEARERR**. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
14. Program the **CTL** and **CFG** registers according to Row 7, as shown in **Table 5-5**.
15. Program the **LLP** register with **LLP(0)**, the pointer to the first Linked List item.
16. Finally, enable the channel by writing a 1 to the **GPDMA0_CHENREG.CH_EN** bit; the transfer is performed. Ensure that bit 0 of the **GPDMA0_DMACFGREG** register is enabled.
17. The GPDMA fetches the first LLI from the location pointed to by **LLP(0)**.
Note: *The **LLI.SAR**, **LLI.DAR**, **LLI.LLP**, and **LLI.CTLH/L** registers are fetched. The **LLI.SAR** register - although fetched - is not used.*
18. Source and destination request single and burst GPDMA transactions in order to transfer the block of data (assuming non-memory peripherals). The GPDMA acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
19. Once the block of data is transferred, the source status information is fetched from the location pointed to by the **SSTATAR** register and stored in the **SSTAT** register if **CFG.SS_UPD_EN** is enabled. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of **Table 5-5**.
The destination status information is fetched from the location pointed to by the **DSTATAR** register and stored in the **DSTAT** register if **CFG.DS_UPD_EN** is enabled. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of **Table 5-5**.
20. The **CTLH** register is written out to system memory. For conditions under which the **CTLH** register is written out to system memory, refer to the Write Back column of **Table 5-5**.
The **CTLH** register is written out to the same location where it was originally fetched; that is, the location of the **CTL** register of the linked list item fetched prior to the start of the block transfer. Only the **CTLH** register is written out, because only the **CTL.BLOCK_TS** and **CTL.DONE** fields have been updated by hardware within the GPDMA. The **LLI.CTLH/L.DONE** bit is asserted to indicate block completion. Therefore, software can poll the **LLI.CTL.DONE** bit field of the **CTL** register in the LLI to ascertain when a block transfer has completed.
Note: *Do not poll the **CTL.DONE** bit in the GPDMA memory map. Instead, poll the **LLI.CTLH/L.DONE** bit in the LLI for that block. If the polled **LLI.CTLH/L.DONE** bit is*

General Purpose DMA (GPDMA)

asserted, then this block transfer has completed. This `LLI.CTLH/L.DONE` bit was cleared at the start of the transfer (Step 8).

21. The **SSTAT** register is now written out to system memory if `CFG.SS_UPD_EN` is enabled. It is written to the SSTAT register location of the LLI pointed to by the previously saved **LLP.LOC** register.

The **DSTAT** register is now written out to system memory if `CFG.DS_UPD_EN` is enabled. It is written to the DSTAT register location of the LLI pointed to by the previously saved **LLP.LOC** register.

The end-of-block interrupt, `int_block`, is generated after the write-back of the control and status registers has completed.

Note: *The write-back location for the control and status registers is the LLI pointed to by the previous value of the **LLP.LOC** register, not the LLI pointed to by the current value of the **LLP.LOC** register.*

22. The GPDMA reloads the **SAR** register from the initial value. Hardware sets the block-complete interrupt. The GPDMA samples the row number, as shown in [Table 5-5](#). If the GPDMA is in Row 1 or Row 5, then the DMA transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. You can either respond to the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register (`RAWTFR[n]`, $n = \text{channel number}$) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, `CLEARTR[n]`, before the channel is enabled. If the GPDMA is not in Row 1 or Row 5 as shown in [Table 5-5](#), the following steps are performed.

23. The DMA transfer proceeds as follows:

a) If interrupts are enabled (`CTL.INT_EN = 1`) and the block-complete interrupt is unmasked (`MASKBLOCK[x] = 1B`, where x is the channel number), hardware sets the block-complete interrupt when the block transfer has completed. It then stalls until the block-complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block-complete ISR (interrupt service routine) should clear the `CFG.RELOAD_SRC` source reload bit. This puts the GPDMA into Row 1, as shown in [Table 5-5](#). If the next block is not the last block in the DMA transfer, then the source reload bit should remain enabled to keep the GPDMA in Row 7, as shown in [Table 5-5](#).

b) If interrupts are disabled (`CTL.INT_EN = 0`) or the block-complete interrupt is masked (`MASKBLOCK[x] = 0B`, where x is the channel number), then hardware does not stall until it detects a write to the block-complete interrupt clear register; instead, it immediately starts the next block transfer. In this case, software must clear the source reload bit, `CFG.RELOAD_SRC` in order to put the device into Row 1 of [Table 5-5](#) before the last block of the DMA transfer has completed.

24. The GPDMA fetches the next LLI from memory location pointed to by the current **LLP** register and automatically reprograms the **DAR**, **CTL**, and **LLP** channel registers. Note that the **SAR** is not reprogrammed, since the reloaded value is used for the next

General Purpose DMA (GPDMA)

DMA block transfer. If the next block is the last block of the DMA transfer, then the **CTL** and **LLP** registers just fetched from the LLI should match Row 1 or Row 5 of **Table 5-5**.

The DMA transfer might look like that shown in **Figure 5-22**.

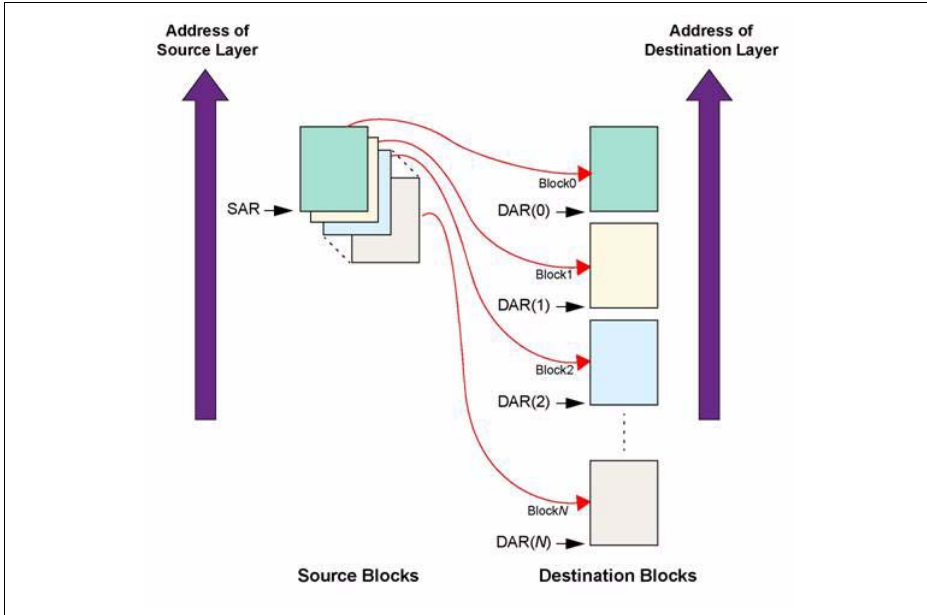


Figure 5-22 Multi-Block DMA Transfer with Source Address Auto-Reloaded and Linked List Destination Address

The DMA transfer flow is shown in **Figure 5-23**.

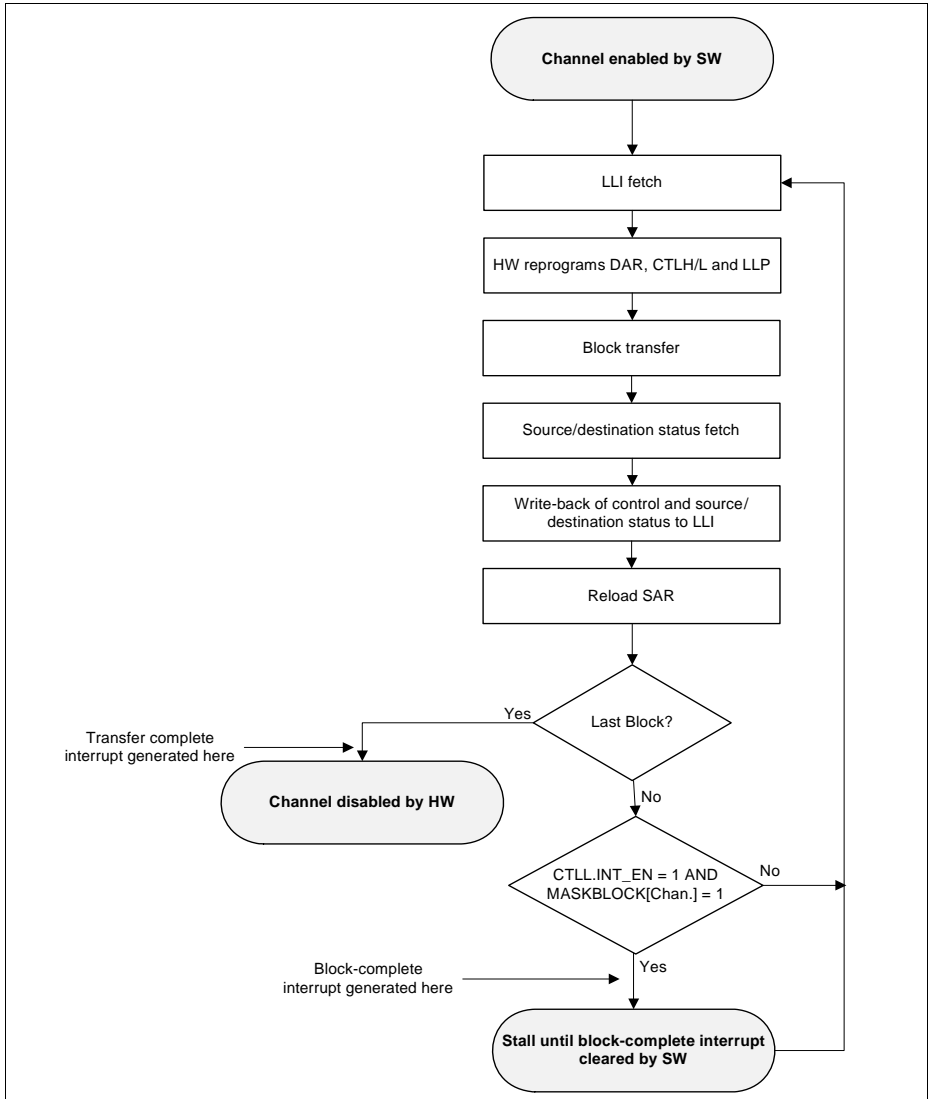


Figure 5-23 DMA Transfer Flow for Source Address Auto-Reloaded and Linked List Destination Address

5.4.6.5 Multi-Block DMA Transfer with Linked List for Source and Contiguous Destination Address

This section is an example for the transfer listed in row 8 in [Table 5-5](#).

Note: This type of transfer is supported by GPDMA0 channels 0 and 1 only.

1. Read the Channel Enable register (see [GPDMA0_CHENREG](#)) to choose a free (disabled) channel.
2. Set up the linked list in memory. Write the control information in the LLI.CTL register location of the block descriptor for each LLI in memory (see [Figure 5-14](#)) for channel x.
3. Write the starting destination address in the [DAR](#) register for channel x.
Note: *The values in the LLI.DAR register location of each Linked List Item (LLI) in memory, although fetched during an LLI fetch, are not used.*
4. Write the channel configuration information into the [CFG](#) register for channel x.
 1. Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
This step requires programming the HS_SEL_SRC, HS_SEL_DST bits. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.
 2. If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripherals. This requires programming the SRC_PER and DEST_PER bits, respectively.
5. Ensure that all LLI.CTLH/L register locations of the LLI (except the last) are set as shown in Row 8 of [Table 5-5](#), while the LLI.CTLH/L register of the last Linked List item must be set as described in Row 1 or Row 5 of [Table 5-5](#). [Figure 5-14](#) shows a Linked List example with two list items.
6. Ensure that the LLI.LLP register locations of all LLIs in memory (except the last) are non-zero and point to the next Linked List Item.
7. Ensure that the LLI.SAR register location of all LLIs in memory point to the start source block address preceding that LLI fetch.
8. Ensure that the LLI.CTLH/L.DONE fields of the LLI.CTLH/L register locations of all LLIs in memory are cleared.
9. If source status fetching is enabled ([CFG.SS_UPD_EN](#) is enabled), program the [SSTATAR](#) register so that the source status information can be fetched from the location pointed to by [SSTATAR](#). For conditions under which the source status information is fetched from system memory, refer to the Write Back column of [Table 5-5](#).
10. If destination status fetching is enabled ([CFG.DS_UPD_EN](#) is enabled), program the [DSTATAR](#) register so that the destination status information can be fetched from the location pointed to by the [DSTATAR](#) register. For conditions under which the

General Purpose DMA (GPDMA)

destination status information is fetched from system memory, refer to the Write Back column of **Table 5-5**.

11. If gather is enabled (**CTL.SRC_GATHER_EN** = 1), program the **SGR** register for channel x.
12. If scatter is enabled (**CTL.DST_SCATTER_EN** = 1), program the **DSR** register for channel x.
13. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: **CLEARTRF**, **CLEARBLOCK**, **CLEARSRCTRAN**, **CLEAR DSTTRAN**, and **CLEARERR**. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
14. Program the **CTL** and **CFG** registers according to Row 8, as shown in **Table 5-5**.
15. Program the **LLP** register with **LLP(0)**, the pointer to the first Linked List item.
16. Finally, enable the channel by writing a 1 to the **GPDMA0_CHENREG.CH_EN** bit; the transfer is performed. Ensure that bit 0 of the **GPDMA0_DMACFGREG** register is enabled.
17. The GPDMA fetches the first LLI from the location pointed to by **LLP(0)**.
Note: *The LLI.SAR, LLI.DAR, LLI.LLP, and LLI.CTLH/L registers are fetched. The LLI.DAR register location of the LLI - although fetched - is not used. The DAR register in the GPDMA remains unchanged.*
18. Source and destination request single and burst GPDMA transactions to transfer the block of data (assuming non-memory peripherals). The GPDMA acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.
19. Once the block of data is transferred, the source status information is fetched from the location pointed to by the **SSTATAR** register and stored in the **SSTAT** register if **CFG.SS_UPD_EN** is enabled. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of **Table 5-5**. The destination status information is fetched from the location pointed to by the **DSTATAR** register and stored in the **DSTAT** register if **CFG.DS_UPD_EN** is enabled. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of **Table 5-5**.
20. The **CTLH** register is written out to system memory. For conditions under which the **CTLH** register is written out to system memory, refer to the Write Back column of **Table 5-5**. The **CTLH** register is written out to the same location where it was originally fetched; that is, the location of the **CTL** register of the linked list item fetched prior to the start of the block transfer. Only the second word of the **CTL** register is written out, **CTLH**, because only the **CTL.BLOCK_TS** and **CTL.DONE** fields have been updated by hardware within the GPDMA. Additionally, the **CTL.DONE** bit is asserted to indicate block completion. Therefore, software can poll the LLI.**CTL.DONE** bit field of the **CTL** register in the LLI to ascertain when a block transfer has completed.

Note: Do not poll the **CTL.DONE** bit in the GPDMA memory map. Instead, poll the LLI.**CTLH/L.DONE** bit in the LLI for that block. If the polled LLI.**CTLH/L.DONE** bit is

General Purpose DMA (GPDMA)

asserted, then this block transfer has completed. This LLI.CTLH/L.DONE bit was cleared at the start of the transfer (Step 8).

21. The **SSTAT** register is now written out to system memory if **CFG.SS_UPD_EN** is enabled. It is written to the **SSTAT** register location of the LLI pointed to by the previously saved **LLP.LOC** register. The **DSTAT** register is now written out to system memory if **CFG.DS_UPD_EN** is enabled. It is written to the **DSTAT** register location of the LLI pointed to by the previously saved **LLP.LOC** register. The end-of-block interrupt, **int_block**, is generated after the write-back of the control and status registers has completed.

Note: The write-back location for the control and status registers is the LLI pointed to by the previous value of the **LLP.LOC** register, not the LLI pointed to by the current value of the **LLP.LOC** register.

22. The GPDMA does not wait for the block interrupt to be cleared, but continues and fetches the next LLI from the memory location pointed to by the current **LLP** register and automatically reprograms the **SAR**, **CTL**, and **LLP** channel registers. The **DAR** register is left unchanged. The DMA transfer continues until the GPDMA samples that the **CTL** and **LLP** registers at the end of a block transfer match those described in Row 1 or Row 5 of **Table 5-5** (as discussed earlier). The GPDMA then knows that the previously transferred block was the last block in the DMA transfer.

The GPDMA transfer might look like that shown in **Figure 5-24**. Note that the destination address is decrementing.

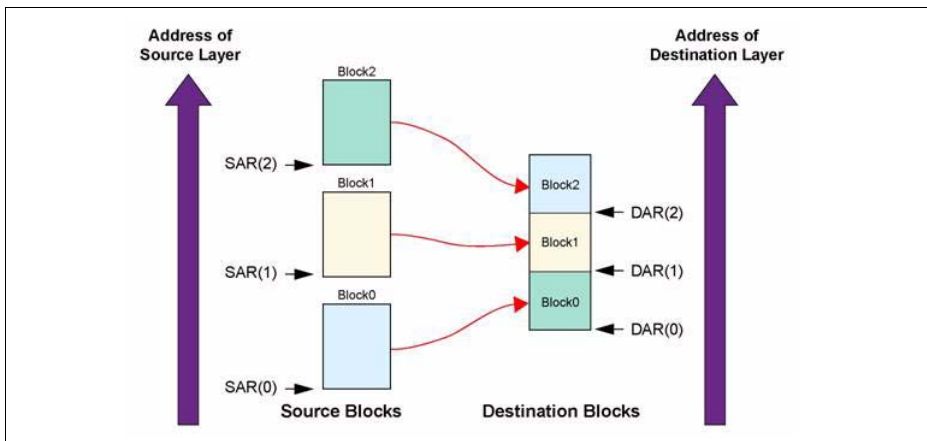


Figure 5-24 Multi-Block DMA Transfer with Linked List Source Address and Contiguous Destination Address

The DMA transfer flow is shown in **Figure 5-25**.

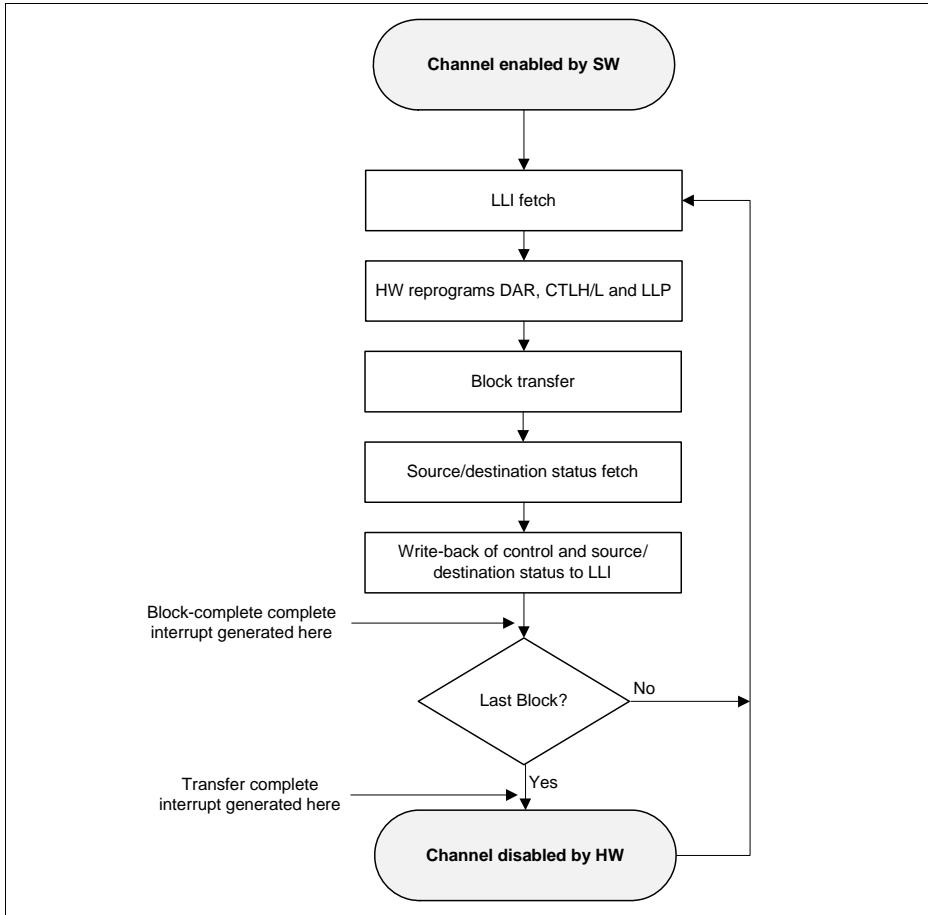


Figure 5-25 DMA Transfer Flow for Source Address Auto-Reloaded and Linked List Destination Address

5.4.6.6 Multi-Block Transfer with Linked List for Source and Destination

This section is an example for the transfer listed in row 10 in **Table 5-5**.

Note: This type of transfer is supported by GPDMA0 channels 0 and 1 only.

1. Read the Channel Enable register (see **GPDMA0_CHENREG**) to choose a free (disabled) channel.

General Purpose DMA (GPDMA)

2. Set up the chain of Linked List Items (otherwise known as block descriptors) in memory. Write the control information in the LLI.**CTL** register location of the block descriptor for each LLI in memory (see **Figure 5-14**) for channel x.
3. Write the channel configuration information into the **CFG** register for channel x.
 - a) Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
This step requires programming the **CFG.HS_SEL_SRC** or **CFG.HS_SEL_DST** bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a 1 activates the software handshaking interface to handle source/destination requests.
 - b) If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the **CFG.SRC_PER** and **CFG.DEST_PER** bits, respectively.
4. Make sure that the LLI.CTLH/L register locations of all LLI entries in memory (except the last) are set as shown in Row 10 of **Table 5-5**. The LLI.CTLH/L register of the last Linked List Item must be set as described in Row 1 or Row 5 of **Table 5-5**. **Figure 5-14** shows a Linked List example with two list items.
5. Make sure that the LLI.LLP register locations of all LLI entries in memory (except the last) are non-zero and point to the base address of the next Linked List Item.
6. Make sure that the LLI.SAR, LLI.DAR register locations of all LLI entries in memory point to the start source/destination block address preceding that LLI fetch.
7. Ensure that the LLI.CTLH/L.DONE field of the LLI.CTLH/L register locations of all LLI entries in memory is cleared.
8. If source status fetching is enabled (**CFG.SS_UPD_EN** is enabled), program the **SSTATAR** register so that the source status information can be fetched from the location pointed to by the **SSTATAR**. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of **Table 5-5**.
9. If destination status fetching is enabled (**CFG.DS_UPD_EN** is enabled), program the **DSTATAR** register so that the destination status information can be fetched from the location pointed to by the **DSTATAR** register. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of **Table 5-5**.
10. If gather is enabled (**CTL.SRC_GATHER_EN** = 1), program the **SGR** register for channel x.
11. If scatter is enabled (**CTL.DST_SCATTER_EN** = 1), program the **DSR** register for channel x.
12. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: **CLEARFR**, **CLEARBLOCK**, **CLEARSRCTRAN**, **CLEARDESTTRAN**, and **CLEARERR**. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
13. Program the **CTL** and **CFG** registers according to Row 10, as shown in **Table 5-5**.

General Purpose DMA (GPDMA)

14. Program the **LLP** register with LLP(0), the pointer to the first linked list item.
15. Finally, enable the channel by writing a 1 to the **GPDMA0_CHENREG.CH_EN** bit; the transfer is performed.
16. The GPDMA fetches the first LLI from the location pointed to by LLP(0). The LLI.SAR, LLI.DAR, LLI.LLP, and LLI.CTL registers are fetched and the GPDMA automatically reprograms the according **SAR**, **DAR**, **LLP**, and **CTL** channel registers.
17. Source and destination request single and burst DMA transactions to transfer the block of data (assuming non-memory peripheral). The GPDMA acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.

18. Once the block of data is transferred, the source status information is fetched from the location pointed to by the **SSTATAR** register and stored in the **SSTAT** register if **CFG.SS_UPD_EN** is enabled. For conditions under which the source status information is fetched from system memory, refer to the Write Back column of **Table 5-5**.

The destination status information is fetched from the location pointed to by the **DSTATAR** register and stored in the **DSTAT** register if **CFG.DS_UPD_EN** is enabled. For conditions under which the destination status information is fetched from system memory, refer to the Write Back column of **Table 5-5**.

19. The **CTLH** register is written out to system memory. For conditions under which the **CTLH** register is written out to system memory, refer to the Write Back column of **Table 5-5**.

The **CTLH** register is written out to the same location where it was originally fetched; that is, the location of the **CTL** register of the linked list item fetched prior to the start of the block transfer. Only the **CTLH** register is written out, because only the **CTL.BLOCK_TS** and **CTL.DONE** fields have been updated by the GPDMA hardware. Additionally, the **CTL.DONE** bit is asserted to indicate block completion. Therefore, software can poll the LLI.CTLH/L.DONE bit of the **CTL** register in the LLI to ascertain when a block transfer has completed.

Note: Do not poll the **CTL.DONE** bit in the GPDMA memory map; instead, poll the LLI.CTLH/L.DONE bit in the LLI for that block. If the polled LLI.CTLH/L.DONE bit is asserted, then this block transfer has completed. This LLI.CTLH/L.DONE bit was cleared at the start of the transfer (Step 7).

20. The **SSTAT** register is now written out to system memory if **CFG.SS_UPD_EN** is enabled. It is written to the **SSTAT** register location of the LLI pointed to by the previously saved **LLP.LOC** register.

The **DSTAT** register is now written out to system memory if **CFG.DS_UPD_EN** is enabled. It is written to the **DSTAT** register location of the LLI pointed to by the previously saved **LLP.LOC** register.

The end-of-block interrupt, **int_block**, is generated after the write-back of the control and status registers has completed.

Note: The write-back location for the control and status registers is the LLI pointed to

General Purpose DMA (GPDMA)

by the previous value of the **LLP.LOC** register, not the LLI pointed to by the current value of the **LLP.LOC** register.

21. The GPDMA does not wait for the block interrupt to be cleared, but continues fetching the next LLI from the memory location pointed to by the current **LLP** register and automatically reprograms the **SAR**, **DAR**, **CTL**, and **LLP** channel registers. The DMA transfer continues until the GPDMA determines that the **CTL** and **LLP** registers at the end of a block transfer match the ones described in Row 1 or Row 5 of **Table 5-5** (as discussed earlier). The GPDMA then knows that the previously transferred block was the last block in the DMA transfer.

The DMA transfer might look like that shown in **Figure 5-26**.

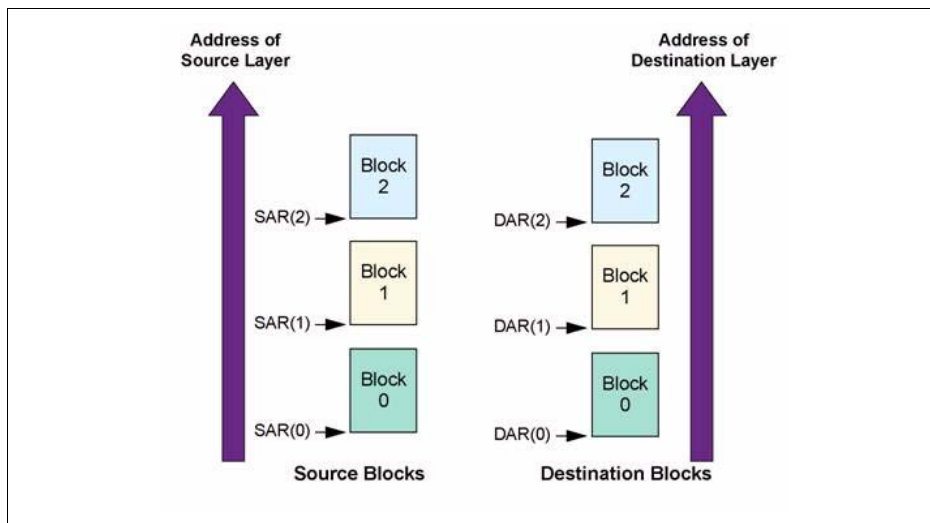


Figure 5-26 Multi-Block with Linked Address for Source and Destination

If the user needs to execute a DMA transfer where the source and destination address are contiguous, but where the amount of data to be transferred is greater than the maximum block size **CTL.BLOCK_TS**, then this can be achieved using the type of multi-block transfer shown in **Figure 5-27**.

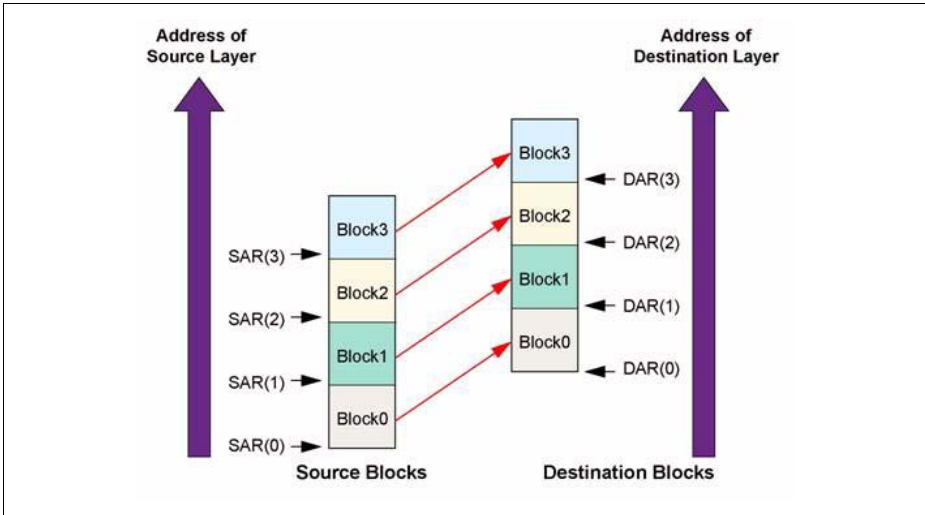


Figure 5-27 Multi-Block with Linked Address for Source and Destination Where SAR and DAR Between Successive Blocks are Contiguous

The DMA transfer flow is shown in **Figure 5-28**.

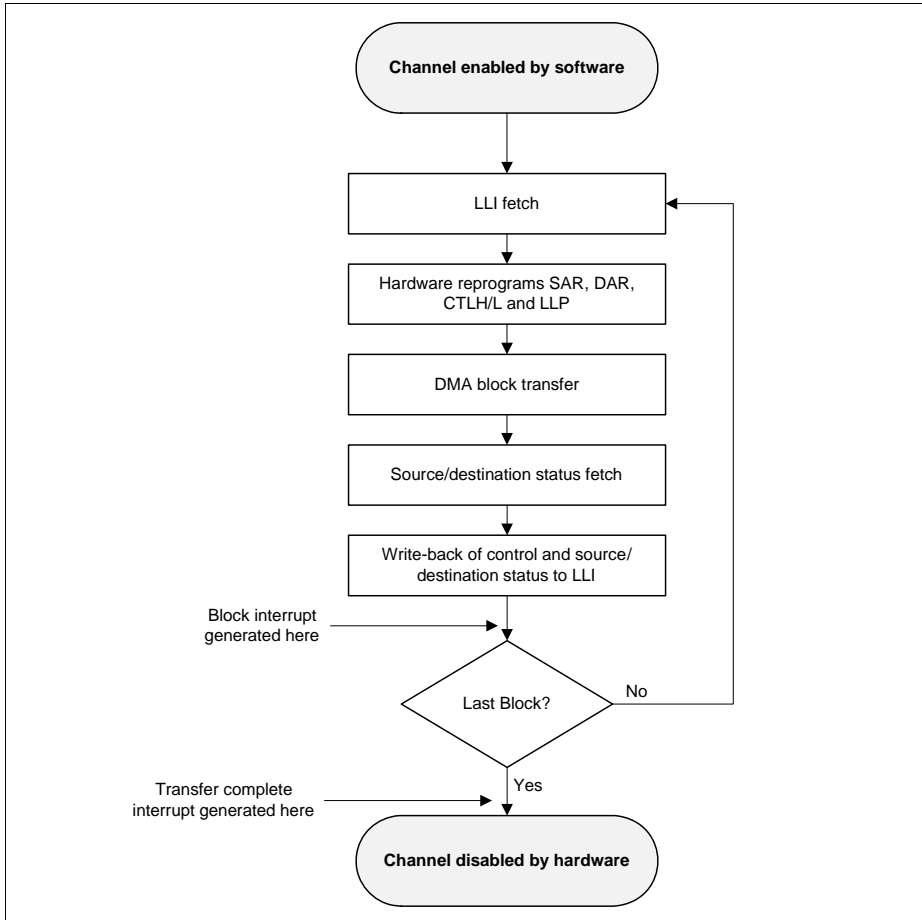


Figure 5-28 DMA Transfer Flow for Source and Destination Linked List Address

5.5 Service Request Generation

Each GPDMA block provides a number of registers (see [Section 5.8.3](#)) to control the request behavior and to provide an interface for software to check for request occurrence.

The following DMA Events can be generated for each channel due to DMA activity:

- IntSrcTran - Source Transaction Complete
- IntDstTran - Destination Transaction Complete
- IntBlock - Block Transfer Complete
- IntTfr - DMA Transfer Complete
- IntErr - Error

DMA Event processing per channel

Each DMA Event for each channel is directly stored in the according “RAW Status” bit shown in [Figure 5-29](#). The user software can control the processing by writing to the according “Mask” and “Clear” bits.

Note: Request forwarding is disabled by default setting of the “Mask” register.

Once the event is forwarded to the “Status” bit its occurrence is registered in the [Combined Interrupt Status Register](#) and a service request is triggered to the NVIC.

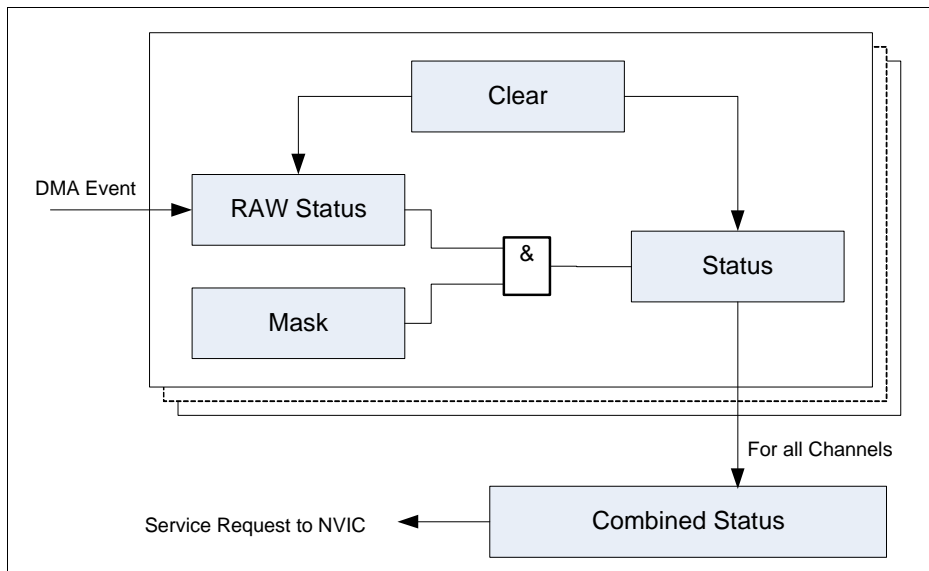


Figure 5-29 DMA Event to Service Request Flow

5.6 Power, Reset and Clock

The GPDMA unit is inside the core power domain, therefore no special considerations about power up or power down sequences need to be taken. For an explanation about the different power domains, please address the SCU (System Control Unit) chapter.

Additionally, if a GPDMA unit is not needed, it can be held in reset via the PRSET2.DMAyRS bitfield (address the SCU chapter for a full description).

The clock used for the GPDMA unit is described on the SCU chapter as f_{DMA} . Please address the specific section under the SCU chapter for a detailed description on the clock configuration schemes.

5.7 Initialization and System Dependencies

The generic initialization sequence for an application that is using the GPDMA, should be the following:

1st Step: Release reset of the GPDMA, via the specific SCU bitfield on the PRCLR2 register.

2nd Step: If the GPDMA is already under use (step 1 was not performed) do the following steps:

- read the channel Enable register to choose a free channel, **CHENREG**. Clear also any pending requests of the specific channel, by writing into the **CLEARTRF**, **CLEARBLOCK**, **CLEARSRCTAN**, **CLEARSTTRAN** and **CLEARERR**
- confirm that all the interrupts have been cleared via the Status and RAW registers.

3rd Step: Configure the GPDMA channels accordingly with the wanted transfer type:

- Configure the starting source address and starting destination address, on the **SAR** and **DAR**, respectively.
- Configure the type of transfer that are going to be used via the **LLP**, **CTL** and **CFG** registers.

4th Step: Enable the GPDMA channel, by setting the specific bitfield on the **CHENREG**.

5th Step: Configure the DLR (DMA Line Router) block to map the DMA requests from the peripherals to the wanted DMA request lines (if not previously done).

6th Step: Configure the peripherals that are linked with DMA requests.

7th Step: Enable the specific Service requests on the peripheral blocks.

8th Step: Start the peripheral(s)

*Note: This is a generic channel initialization example. Please refer to **Section 5.3** and **Section 5.4** for a complete description and examples of how to control the complete flow for a GPDMA channel.*

5.8 Registers

This chapter includes information on how to program the GPDMA.

Register references

There are references to software parameters throughout this chapter. The software parameters are the field names in each register description table and are prefixed by the register name; for example, the Block Transfer Size field in the Control register for channel x of GPDMA0 is designated as "GPDMA0_CHx_CTLH.BLOCK_TS"

Illegal Register Access

An illegal access can be any of the following:

1. A write to the **SAR**, **DAR**, **LLP**, **CTL**, **SSTAT**, **DSTAT**, **SSTATAR**, **DSTATAR**, **SGR**, or **DSR** registers occurs when the channel is enabled.
2. A read from the Interrupt Clear Registers is attempted.
3. A write to the Interrupt Status Registers, **GPDMA0_STATUSINT**, **ID** or **VERSION** is attempted.

An illegal access (read/write) returns an AHB error response.

Table 5-6 Registers Address Space

Module	Base Address	End Address	Note
GPDMA0_CH0	5001 4000 _H	5001 4054 _H	
GPDMA0_CH1	5001 4058 _H	5001 40AC _H	
GPDMA0_CH2	5001 40B0 _H	5001 4104 _H	
GPDMA0_CH3	5001 4108 _H	5001 415C _H	
GPDMA0_CH4	5001 4160 _H	5001 41B4 _H	
GPDMA0_CH5	5001 41B8 _H	5001 420C _H	
GPDMA0_CH6	5001 4210 _H	5001 4264 _H	
GPDMA0_CH7	5001 4268 _H	5001 42BC _H	
GPDMA0	5001 4000 _H	5001 7FFF _H	

Table 5-7 Register Overview

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
Channel Registers					
SAR	Source Address Register	0000 _H + x*58 _H	U, PV	U, PV	Page 5-65
DAR	Destination Address Register	0008 _H + x*58 _H	U, PV	U, PV	Page 5-66
Control Registers					
CTLH	Control Register High	001C _H + x*5C _H	U, PV	U, PV	Page 5-69
CTLL	Control Register Low	0018 _H + x*58 _H	U, PV	U, PV	Page 5-71
LLP	Linked List Pointer Register	0010 _H + x*58 _H	U, PV	U, PV	Page 5-68
SSTAT	Source Status Register	0020 _H + x*58 _H	U, PV	U, PV	Page 5-77
DSTAT	Destination Status Register	0028 _H + x*58 _H	U, PV	U, PV	Page 5-78
SSTATAR	Source Status Register	0030 _H + x*58 _H	U, PV	U, PV	Page 5-79
DSTATAR	Destination Status Register	0038 _H + x*58 _H	U, PV	U, PV	Page 5-80
CFGH	Configuration Register High	0044 _H + x*5C _H	U, PV	U, PV	Page 5-81
CFGL	Configuration Register Low	0040 _H + x*58 _H	U, PV	U, PV	Page 5-87
SGR	Source Gather Register	0048 _H + x*58 _H	U, PV	U, PV	Page 5-94
DSR	Destination Scatter Register	0050 _H + x*58 _H	U, PV	U, PV	Page 5-95
Interrupt Registers					

General Purpose DMA (GPDMA)

Table 5-7 Register Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
RAW* with *TFR, *BLOCK, *SRCTRAN, *DSTTRAN, *ERR	Interrupt Raw Status Registers	02C0 _H - 02E0 _H	U, PV	U, PV	Page 5-98
STATUS* with *TFR, *BLOCK, *SRCTRAN, *DSTTRAN, *ERR	Interrupt Status Registers	02E8 _H - 0308 _H	U, PV	U, PV	Page 5-100
MASK* with *TFR, *BLOCK, *SRCTRAN, *DSTTRAN, *ERR	Interrupt Mask Registers	0310 _H - 0330 _H	U, PV	U, PV	Page 5-102
CLEAR* with *TFR, *BLOCK, *SRCTRAN, *DSTTRAN, *ERR	Interrupt Clear Registers	0338 _H - 0358 _H	U, PV	U, PV	Page 5-104
STATUSINT	Combined Interrupt Status Register	0360 _H	U, PV	U, PV	Page 5-105
Software Handshaking Registers					
REQSRCREG	Source Software Transaction Request Register	0368 _H	U, PV	U, PV	Page 5-107
REQDSTREG	Destination Software Transaction Request Register	0370 _H	U, PV	U, PV	Page 5-108
SGLREQSRCR EG	Single Source Transaction Request Register	0378 _H	U, PV	U, PV	Page 5-109

General Purpose DMA (GPDMA)

Table 5-7 Register Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
SGLREQDSTR EG	Single Destination Transaction Request Register	0380 _H	U, PV	U, PV	Page 5-109
LSTSRCREG	Last Source Transaction Request Register	0388 _H	U, PV	U, PV	Page 5-110
LSTDSTREG	Last Destination Transaction Request Register	0390 _H	U, PV	U, PV	Page 5-111
Configuration and Channel Enable Registers					
DMACFGREG	Configuration Register	0398 _H	U, PV	U, PV	Page 5-63
CHENREG	Channel Enable Register	03A0 _H	U, PV	U, PV	Page 5-63
Miscellaneous GPDMA Registers					
ID	GPDMA Module ID	03A8 _H	U, PV	U, PV	Page 5-113
Reserved	Reserved	03B0 _H - 03F4 _H	nBE	nBE	
TYPE	GPDMA Component Type	03F8 _H	U, PV	U, PV	Page 5-113
VERSION	GPDMA Component Version	03FC _H	U, PV	U, PV	Page 5-113
Reserved	Reserved	0400 _H - 7FFC _H	nBE	nBE	

1) x = channel number

5.8.1 Configuration and Channel Enable Registers

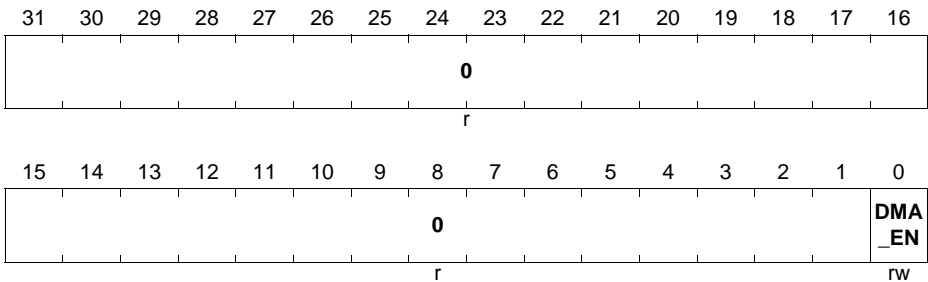
DMACFGREG

This register is used to enable the GPDMA, which must be done before any channel activity can begin.

GPDMA0_DMACFGREG

GPDMA Configuration Register (398_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DMA_EN	0	rw	GPDMA Enable bit. 0 _B GPDMA Disabled 1 _B GPDMA Enabled.
0	[31:1]	r	Reserved

If the global channel enable bit is cleared while any channel is still active, then DMACFGREG.DMA_EN still returns 1 to indicate that there are channels still active until hardware has terminated all activity on all channels, at which point the DMACFGREG.DMA_EN bit returns 0.

CHENREG

This is the GPDMA “Channel Enable Register”. If software needs to set up a new channel, then it can read this register in order to find out which channels are currently inactive; it can then enable an inactive channel with the required priority.

All bits of this register are cleared to 0 when the global GPDMA channel enable bit, **DMACFGREG[0]**, is 0. When the global channel enable bit is 0, then a write to the **CHENREG** register is ignored and a read will always read back 0.

The channel enable bit, **CHENREG.CH_EN**, is written only if the corresponding channel write enable bit, **CHENREG.CH_EN_WE**, is asserted on the same AHB write transfer. For example, writing hex 01x1 writes a 1 into **CHENREG[0]**, while **CHENREG[7:1]**

General Purpose DMA (GPDMA)

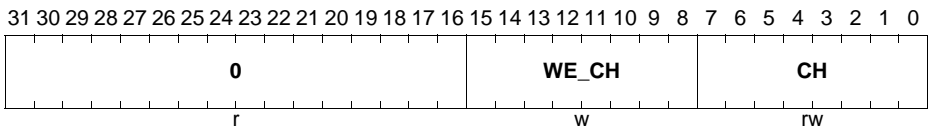
remains unchanged. Writing hex 00xx leaves **CHENREG**[7:0] unchanged. Note that a read-modified write is not required.

GPDMA0_CHENREG

GPDMA Channel Enable Register

(3A0_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CH	[7:0]	rw	Enables/Disables the channel Setting this bit enables a channel; clearing this bit disables the channel. 0 _B Disable the Channel 1 _B Enable the Channel The CHENREG.CH_EN bit is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer.
WE_CH	[15:8]	w	Channel enable write enable
0	[31:16]	r	Reserved

5.8.2 Channel Registers

The **SAR**, **DAR**, **LLP**, **CTL**, and **CFG** channel registers should be programmed prior to enabling the channel. However, if an LLI update occurs before commencing data transfer, **SAR** and **DAR** may not need to be programmed prior to enabling the channel; refer to rows 6 to 10 in **Table 5-5**. It is an illegal register access when a write to the **SAR**, **DAR**, **LLP**, **CTL**, **SSTAT**, **DSTAT**, **SSTATAR**, **DSTATAR**, **SGR**, or **DSR** registers occurs when the channel is enabled.

General Purpose DMA (GPDMA)

SAR

The starting source address is programmed by software before the DMA channel is enabled, or by an LLI update before the start of the DMA transfer. While the DMA transfer is in progress, this register is updated to reflect the source address of the current AHB transfer.

*Note: You must program the SAR address to be aligned to **CTL.SRC_TR_WIDTH**.*

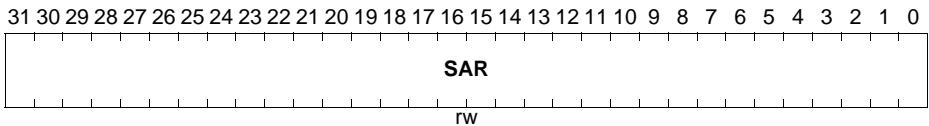
For information on how the **SAR** is updated at the start of each DMA block for multi-block transfers, refer to **Table 5-5**.

GPDMA0_CHx_SAR (x=0-7)

Source Address Register for Channel x

(00_H + x*58_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SAR	[31:0]	rw	Current Source Address of DMA transfer Updated after each source transfer. The SINC field in the CTL register determines whether the address increments, decrements, or is left unchanged on every source transfer throughout the block transfer. Reset: 0 _D

DAR

The starting destination address is programmed by software before the DMA channel is enabled, or by an LLI update before the start of the DMA transfer. While the DMA transfer is in progress, this register is updated to reflect the destination address of the current AHB transfer.

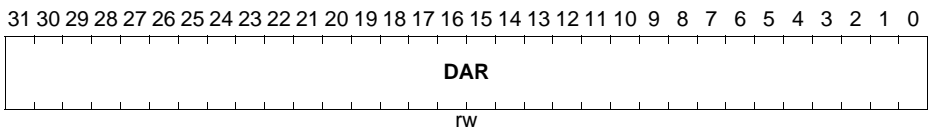
*Note: You must program the DAR to be aligned to **CTL.DST_TR_WIDTH**.*

GPDMA0_CHx_DAR (x=0-7)

Destination Address Register for Channel x

($08_H + x*58_H$)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DAR	[31:0]	rw	Current Destination address of DMA transfer Updated after each destination transfer. The DINC field in the CTL register determines whether the address increments, decrements, or is left unchanged on every destination transfer throughout the block transfer. Reset: 0 _D

Hardware Realignment of SAR/DAR Registers

In a particular circumstance, during contiguous multi-block DMA transfers, the destination address can become misaligned between the end of one block and the start of the next block. When this situation occurs, GPDMA re-aligns the destination address before the start of the next block.

Consider the following example. If the block length is 9, the source transfer width is 16 (halfword), and the destination transfer width is 32 (word) — the destination is programmed for contiguous block transfers — then the destination performs four word transfers followed by a halfword transfer to complete the block transfer to the destination. At the end of the destination block transfer, the address is aligned to a 16-bit transfer as the last AMBA transfer is halfword. This is misaligned to the programmed transfer size of 32 bits for the destination. However, for contiguous destination multi-block transfers, GPDMA re-aligns the DAR address to the nearest 32-bit address (next 32-bit address upwards if address control is incrementing or next address downwards if address control is decrementing).

General Purpose DMA (GPDMA)

The destination address is automatically realigned by the GPDMA in the following DMA transfer setup scenario:

- Contiguous multi-block transfers on destination side, AND
- $DST_TR_WIDTH > SRC_TR_WIDTH$, AND
- $(BLOCK_TS * SRC_TR_WIDTH) / DST_TR_WIDTH \neq \text{integer}$ (where SRC_TR_WIDTH , DST_TR_WIDTH is byte width of transfer)

LLP

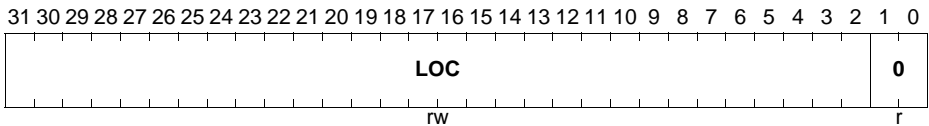
You need to program this register to point to the first Linked List Item (LLI) in memory prior to enabling the channel if block chaining is enabled.

GPDMA0_CHx_LLP (x = 0-1)

Linked List Pointer Register for Channel x

$$(10_H + x * 58_H)$$

Reset Value: 0000 0000_H



Field	Bits	Type	Description
LOC	[31:2]	rw	Starting Address In Memory of next LLI if block chaining is enabled. Note that the two LSBs of the starting address are not stored because the address is assumed to be aligned to a 32-bit boundary.
0	[1:0]	r	Reserved

The LLP register has two functions:

- The logical result of the equation $LLP.LOC \neq 0$ is used to set up the type of DMA transfer — single or multi-block. [Table 5-5](#) shows how the method of updating the channel registers is a function of $LLP.LOC \neq 0$. If $LLP.LOC$ is set to 0, then transfers using linked lists are not enabled. This register must be programmed prior to enabling the channel in order to set up the transfer type.
- $LLP.LOC \neq 0$ contains the pointer to the next LLI for block chaining using linked lists. The **LLP** register can also point to the address where write-back of the control and source/destination status information occur after block completion.

CTL

These registers contain fields that control the DMA transfer.

The CTLH and CTLL registers are part of the block descriptor (linked list item - LLI) when block chaining is enabled. It can be varied on a block-by-block basis within a DMA transfer when block chaining is enabled.

If status write-back is enabled, the upper control register, CTLH, is written to the control register location of the LLI in system memory at the end of the block transfer.

Note: You need to program these registers prior to enabling the channel.

CTLH

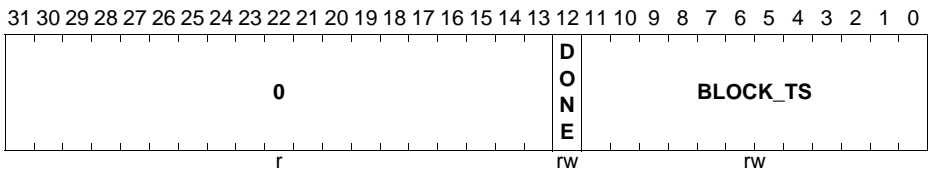
Control Register High.

GPDMA0_CHx_CTLH (x=0-7)

Control Register High for Channel x

(1C_H + x*58_H)

Reset Value: 0000 0002_H



General Purpose DMA (GPDMA)

Field	Bits	Type	Description
BLOCK_TS	[11:0]	rw	<p>Block Transfer Size</p> <p>When the GPDMA is the flow controller, the user writes this field before the channel is enabled in order to indicate the block size. The number programmed here indicates the total number of single transactions to perform for every block transfer.</p> <p>Once the transfer starts, the read-back value is the total number of data items already read from the source peripheral.</p> <p><i>Note: The width of the single transaction is determined by CTL.SRC_TR_WIDTH.</i></p>
DONE	12	rw	<p>Done bit</p> <p>If this bit is set and status write-back is enabled then CTLH is written to the control register location of the Linked List Item (LLI) in system memory at the end of the block transfer.</p> <p>Software can poll the LLI CTLH.DONE bit to see when a block transfer is complete. The LLI CTLH.DONE bit should be cleared when the linked lists are set up in memory prior to enabling the channel.</p>
0	[31:13]	r	Reserved

General Purpose DMA (GPDMA)

CTLL

Control Register Low.

GPDMA0_CHx_CTLL (x=0-1)

Control Register Low for Channel x

$$(18_H + x*58_H)$$

Reset Value: 0030 4801_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		LLP_SRC_EN	LLP_DST_EN	0				TT_FC			0	DST_SC_ATT_EN	SRC_GA_THR_EN	SRC_MSIZ	
r		rw	rw	r				rw			r	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC_MSIZ_E		DEST_MSIZ		SINC		DINC		SRC_TR_WIDTH		DST_TR_WIDTH		INT_EN			
rw		rw		rw		rw		rw		rw		rw			

Field	Bits	Type	Description
INT_EN	0	rw	Interrupt Enable Bit If set, then all interrupt-generating sources are enabled. Functions as a global mask bit for all interrupts for the channel; Raw* interrupt registers still assert if INT_EN = 0.
DST_TR_WIDTH	[3:1]	rw	Destination Transfer Width Table 5-9 lists the decoding for this field.
SRC_TR_WIDTH	[6:4]	rw	Source Transfer Width Table 5-9 lists the decoding for this field.

General Purpose DMA (GPDMA)

Field	Bits	Type	Description
DINC	[8:7]	rw	<p>Destination Address Increment</p> <p>Indicates whether to increment or decrement the destination address on every destination transfer. If your device is writing data to a destination peripheral FIFO with a fixed address, then set this field to "No change".</p> <p>00_B Increment 01_B Decrement 1x_B No change</p> <p><i>Note: Incrementing or decrementing is done for alignment to the next CTLL.DST_TR_WIDTH boundary.</i></p>
SINC	[10:9]	rw	<p>Source Address Increment</p> <p>Indicates whether to increment or decrement the source address on every source transfer. If the device is fetching data from a source peripheral FIFO with a fixed address, then set this field to "No change".</p> <p>00_B Increment 01_B Decrement 1x_B No change</p> <p><i>Note: Incrementing or decrementing is done for alignment to the next CTLL.SRC_TR_WIDTH boundary.</i></p>
DEST_MSIZ	[13:11]	rw	<p>Destination Burst Transaction Length</p> <p>Number of data items, each of width DST_TR_WIDTH, to be written to the destination every time a destination burst transaction request is made from either the corresponding hardware or software handshaking interface. Table 5-8 lists the decoding for this field.</p> <p><i>Note: This value is not related to the AHB bus master HBURST bus.</i></p>

General Purpose DMA (GPDMA)

Field	Bits	Type	Description
SRC_MSIZ	[16:14]	rw	<p>Source Burst Transaction Length Number of data items, each of width SRC_TR_WIDTH, to be read from the source every time a source burst transaction request is made from either the corresponding hardware or software handshaking interface. Table 5-8 lists the decoding for this field;</p> <p><i>Note: This value is not related to the AHB bus master HBURST bus.</i></p>
SRC_GATHER_EN	17	rw	<p>Source gather enable 0_B Gather disabled 1_B Gather enabled Gather on the source side is applicable only when the SINC bit indicates an incrementing or decrementing address control.</p>
DST_SCATTER_EN	18	rw	<p>Destination scatter enable 0_B Scatter disabled 1_B Scatter enabled Scatter on the destination side is applicable only when the DINC bit indicates an incrementing or decrementing address control.</p>
TT_FC	[22:20]	rw	<p>Transfer Type and Flow Control The following transfer types are supported.</p> <ul style="list-style-type: none"> • Memory to Memory • Memory to Peripheral • Peripheral to Memory • Peripheral to Peripheral <p>Table 5-10 lists the decoding for this field.</p>
LLP_DST_EN	27	rw	<p>Linked List Pointer for Destination Enable Block chaining is enabled on the destination side only if the LLP_DST_EN field is high and LLP.LOC is non-zero.</p>
LLP_SRC_EN	28	rw	<p>Linked List Pointer for Source Enable Block chaining is enabled on the source side only if the LLP_SRC_EN field is high and LLP.LOC is non-zero.</p>
0	[31:29], [26:23], 19	r	Reserved

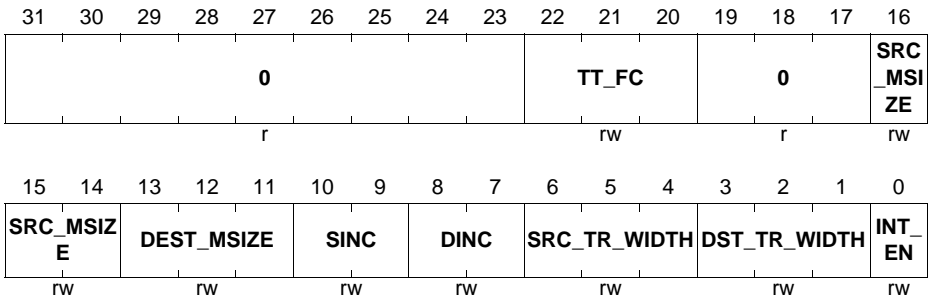
General Purpose DMA (GPDMA)

GPDMA0_CHx_CTLL (x=2-7)

Control Register Low for Channel x

($18_H + x*58_H$)

Reset Value: 0030 4801_H



Field	Bits	Type	Description
INT_EN	0	rw	Interrupt Enable Bit If set, then all interrupt-generating sources are enabled. Functions as a global mask bit for all interrupts for the channel; Raw* interrupt registers still assert if INT_EN = 0.
DST_TR_WIDTH	[3:1]	rw	Destination Transfer Width Table 5-9 lists the decoding for this field.
SRC_TR_WIDTH	[6:4]	rw	Source Transfer Width Table 5-9 lists the decoding for this field.
DINC	[8:7]	rw	Destination Address Increment Indicates whether to increment or decrement the destination address on every destination transfer. If your device is writing data to a destination peripheral FIFO with a fixed address, then set this field to "No change". 00 _B Increment 01 _B Decrement 1x _B No change <i>Note: Incrementing or decrementing is done for alignment to the next CTLL.DST_TR_WIDTH boundary.</i>

General Purpose DMA (GPDMA)

Field	Bits	Type	Description
SINC	[10:9]	rw	<p>Source Address Increment</p> <p>Indicates whether to increment or decrement the source address on every source transfer. If the device is fetching data from a source peripheral FIFO with a fixed address, then set this field to "No change".</p> <p>00_B Increment 01_B Decrement 1x_B No change</p> <p><i>Note: Incrementing or decrementing is done for alignment to the next CTLL.SRC_TR_WIDTH boundary.</i></p>
DEST_MSIZ	[13:11]	rw	<p>Destination Burst Transaction Length</p> <p>Number of data items, each of width DST_TR_WIDTH, to be written to the destination every time a destination burst transaction request is made from either the corresponding hardware or software handshaking interface. Table 5-8 lists the decoding for this field.</p> <p><i>Note: This value is not related to the AHB bus master HBURST bus.</i></p>
SRC_MSIZ	[16:14]	rw	<p>Source Burst Transaction Length</p> <p>Number of data items, each of width SRC_TR_WIDTH, to be read from the source every time a source burst transaction request is made from either the corresponding hardware or software handshaking interface. Table 5-8 lists the decoding for this field.</p> <p><i>Note: This value is not related to the AHB bus master HBURST bus.</i></p>
TT_FC	[22:20]	rw	<p>Transfer Type and Flow Control</p> <p>The following transfer types are supported.</p> <ul style="list-style-type: none"> • Memory to Memory • Memory to Peripheral • Peripheral to Memory • Peripheral to Peripheral <p>Table 5-10 lists the decoding for this field.</p>
0	[31:23], [19:17]	r	Reserved

Table 5-8 CTLL.SRC_MSIZ and CTLL.DST_MSIZ Field Decoding

CTLL.SRC_MSIZ / CTLL.DST_MSIZ	Number of data items to be transferred(of width CTLL.SRC_TR_WIDTH or CTLL.DST_TR_WIDTH)
000 _B	1
001 _B	4
010 _B	8
others	reserved

Table 5-9 CTLL.SRC_TR_WIDTH and CTLL.DST_TR_WIDTH Field Decoding

CTLL.SRC_TR_WIDTH / CTLL.DST_TR_WIDTH	Size (bits)
000 _B	8
001 _B	16
010 _B	32
others	reserved

Table 5-10 CTLL.TT_FC Field Decoding

CTLL.TT_FC Field	Transfer Type	Flow Controller
000 _B	Memory to Memory	GPDMA
001 _B	Memory to Peripheral	GPDMA
010 _B	Peripheral to Memory	GPDMA
011 _B	Peripheral to Peripheral	GPDMA
100 _B	Peripheral to Memory	Peripheral
101 _B	Peripheral to Peripheral	Source Peripheral
110 _B	Memory to Peripheral	Peripheral
111 _B	Peripheral to Peripheral	Destination Peripheral

SSTAT

After each block transfer completes, hardware can retrieve the source status information from the address pointed to by the contents of the **SSTATAR** register. This status information is then stored in the SSTAT register and written out to the SSTAT register location of the LLI before the start of the next block.

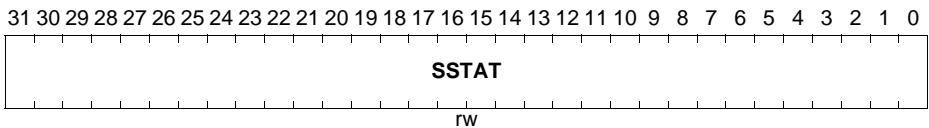
Note: This register is a temporary placeholder for the source status information on its way to the SSTAT register location of the LLI. The source status information should be retrieved by software from the SSTAT register location of the LLI, and not by a read of this register over the GPDMA slave interface.

GPDMA0_CHx_SSTAT (x=0-1)

Source Status Register for Channel x

($20_H + x * 58_H$)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SSTAT	[31:0]	rw	Source Status retrieved by hardware from the address pointed to by the contents of the SSTATAR register.

DSTAT

After the completion of each block transfer, hardware can retrieve the destination status information from the address pointed to by the contents of the **DSTATAR** register. This status information is then stored in the DSTAT register and written out to the DSTAT register location of the LLI before the start of the next block. This register does only exist for channels 0 and 1, for other channels the read-back value is always 0.

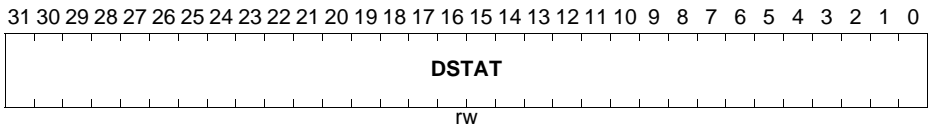
Note: This register is a temporary placeholder for the destination status information on its way to the DSTAT register location of the LLI. The destination status information should be retrieved by software from the DSTAT register location of the LLI and not by a read of this register over the GPDMA slave interface.

GPDMA0_CHx_DSTAT (x=0-1)

Destination Status Register for Channel x

$(28_H + x*58_H)$

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DSTAT	[31:0]	rw	Destination Status retrieved by hardware from the address pointed to by the contents of the DSTATAR register.

General Purpose DMA (GPDMA)

SSTATAR

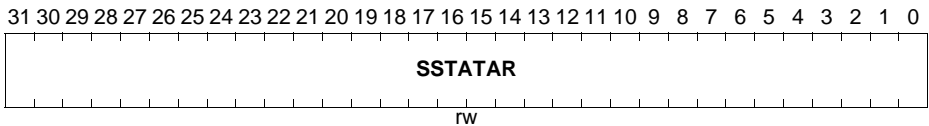
After the completion of each block transfer, hardware can retrieve the source status information from the address pointed to by the contents of the SSTATAR register.

GPDMA0_CHx_SSTATAR (x=0-1)

Source Status Address Register for Channel x

$$(30_H + x * 58_H)$$

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SSTATAR	[31:0]	rw	<p>Source Status Address</p> <p>Pointer from where hardware can fetch the source status information, which is registered in the SSTAT register and written out to the SSTAT register location of the LLI before the start of the next block.</p>

General Purpose DMA (GPDMA)

DSTATAR

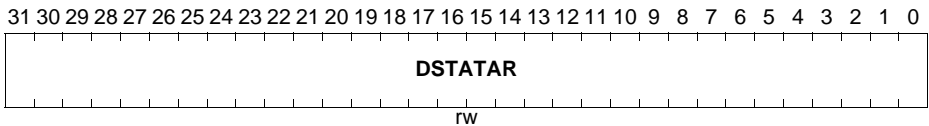
After the completion of each block transfer, hardware can retrieve the destination status information from the address pointed to by the contents of the DSTATAR register.

GPDMA0_CHx_DSTATAR (x=0-1)

Destination Status Address Register for Channel x

$$(38_H + x * 58_H)$$

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DSTATAR	[31:0]	rw	Destination Status Address Pointer from where hardware can fetch the destination status information, which is registered in the DSTAT register and written out to the DSTAT register location of the LLI before the start of the next block.

CFG

These registers contain fields that configure the DMA transfer. The channel configuration register remains fixed for all blocks of a multi-block transfer.

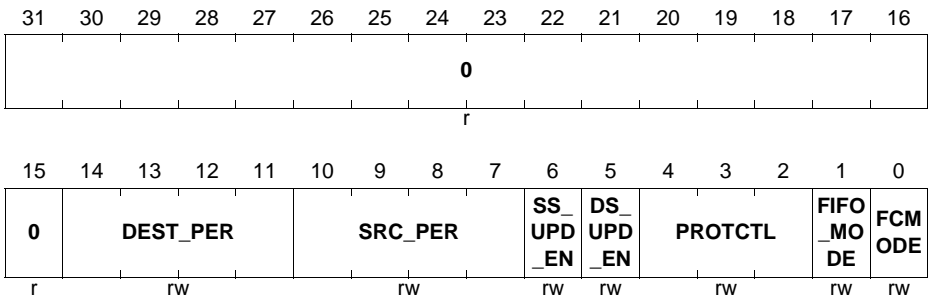
Note: You need to program this register prior to enabling the channel.

GPDMA0_CHx_CFGH (x=0-1)

Configuration Register High for Channel x

(44_H + x*58_H)

Reset Value: 0000 0004_H



General Purpose DMA (GPDMA)

Field	Bits	Type	Description
FCMODE	0	rw	<p>Flow Control Mode</p> <p>Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller.</p> <p>0_B Source transaction requests are serviced when they occur. Data pre-fetching is enabled.</p> <p>1_B Source transaction requests are not serviced until a destination transaction request occurs. In this mode, the amount of data transferred from the source is limited so that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.</p>
FIFO_MODE	1	rw	<p>FIFO Mode Select</p> <p>Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced.</p> <p>0_B Space/data available for single AHB transfer of the specified transfer width.</p> <p>1_B Data available is greater than or equal to half the FIFO depth for destination transfers and space available is greater than half the fifo depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.</p>
PROTCTL	[4:2]	rw	<p>Protection Control</p> <p>Used to drive the AHB HPROT[3:1] bus. The AMBA Specification recommends that the default value of HPROT indicates a non-cached, non-buffered, privileged data access. The reset value is used to indicate such an access. HPROT[0] is tied high because all transfers are data accesses, as there are no opcode fetches.</p> <p>There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals. Table 5-11 shows the mapping of bits in this field to the AHB HPROT[3:1] bus.</p>

General Purpose DMA (GPDMA)

Field	Bits	Type	Description
DS_UPD_EN	5	rw	<p>Destination Status Update Enable</p> <p>Destination status information is fetched only from the location pointed to by the DSTATAR register, stored in the DSTAT register and written out to the DSTAT location of the LLI if DS_UPD_EN is high.</p>
SS_UPD_EN	6	rw	<p>Source Status Update Enable</p> <p>Source status information is fetched only from the location pointed to by the SSTATAR register, stored in the SSTAT register and written out to the SSTAT location of the LLI if SS_UPD_EN is high.</p>
SRC_PER	[10:7]	rw	<p>Source Peripheral</p> <p>Assigns a DLR line as hardware handshaking interface to the source of channel x</p> <p>00_H assigns DLR line 0 01_H assigns DLR line 1 02_H assigns DLR line 2 03_H assigns DLR line 3 04_H assigns DLR line 4 05_H assigns DLR line 5 06_H assigns DLR line 6 07_H assigns DLR line 7 Other values not defined.</p> <p>Notes</p> <ol style="list-style-type: none"> For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface. If GPDMA0_CHx_CFGL.HS_SEL_SRC=1 this field is ignored.

General Purpose DMA (GPDMA)

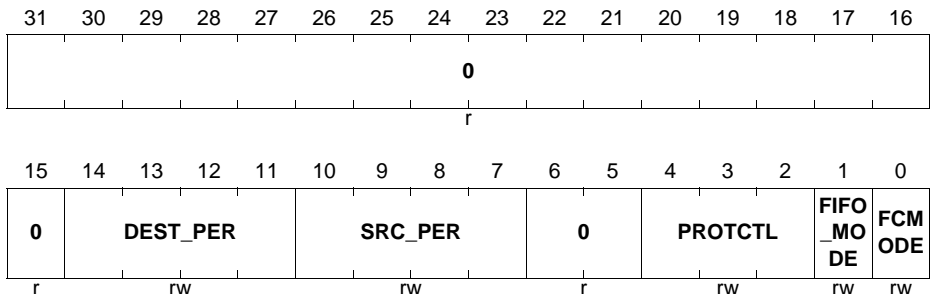
Field	Bits	Type	Description
DEST_PER	[14:11]	rw	<p>Destination Peripheral Assigns a DLR line as hardware handshaking interface to the destination of channel x</p> <p>00_H assigns DLR line 0 01_H assigns DLR line 1 02_H assigns DLR line 2 03_H assigns DLR line 3 04_H assigns DLR line 4 05_H assigns DLR line 5 06_H assigns DLR line 6 07_H assigns DLR line 7 Other values not defined.</p> <p>Notes</p> <ol style="list-style-type: none"> For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface. If GPDMA0_CHx_CFGL.HS_SEL_DST=1 this field is ignored.
0	[31:15]	r	Reserved

GPDMA0_CHx_CFGH (x=2-7)

Configuration Register High for Channel x

$$(44_H + x * 58_H)$$

Reset Value: 0000 0004_H



General Purpose DMA (GPDMA)

Field	Bits	Type	Description
FCMODE	0	rw	<p>Flow Control Mode</p> <p>Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller.</p> <p>0_B Source transaction requests are serviced when they occur. Data pre-fetching is enabled.</p> <p>1_B Source transaction requests are not serviced until a destination transaction request occurs. In this mode, the amount of data transferred from the source is limited so that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.</p>
FIFO_MODE	1	rw	<p>FIFO Mode Select</p> <p>Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced.</p> <p>0_B Space/data available for single AHB transfer of the specified transfer width.</p> <p>1_B Data available is greater than or equal to half the FIFO depth for destination transfers and space available is greater than half the fifo depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.</p>
PROTCTL	[4:2]	rw	<p>Protection Control</p> <p>Used to drive the AHB HPROT[3:1] bus. The AMBA Specification recommends that the default value of HPROT indicates a non-cached, non-buffered, privileged data access. The reset value is used to indicate such an access. HPROT[0] is tied high because all transfers are data accesses, as there are no opcode fetches.</p> <p>There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals. Table 5-11 shows the mapping of bits in this field to the AHB HPROT[3:1] bus.</p>

General Purpose DMA (GPDMA)

Field	Bits	Type	Description
SRC_PER	[10:7]	rw	<p>Source Peripheral</p> <p>Assigns a DLR line as hardware handshaking interface to the source of channel x</p> <p>00_H assigns DLR line 0 01_H assigns DLR line 1 02_H assigns DLR line 2 03_H assigns DLR line 3 04_H assigns DLR line 4 05_H assigns DLR line 5 06_H assigns DLR line 6 07_H assigns DLR line 7 Other values not defined.</p> <p>Notes</p> <ol style="list-style-type: none"> 1. For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface. 2. If GPDMA0_CHx_CFGL.HS_SEL_SRC=1 this field is ignored.

General Purpose DMA (GPDMA)

Field	Bits	Type	Description
DEST_PER	[14:11]	rw	<p>Destination Peripheral Assigns a DLR line as hardware handshaking interface to the destination of channel x</p> <p>00_H assigns DLR line 0 01_H assigns DLR line 1 02_H assigns DLR line 2 03_H assigns DLR line 3 04_H assigns DLR line 4 05_H assigns DLR line 5 06_H assigns DLR line 6 07_H assigns DLR line 7 Other values not defined.</p> <p>Notes</p> <ol style="list-style-type: none"> For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface. If GPDMA0_CHx_CFGL.HS_SEL_DST=1 this field is ignored.
0	[31:15], [6:5]	r	Reserved

GPDMA0_CHx_CFGL (x=0-1)

Configuration Register Low for Channel x

(40_H + x*58_H)

Reset Value: 0000 0EX0_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL_OAD_DS_T	REL_OAD_SR_C	MAX_ABRST										SRC_HS_PO_L	DST_HS_PO_L	LOC_K_B	LOC_K_C_H
rw	rw	rw										rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK_B_L	LOCK_CH_L	HS_SEL_SR_C	HS_SEL_DS_T	FIFO_EM_PTY	CH_SUS_P	CH_PRIOR						0			
rw	rw	rw	rw	r	rw	rw						r			

General Purpose DMA (GPDMA)

Field	Bits	Type	Description
CH_PRIOR	[7:5]	rw	<p>Channel priority</p> <p>A priority of 7 is the highest priority, and 0 is the lowest. The value programmed to this field must be within 0 and 7. A programmed value outside this range will cause erroneous behavior.</p> <p>Reset: Channel Number</p> <p>For example: Chan0 = 000_B Chan1 = 001_B</p>
CH_SUSP	8	rw	<p>Channel Suspend</p> <p>Suspends all DMA data transfers from the source until this bit is cleared. There is no guarantee that the current transaction will complete. Can also be used in conjunction with CFGLx.FIFO_EMPTY to cleanly disable a channel without losing any data.</p> <p>0_B Not suspended. 1_B Suspend DMA transfer from the source.</p>
FIFO_EMPTY	9	r	<p>Indicates if there is data left in the channel FIFO</p> <p>Can be used in conjunction with CFGLx.CH_SUSP to cleanly disable a channel.</p> <p>1_B Channel FIFO empty 0_B Channel FIFO not empty</p>
HS_SEL_DST	10	rw	<p>Destination Software or Hardware Handshaking Select</p> <p>This register selects which of the handshaking interfaces - hardware or software - is active for destination requests on this channel.</p> <p>0_B Hardware handshaking interface. Software-initiated transaction requests are ignored. 1_B Software handshaking interface. Hardware-initiated transaction requests are ignored.</p> <p>If the destination peripheral is memory, then this bit is ignored.</p>

General Purpose DMA (GPDMA)

Field	Bits	Type	Description
HS_SEL_SRC	11	rw	<p>Source Software or Hardware Handshaking Select</p> <p>This register selects which of the handshaking interfaces - hardware or software - is active for source requests on this channel.</p> <p>0_B Hardware handshaking interface. Software-initiated transaction requests are ignored.</p> <p>1_B Software handshaking interface. Hardware-initiated transaction requests are ignored.</p> <p>If the source peripheral is memory, then this bit is ignored.</p>
LOCK_CH_L	[13:12]	rw	<p>Channel Lock Level</p> <p>Indicates the duration over which CFGLx.LOCK_CH bit applies.</p> <p>00_B Over complete DMA transfer</p> <p>01_B Over complete DMA block transfer</p> <p>1x_B Over complete DMA transaction</p>
LOCK_B_L	[15:14]	rw	<p>Bus Lock Level</p> <p>Indicates the duration over which CFGLx.LOCK_B bit applies.</p> <p>00_B Over complete DMA transfer</p> <p>01_B Over complete DMA block transfer</p> <p>1x_B Over complete DMA transaction</p>
LOCK_CH	16	rw	<p>Channel Lock Bit</p> <p>When the channel is granted control of the master bus interface and if the CFGLx.LOCK_CH bit is asserted, then no other channels are granted control of the master bus interface for the duration specified in CFGLx.LOCK_CH_L. Indicates to the master bus interface arbiter that this channel wants exclusive access to the master bus interface for the duration specified in CFGLx.LOCK_CH_L.</p>
LOCK_B	17	rw	<p>Bus Lock Bit</p> <p>When active, the AHB bus master signal hlock is asserted for the duration specified in CFGLx.LOCK_B_L. For more information, refer to Section 5.2.6.</p>

General Purpose DMA (GPDMA)

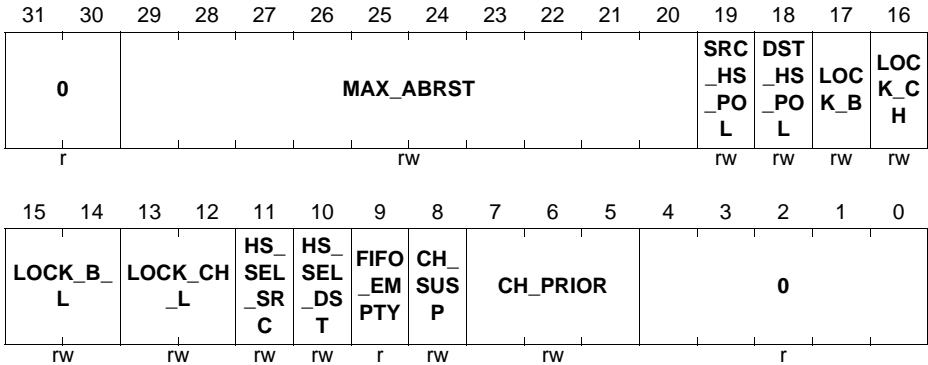
Field	Bits	Type	Description
DST_HS_POL	18	rw	Destination Handshaking Interface Polarity 0 _B Active high 1 _B Active low For information on this, refer to Section 5.2.4 .
SRC_HS_POL	19	rw	Source Handshaking Interface Polarity 0 _B Active high 1 _B Active low For information on this, refer to Section 5.2.4 .
MAX_ABRST	[29:20]	rw	Maximum AMBA Burst Length Maximum AMBA burst length that is used for DMA transfers on this channel. A value of 0 indicates that software is not limiting the maximum AMBA burst length for DMA transfers on this channel.
RELOAD_SRC	30	rw	Automatic Source Reload The SAR register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated. For conditions under which this occurs, refer to Table 5-5 .
RELOAD_DST	31	rw	Automatic Destination Reload The DAR register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated. For conditions under which this occurs, refer to Table 5-5 .
0	[4:0]	r	Reserved

GPDMA0_CHx_CFGL (x=2-7)

Configuration Register Low for Channel x

$$(40_H + x * 58_H)$$

Reset Value: 0000 0EX0_H



Field	Bits	Type	Description
CH_PRIOR	[7:5]	rw	<p>Channel priority</p> <p>A priority of 7 is the highest priority, and 0 is the lowest. The value programmed to this field must be within 0 and 7. A programmed value outside this range will cause erroneous behavior.</p> <p>Reset: Channel Number</p> <p>For example: Chan0 = 000_B Chan1 = 001_B</p>
CH_SUSP	8	rw	<p>Channel Suspend</p> <p>Suspends all DMA data transfers from the source until this bit is cleared. There is no guarantee that the current transaction will complete. Can also be used in conjunction with CFGLx.FIFO_EMPTY to cleanly disable a channel without losing any data.</p> <p>0_B Not suspended. 1_B Suspend DMA transfer from the source.</p>
FIFO_EMPTY	9	r	<p>Indicates if there is data left in the channel FIFO</p> <p>Can be used in conjunction with CFGLx.CH_SUSP to cleanly disable a channel.</p> <p>1_B Channel FIFO empty 0_B Channel FIFO not empty</p>

General Purpose DMA (GPDMA)

Field	Bits	Type	Description
HS_SEL_DST	10	rw	<p>Destination Software or Hardware Handshaking Select</p> <p>This register selects which of the handshaking interfaces - hardware or software - is active for destination requests on this channel.</p> <p>0_B Hardware handshaking interface. Software-initiated transaction requests are ignored.</p> <p>1_B Software handshaking interface. Hardware-initiated transaction requests are ignored.</p> <p>If the destination peripheral is memory, then this bit is ignored.</p>
HS_SEL_SRC	11	rw	<p>Source Software or Hardware Handshaking Select</p> <p>This register selects which of the handshaking interfaces - hardware or software - is active for source requests on this channel.</p> <p>0_B Hardware handshaking interface. Software-initiated transaction requests are ignored.</p> <p>1_B Software handshaking interface. Hardware-initiated transaction requests are ignored.</p> <p>If the source peripheral is memory, then this bit is ignored.</p>
LOCK_CH_L	[13:12]	rw	<p>Channel Lock Level</p> <p>Indicates the duration over which CFGLx.LOCK_CH bit applies.</p> <p>00_B Over complete DMA transfer</p> <p>01_B Over complete DMA block transfer</p> <p>1x_B Over complete DMA transaction</p>
LOCK_B_L	[15:14]	rw	<p>Bus Lock Level</p> <p>Indicates the duration over which CFGLx.LOCK_B bit applies.</p> <p>00_B Over complete DMA transfer</p> <p>01_B Over complete DMA block transfer</p> <p>1x_B Over complete DMA transaction</p>

General Purpose DMA (GPDMA)

Field	Bits	Type	Description
LOCK_CH	16	rw	Channel Lock Bit When the channel is granted control of the master bus interface and if the CFGLx.LOCK_CH bit is asserted, then no other channels are granted control of the master bus interface for the duration specified in CFGLx.LOCK_CH_L. Indicates to the master bus interface arbiter that this channel wants exclusive access to the master bus interface for the duration specified in CFGLx.LOCK_CH_L.
LOCK_B	17	rw	Bus Lock Bit When active, the AHB bus master signal hlock is asserted for the duration specified in CFGLx.LOCK_B_L. For more information, refer to Section 5.2.6 .
DST_HS_POL	18	rw	Destination Handshaking Interface Polarity 0 _B Active high 1 _B Active low For information on this, refer to Section 5.2.4 .
SRC_HS_POL	19	rw	Source Handshaking Interface Polarity 0 _B Active high 1 _B Active low For information on this, refer to Section 5.2.4 .
MAX_ABRST	[29:20]	rw	Maximum AMBA Burst Length Maximum AMBA burst length that is used for DMA transfers on this channel. A value of 0 indicates that software is not limiting the maximum AMBA burst length for DMA transfers on this channel.
0	[31:30], [4:0]	r	Reserved

Table 5-11 PROTCTL field to HPROT Mapping

1 _B	HPROT[0]
CFGHx.PROTCTL[1]	HPROT[1]
CFGHx.PROTCTL[2]	HPROT[2]
CFGHx.PROTCTL[3]	HPROT[3]

SGR

The Source Gather register contains two fields:

- Source gather count field (SGRx.SGC) - Specifies the number of contiguous source transfers of **CTL.SRC_TR_WIDTH** between successive gather intervals. This is defined as a gather boundary.
- Source gather interval field (SGRx.SGI) - Specifies the source address increment/decrement in multiples of **CTL.SRC_TR_WIDTH** on a gather boundary when gather mode is enabled for the source transfer.

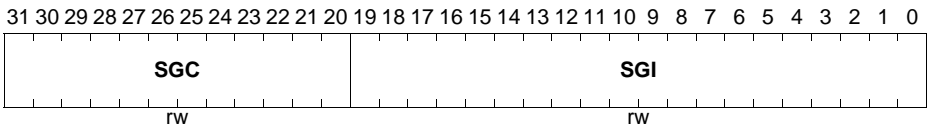
The **CTL.SINC** field controls whether the address increments or decrements. When the **CTL.SINC** field indicates a fixed-address control, then the address remains constant throughout the transfer and the SGR register is ignored. For more information, see [Section 5.2.7](#).

GPDMA0_CHx_SGR (x=0-1)

Source Gather Register for Channel x

(48_H + x*58_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SGI	[19:0]	rw	Source gather interval
SGC	[31:20]	rw	Source gather count Source contiguous transfer count between successive gather boundaries.

DSR

The Destination Scatter register contains two fields:

- Destination scatter count field (DSRx.DSC) - Specifies the number of contiguous destination transfers of **CTL.DST_TR_WIDTH** between successive scatter boundaries.
- Destination scatter interval field (DSRx.DSI) - Specifies the destination address increment/decrement in multiples of **CTL.DST_TR_WIDTH** on a scatter boundary when scatter mode is enabled for the destination transfer.

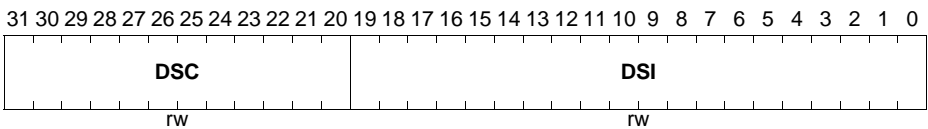
The **CTL.DINC** field controls whether the address increments or decrements. When the **CTL.DINC** field indicates a fixed address control, then the address remains constant throughout the transfer and the DSR register is ignored. For more information, see [Section 5.2.7](#).

GPDMA0_CHx_DSR (x=0-1)

Destination Scatter Register for Channel x

$$(50_H + x*58_H)$$

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DSI	[19:0]	rw	Destination scatter interval
DSC	[31:20]	rw	Destination scatter count Destination contiguous transfer count between successive scatter boundaries.

5.8.3 Interrupt Registers

The following sections describe the registers pertaining to interrupts, their status, and how to clear them. For each channel, there are five types of interrupt sources:

- **IntBlock** - Block Transfer Complete Interrupt. This interrupt is generated on DMA block transfer completion to the destination peripheral.
- **IntDstTran** - Destination Transaction Complete Interrupt
This interrupt is generated after completion of the last AHB transfer of the requested single/burst transaction from the handshaking interface (either the hardware or software handshaking interface) on the destination side.

Note: If the destination for a channel is memory, then that channel will never generate the IntDstTran interrupt. Because of this, the corresponding bit in this field will not be set.

- **IntErr** - Error Interrupt
This interrupt is generated when an ERROR response is received from an AHB slave on the HRESP bus during a DMA transfer. In addition, the DMA transfer is cancelled and the channel is disabled.
- **IntSrcTran** - Source Transaction Complete Interrupt
This interrupt is generated after completion of the last AHB transfer of the requested single/burst transaction from the handshaking interface (either the hardware or software handshaking interface) on the source side.

Note: If the source or destination is memory, then IntSrcTran/IntDstTran interrupts should be ignored, as there is no concept of a "DMA transaction level" for memory.

- **IntTfr** - DMA Transfer Complete Interrupt
This interrupt is generated on DMA transfer completion to the destination peripheral.

There are several groups of interrupt-related registers:

- **Interrupt Raw Status Registers**
- **Interrupt Status Registers**
- **Interrupt Mask Registers**
- **Interrupt Clear Registers**
- **Combined Interrupt Status Register**

When a channel has been enabled to generate interrupts, the following is true:

- Interrupt events are stored in the Raw Status registers.
- The contents of the Raw Status registers are masked with the contents of the Mask registers.
- The masked interrupts are stored in the Status registers.
- The contents of the Status registers are used to drive the int_* port signals.
- Writing to the appropriate bit in the Clear registers clears an interrupt in the Raw Status registers and the Status registers on the same clock cycle.

General Purpose DMA (GPDMA)

The contents of each of the five Status registers is ORed to produce a single bit for each interrupt type in the Combined Status register; that is, STATUSINT.

Note: For interrupts to propagate past the raw interrupt register stage, **CTL.INT_EN** must be set to 1_B, and the relevant interrupt must be unmasked in the mask* interrupt register.*

Interrupt Raw Status Registers

Interrupt events are stored in these Raw Interrupt Status registers before masking: RAWBLOCK, RawDstTran, RawErr, RawSrcTran, and RAWTFR. Each Raw Interrupt Status register has a bit allocated per channel; for example, RAWTFR[2] is the Channel 2 raw transfer complete interrupt.

Each bit in these registers is cleared by writing a 1 to the corresponding location in the CLEARTRF, CLEARBLOCK, CLEARSRCTRAN, CLEARDSTTRAN, CLEARERR registers.

Note: Write access is available to these registers for software testing purposes only. Under normal operation, writes to these registers are not recommended.

RAWTFR

Raw DMA Transfer Complete Interrupt Status.

RAWBLOCK

Raw Block Transfer Complete Interrupt Status.

RAWSRCTRAN

Raw Source Transaction Complete Interrupt Status.

RAWDSTTRAN

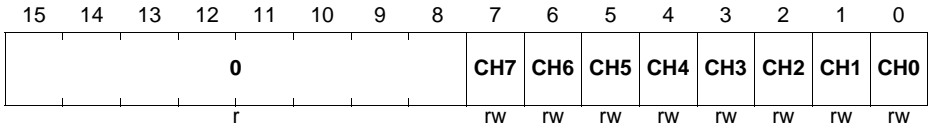
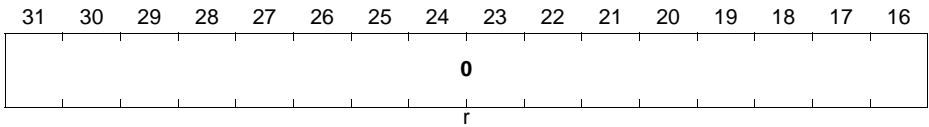
Raw Destination Transaction Complete Interrupt Status.

RAWERR

Raw Error Interrupt Status.

General Purpose DMA (GPDMA)

GPDMA0_RAWTFR		
Raw IntTfr Status	(2C0 _H)	Reset Value: 0000 0000 _H
GPDMA0_RAWBLOCK		
Raw IntBlock Status	(2C8 _H)	Reset Value: 0000 0000 _H
GPDMA0_RAWSRCTRAN		
Raw IntSrcTran Status	(2D0 _H)	Reset Value: 0000 0000 _H
GPDMA0_RAWDSTTRAN		
Raw IntBlock Status	(2D8 _H)	Reset Value: 0000 0000 _H
GPDMA0_RAWERR		
Raw IntErr Status	(2E0 _H)	Reset Value: 0000 0000 _H



Field	Bits	Type	Description
CHx (x=0-7)	x	rw	Raw Interrupt Status for channel x
0	[31:8]	r	Reserved

Interrupt Status Registers

All interrupt events from all channels are stored in these Interrupt Status registers after masking: STATUSBLOCK, STATUSDSTTRAN, STATUSERR, STATUSSRCTRAN, and STATUSTFR. Each Interrupt Status register has a bit allocated per channel; for example, STATUSTFR[2] is the Channel 2 status transfer complete interrupt. The contents of these registers are used to generate the interrupt signals (int or int_n bus, depending on interrupt polarity) leaving the GPDMA.

STATUSTFR

DMA Transfer Complete Interrupt Status.

STATUSBLOCK

Block Transfer Complete Interrupt Status.

STATUSSRCTRAN

Source Transaction Complete Interrupt Status.

STATUSDSTTRAN

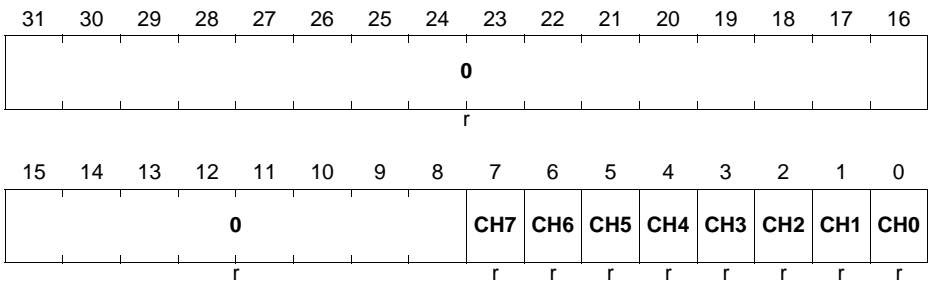
Block Transfer Complete Interrupt Status.

STATUSERR

Error Interrupt Status.

General Purpose DMA (GPDMA)

GPDMA0_STATUSTFR		
IntTfr Status	(2E8 _H)	Reset Value: 0000 0000 _H
GPDMA0_STATUSBLOCK		
IntBlock Status	(2F0 _H)	Reset Value: 0000 0000 _H
GPDMA0_STATUSSRCTRAN		
IntSrcTran Status	(2F8 _H)	Reset Value: 0000 0000 _H
GPDMA0_STATUSDSTTRAN		
IntBlock Status	(300 _H)	Reset Value: 0000 0000 _H
GPDMA0_STATUSERR		
IntErr Status	(308 _H)	Reset Value: 0000 0000 _H



Field	Bits	Type	Description
CHx (x=0-7)	x	r	Interrupt Status for channel x
0	[31:8]	r	Reserved

Interrupt Mask Registers

The contents of the Raw Status registers are masked with the contents of the Mask registers: MASKBLOCK, MASKDSTTRAN, MASKERR, MASKSRCTRAN, and MASKTFR. Each Interrupt Mask register has a bit allocated per channel; for example, MASKTFR[2] is the mask bit for the Channel 2 transfer complete interrupt.

When the source peripheral of DMA channel *i* is memory, then the source transaction complete interrupt, MASKSRCTRAN[*z*], must be masked to prevent an erroneous triggering of an interrupt on the int_combined signal. Similarly, when the destination peripheral of DMA channel *i* is memory, then the destination transaction complete interrupt, MASKDSTTRAN[*i*], must be masked to prevent an erroneous triggering of an interrupt on the int_combined(_n) signal.

A channel INT_MASK bit will be written only if the corresponding mask write enable bit in the INT_MASK_WE field is asserted on the same AHB write transfer. This allows software to set a mask bit without performing a read-modified write operation. For example, writing hex 01x1 to the MASKTFR register writes a 1 into MASKTFR[0], while MASKTFR[7:1] remains unchanged. Writing hex 00xx leaves MASKTFR[7:0] unchanged.

Writing a 1 to any bit in these registers un masks the corresponding interrupt, thus allowing the GPDMA to set the appropriate bit in the Status registers and int_* port signals.

MASKTFR

Mask for Raw DMA Transfer Complete Interrupt Status.

MASKBLOCK

Mask for Raw Block Transfer Complete Interrupt Status.

MASKSRCTRAN

Mask for Raw Source Transaction Complete Interrupt Status.

MASKDSTTRAN

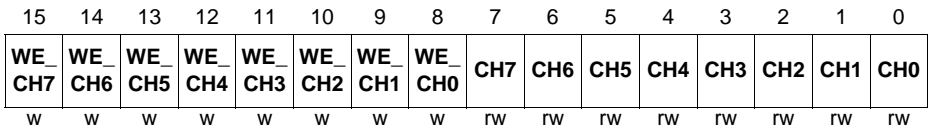
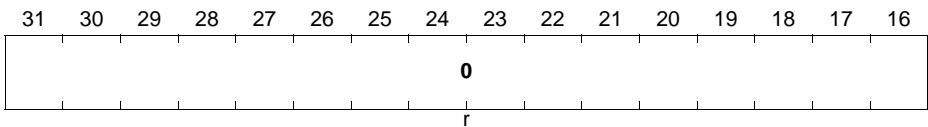
Mask for Raw Block Transfer Complete Interrupt Status.

MASKERR

Mask for Raw Error Interrupt Status.

General Purpose DMA (GPDMA)

GPDMA0_MASKTFR		
Mask for Raw IntTfr Status	(310 _H)	Reset Value: 0000 0000 _H
GPDMA0_MASKBLOCK		
Mask for Raw IntBlock Status	(318 _H)	Reset Value: 0000 0000 _H
GPDMA0_MASKSRCTRAN		
Mask for Raw IntSrcTran Status	(320 _H)	Reset Value: 0000 0000 _H
GPDMA0_MASKDSTTRAN		
Mask for Raw IntBlock Status	(328 _H)	Reset Value: 0000 0000 _H
GPDMA0_MASKERR		
Mask for Raw IntErr Status	(330 _H)	Reset Value: 0000 0000 _H



Field	Bits	Type	Description
CHx (x=0-7)	x	rW	Mask bit for channel x 0 _B masked 1 _B unmasked
WE_CHx (x=0-7)	8+x	w	Write enable for mask bit of channel x 0 _B write disabled 1 _B write enabled
0	[31:16]	r	Reserved

Interrupt Clear Registers

Each bit in the Raw Status and Status registers is cleared on the same cycle by writing a 1 to the corresponding location in the Clear registers: CLEARBLOCK, CLEARSTTRAN, CLEARERR, CLEARSRCTRAN, and CLEARTFR. Each Interrupt Clear register has a bit allocated per channel; for example, CLEARTFR[2] is the clear bit for the Channel 2 transfer complete interrupt. Writing a 0 has no effect. These registers are not readable.

CLEARTFR

Clear DMA Transfer Complete Interrupt Status and Raw Status.

CLEARBLOCK

Clear Block Transfer Complete Interrupt Status and Raw Status.

CLEARSRCTRAN

Clear Source Transaction Complete Interrupt Status and Raw Status.

CLEARSTTRAN

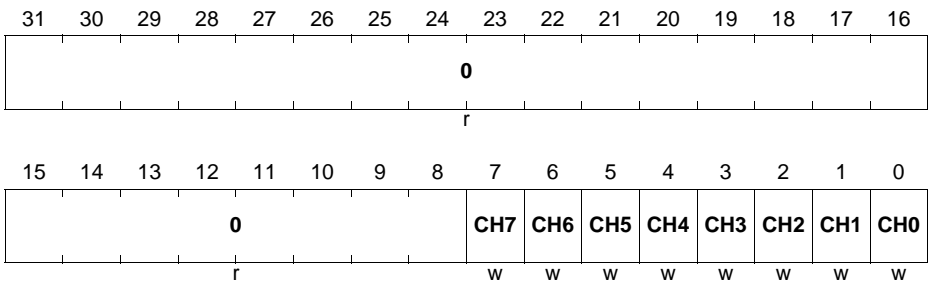
Clear Block Transfer Complete Interrupt Status and Raw Status.

CLEARERR

Clear Error Interrupt Status and Raw Status.

General Purpose DMA (GPDMA)

GPDMA0_CLEARTRF		
IntTfr Status	(338 _H)	Reset Value: 0000 0000 _H
GPDMA0_CLEARBLOCK		
IntBlock Status	(340 _H)	Reset Value: 0000 0000 _H
GPDMA0_CLEARSRCTRAN		
IntSrcTran Status	(348 _H)	Reset Value: 0000 0000 _H
GPDMA0_CLEARSTTRAN		
IntBlock Status	(350 _H)	Reset Value: 0000 0000 _H
GPDMA0_CLEARERR		
IntErr Status	(358 _H)	Reset Value: 0000 0000 _H



Field	Bits	Type	Description
CHx (x=0-7)	x	w	Clear Interrupt Status and Raw Status for channel x 0 _B no effect 1 _B clear status
0	[31:8]	r	Reserved

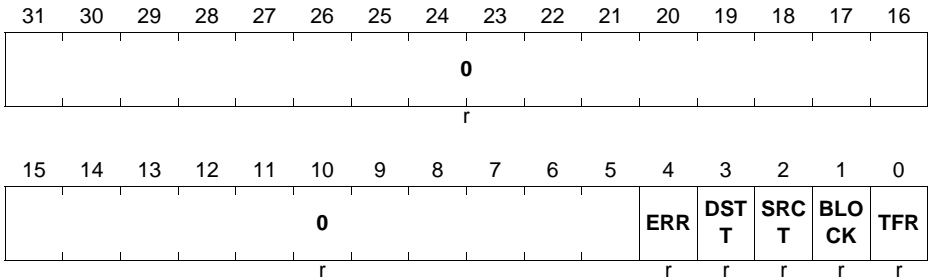
Combined Interrupt Status Register

The contents of each of the five Status registers - STATUSTFR, STATUSBLOCK, STATUSSRCTRAN, STATUSDSTTRAN, STATUSERR - is ORed to produce a single bit for each interrupt type in the Combined Status register (STATUSINT). This register is read-only.

GPDMA0_STATUSINT

Combined Interrupt Status Register (360_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TFR	0	r	OR of the contents of STATUSTFR register
BLOCK	1	r	OR of the contents of STATUSBLOCK register
SRCT	2	r	OR of the contents of STATUSSRCTRAN register
DSTT	3	r	OR of the contents of STATUSDSTTRAN register
ERR	4	r	OR of the contents of STATUSERR register
0	[31:5]	r	Reserved

5.8.4 Software Handshaking Registers

The registers that comprise the software handshaking registers allow software to initiate single or burst transaction requests in the same way that handshaking interface signals do in hardware.

Setting **CFG.HS_SEL_SRC** to 1 enables software handshaking on the source of channel x. Setting **CFG.HS_SEL_DST** to 1 enables software handshaking on the destination of channel x.

REQSRCREG

A bit is assigned for each channel in this register. **REQSRCREG[n]** is ignored when software handshaking is not enabled for the source of channel n.

A channel **SRC_REQ** bit is written only if the corresponding channel write enable bit in the **SRC_REQ_WE** field is asserted on the same AHB write transfer, and if the channel is enabled in the **CHENREG** register. For example, writing hex 0101 writes a 1 into **REQSRCREG[0]**, while **REQSRCREG[7:1]** remains unchanged. Writing hex 00xx leaves **REQSRCREG[7:0]** unchanged. This allows software to set a bit in the **REQSRCREG** register without performing a read-modified write operation.

The functionality of this register depends on whether the source is a flow control peripheral or not.

GPDMA0_REQSRCREG

Source Software Transaction Request Register

(368_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WE_	WE_	WE_	WE_	WE_	WE_	WE_	WE_	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
w	w	w	w	w	w	w	w	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CHx (x=0-7)	x	rw	Source request for channel x
WE_CHx (x=0-7)	8+x	w	Source request write enable for channel x 0 _B write disabled 1 _B write enabled
0	[31:16]	r	Reserved

REQDSTREG

A bit is assigned for each channel in this register. REQDSTREG[n] is ignored when software handshaking is not enabled for the source of channel n.

A channel DST_REQ bit is written only if the corresponding channel write enable bit in the DST_REQ_WE field is asserted on the same AHB write transfer, and if the channel is enabled in the **CHENREG** register.

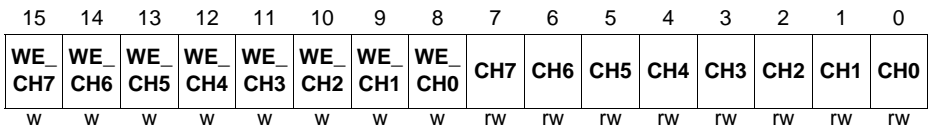
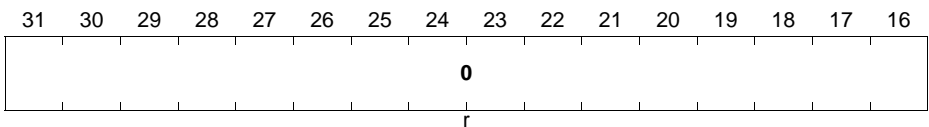
The functionality of this register depends on whether the destination is a flow control peripheral or not.

GPDMA0_REQDSTREG

Destination Software Transaction Request Register

(370_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CHx (x=0-7)	x	rw	Source request for channel x
WE_CHx (x=0-7)	8+x	w	Source request write enable for channel x 0 _B write disabled 1 _B write enabled
0	[31:16]	r	Reserved

General Purpose DMA (GPDMA)

SGLREQSRCREG

A bit is assigned for each channel in this register. SGLREQSRCREG[n] is ignored when software handshaking is not enabled for the source of channel n.

A channel SRC_SGLREQ bit is written only if the corresponding channel write enable bit in the SRC_SGLREQ_WE field is asserted on the same AHB write transfer, and if the channel is enabled in the **CHENREG** register.

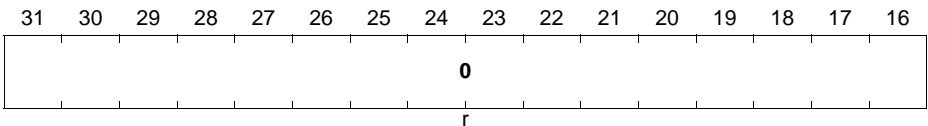
The functionality of this register depends on whether the source is a flow control peripheral or not.

GPDMA0_SGLREQSRCREG

Single Source Transaction Request Register

(378_H)

Reset Value: 0000 0000_H



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WE_ CH7	WE_ CH6	WE_ CH5	WE_ CH4	WE_ CH3	WE_ CH2	WE_ CH1	WE_ CH0	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
w	w	w	w	w	w	w	w	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CHx (x=0-7)	x	rw	Source request for channel x
WE_CHx (x=0-7)	8+x	w	Source request write enable for channel x 0 _B write disabled 1 _B write enabled
0	[31:16]	r	Reserved

SGLREQDSTREG

A bit is assigned for each channel in this register. SGLREQDSTREG[n] is ignored when software handshaking is not enabled for the destination of channel n.

A channel DST_SGLREQ bit is written only if the corresponding channel write enable bit in the DST_SGLREQ_WE field is asserted on the same AHB write transfer, and if the channel is enabled in the **CHENREG** register.

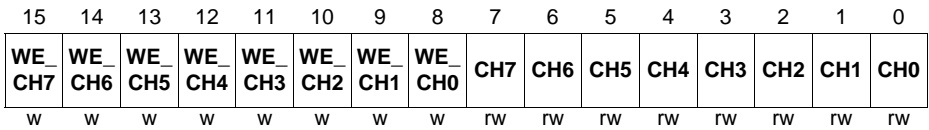
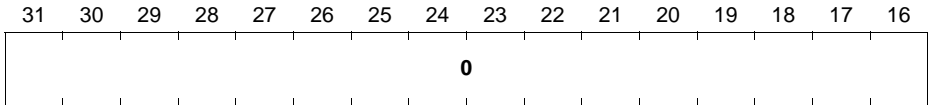
The functionality of this register depends on whether the destination is a flow control peripheral or not.

GPDMA0_SGLREQDSTREG

Single Destination Transaction Request Register

(380_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CHx (x=0-7)	x	rw	Source request for channel x
WE_CHx (x=0-7)	8+x	w	Source request write enable for channel x 0 _B write disabled 1 _B write enabled
0	[31:16]	r	Reserved

LSTSRCREG

A bit is assigned for each channel in this register. LSTSRCREG[n] is ignored when software handshaking is not enabled for the source of channel n, or when the source of channel n is not a flow controller.

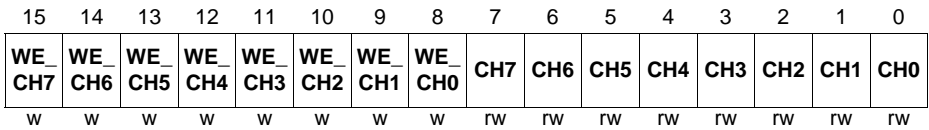
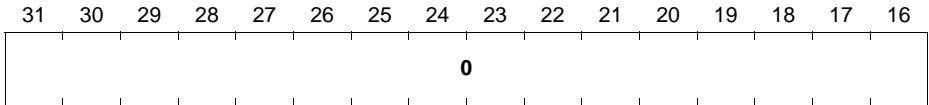
A channel LSTSRC bit is written only if the corresponding channel write enable bit in the LSTSRC_WE field is asserted on the same AHB write transfer, and if the channel is enabled in the **CHENREG** register.

GPDMA0_LSTSRCREG

Last Source Transaction Request Register

(388_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CHx (x=0-7)	x	rw	Source last request for channel x 0 _B Not last transaction in current block 1 _B Last transaction in current block
WE_CHx (x=0-7)	8+x	w	Source last transaction request write enable for channel x 0 _B write disabled 1 _B write enabled
0	[31:16]	r	Reserved

LSTDSTREG

A bit is assigned for each channel in this register. LSTDSTREG[n] is ignored when software handshaking is not enabled for the destination of channel n or when the destination of channel n is not a flow controller.

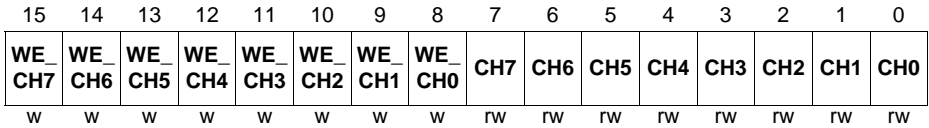
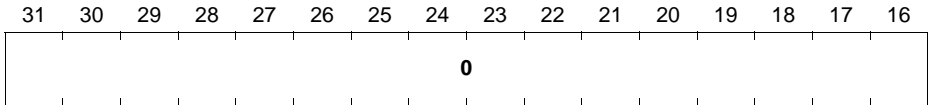
A channel LSTDST bit is written only if the corresponding channel write enable bit in the LSTDST_WE field is asserted on the same AHB write transfer, and if the channel is enabled in the **CHENREG** register.

GPDMA0_LSTDSTREG

Last Destination Transaction Request Register

(390_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CHx (x=0-7)	x	rw	Destination last request for channel x 0 _B Not last transaction in current block 1 _B Last transaction in current block
WE_CHx (x=0-7)	8+x	w	Destination last transaction request write enable for channel x 0 _B write disabled 1 _B write enabled
0	[31:16]	r	Reserved

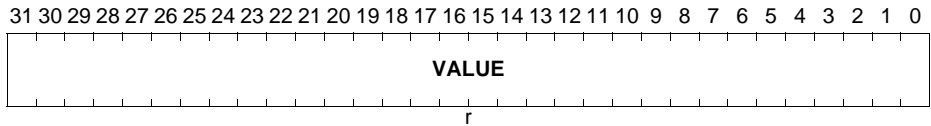
5.8.5 Miscellaneous GPDMA Registers

ID

This is the GPDMA ID register, which is a read-only register that reads back the hardcoded module ID number.

GPDMA0_ID

GPDMA0 ID Register (3A8_H) **Reset Value: 00AF C0XX_H**



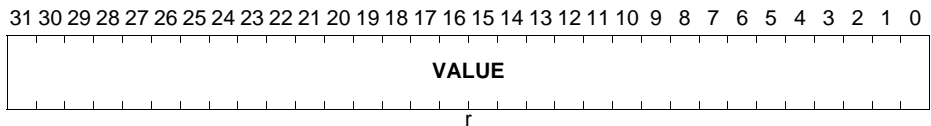
Field	Bits	Type	Description
VALUE	[31:0]	r	Hardcoded GPDMA Peripheral ID

TYPE

This is the GPDMA Component Type register, which is a read-only register that specifies the type of the packaged component.

GPDMA0_TYPE

GPDMA Component Type (3F8_H) **Reset Value: 4457 1110_H**



Field	Bits	Type	Description
VALUE	[31:0]	r	Component Type number = 44_57_11_10.

VERSION

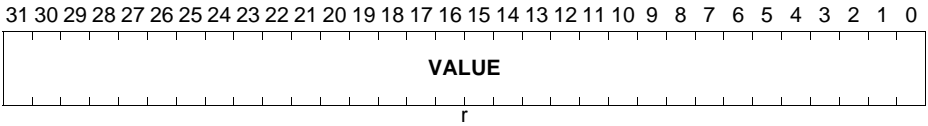
This is the GPDMA Component Version register, which is a read-only register that specifies the version of the packaged component.

GPDMA0_VERSION

DMA Component Version

(3FC_H)

Reset Value: 3231 342A_H



Field	Bits	Type	Description
VALUE	[31:0]	r	Version number of the component

6 Flexible CRC Engine (FCE)

The FCE provides a parallel implementation of Cyclic Redundancy Code (CRC) algorithms. The current FCE version for the XMC4300 microcontroller implements the IEEE 802.3 ethernet CRC32, the CCITT CRC16 and the SAE J1850 CRC8 polynomials. The primary target of FCE is to be used as an hardware acceleration engine for software applications or operating systems services using CRC signatures.

The FCE operates as a standard peripheral bus slave and is fully controlled through a set of configuration and control registers. The different CRC algorithms are independent from each other, they can be used concurrently by different software tasks.

Note: The FCE kernel register names described in “Registers” on Page 6-11 are referenced in a product Reference Manual by the module name prefix “FCE_”.

Input documents

- [5] A painless guide to CRC Error Detection Algorithms, Ross N. Williams
- [6] 32-Bit Cyclic Redundancy Codes for Internet Applications, Philip Koopman, International Conference on Dependable Systems and Networks (DSN), 2002

Related standards and norms

- [7] IEEE 802.3 Ethernet 32-bits CRC

Table 6-1 FCE Abbreviations

CRC	Cyclic Redundancy Checksum
FCE	Flexible CRC Engine
IR	Input Register
RES	Result
STS	Status
CFG	Configuration

6.1 Overview

This section provides an overview of the features, applications and architecture of the FCE module.

6.1.1 Features

The FCE provides the following features:

- The FCE implements the following CRC polynomials:

Flexible CRC Engine (FCE)

- CRC kernel 0 and 1: IEEE 802.3 CRC32 ethernet polynomial: $0x04C11DB7^{(1)} - x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- CRC kernel 2: CCITT CRC16 polynomial: $0x1021 - x^{16} + x^{12} + x^5 + 1$
- CRC kernel 3: SAE J1850 CRC8 polynomial: $0x1D - x^8 + x^4 + x^3 + x^2 + 1$
- Parallel CRC implementation
 - Data blocks to be computed by FCE shall be a multiple of the polynomial degree
 - Start address of Data blocks to be computed by FCE shall be aligned to the polynomial degree
- Register Interface:
 - Input Register
 - CRC Register
 - Configuration Registers enabling to control the CRC operation and perform automatic checksum checks at the end of a message.
 - Extended register interface to control reliability of FCE execution in safety applications.
- Error notification scheme via dedicated interrupt node for:
 - Transient error detection: error interrupt generation (maskable) with local status register (cleared by software)
 - Checksum failure: error interrupt generation (maskable) with local status register (cleared by software)
- FCE provides one interrupt line to the interrupt system. Each CRC engine has its own set of flag registers.

6.1.2 Application Mapping

Among other applications, CRC algorithms are commonly used to calculate message signatures to:

- Check message integrity during transport over communication channels like internal buses or interfaces between microcontrollers
- Sign blocks of data residing in variable or invariable storage elements
- Compute signatures for program flow monitoring

One important property to be taken into account by the application when choosing a polynomial is the hamming distance: see [Section 6.9](#).

6.1.3 Block Diagram

The FCE is a standard peripheral slave module which is controlled over a set of memory mapped registers. The FCE is fully synchronous with the CPU clock and runs with a 1:1 clock ratio.

1) The polynomial hexadecimal representation covers the coefficients (degree - 1) down to 0.

Flexible CRC Engine (FCE)

Depending on the hardware configuration the FCE may implement more CRC kernels with different CRC polynomials. The specific configuration for the XMC4300 microcontroller is shown in the **Figure 6-1 “FCE Block Diagram” on Page 6-3**.

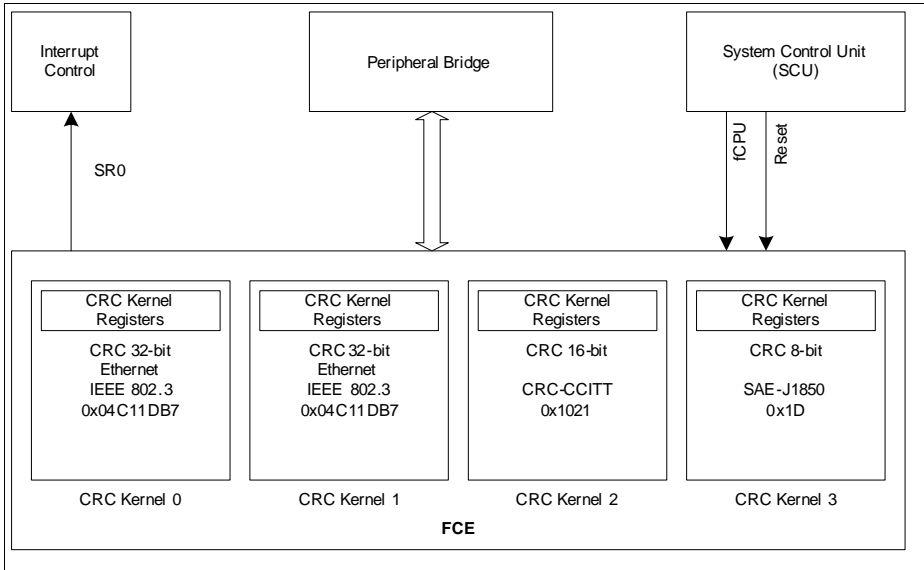


Figure 6-1 FCE Block Diagram

Every CRC kernel will present the same hardware and software architecture. The rest of this document will focus only on the description of the generic CRC kernel architecture. In a multi-kernel implementation the interrupt lines are ored together, the FCE only presents a single interrupt node to the system. Each CRC kernel implements a status register that enables the software to identify which interrupt source is active. Please refer to the **STSm (m = 0-3)** register for a detailed description of the status and interrupt handling.

6.2 Functional Description

A checksum algorithm based on CRC polynomial division is characterized by the following properties:

1. polynomial degree (e.g. 32, that represents the highest power of two of the polynomial)
2. polynomial (e.g. 0x04C11DB7: the 33rd bit is omitted because always equal to 1)
3. init value: the initial value of the CRC register

Flexible CRC Engine (FCE)

4. input data reflected: indicates if each byte of the input parallel data is reflected before being used to compute the CRC
5. result data reflected: indicates if the final CRC value is reflected or not
6. XOR value: indicates if a final XOR operation is done before returning the CRC result

All the properties are fixed once a polynomial has been chosen. However the FCE provides the capability to control the two reflection steps and the final XOR through the CFG register. The reset values are compatible with the implemented algorithm. The final XOR control enables to select either 0xFFFFFFFF or 0x00000000 to be XORed with the POST_CRC1 value. These two values are those used by the most common CRC polynomials.

Note: The reflection steps and final XOR do not modify the properties of the CRC algorithm in terms of error detection, only the CRC final signature is affected.

The next two figures provides an overview of the control and status features of a CRC kernel.

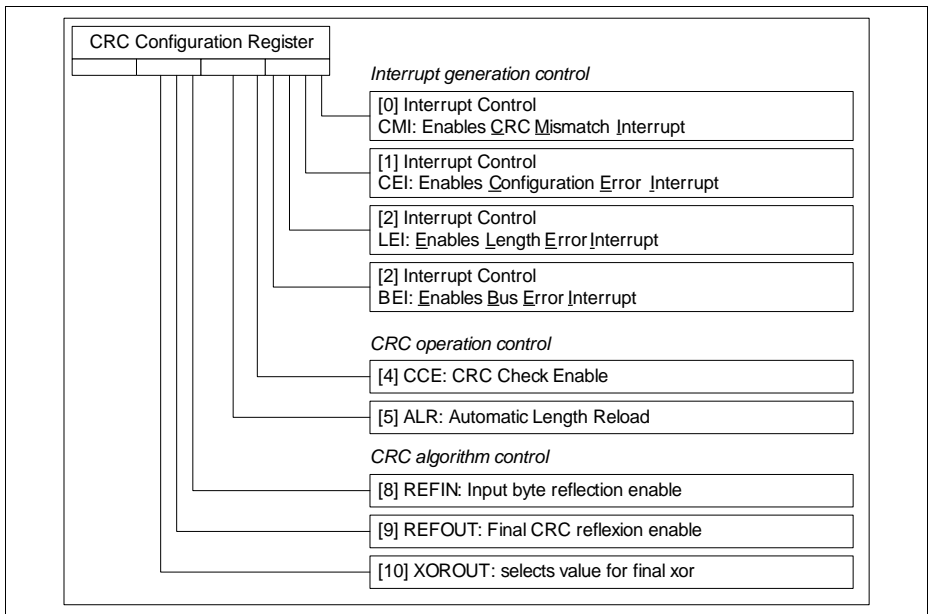


Figure 6-2 CRC kernel configuration register

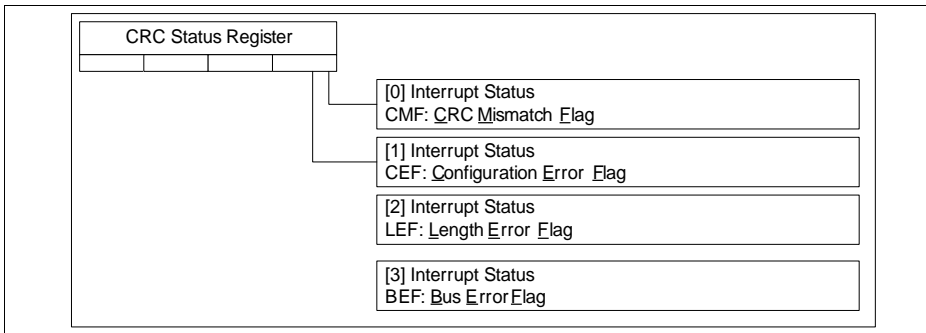


Figure 6-3 CRC kernel status register

6.2.1 Basic Operation

The software must first ensure that the CRC kernel is properly configured, especially the initial CRC value written via the **CRC** register. Then, it writes as many times as necessary into the **IR** register according to the length of the message. The resulting signature is stored in the CRC engine result register, **RESm**, which can be read by the software.

Depending on the CRC kernel accesses by software the following rules apply:

- When accessing a CRC kernel of degree $\langle N \rangle$ only the bits $N-1$ down to 0 are used by the CRC kernel. The upper bits are ignored on write. When reading from a CRC kernel register the non-used upper bits are set to 0.

6.2.2 Automatic Signature Check

The automatic signature check compares the signature at the end of a message with the expected signature configured in the CHECK register. In case of a mismatch, an event is generated (see [Section 6.3](#)). This feature is enabled by the CFG.CCE bit field (see [CFGm \(m = 0-3\)](#) register).

If the software wishes to use this feature, the LENGTH register and CHECK registers must be configured with respectively the length as number of words of the message and the expected signature (CHECK). The word length is defined by the degree of the polynomial used. The CHECK value takes into account the final CRC reflection and XOR operation.

When the **CFG.CCE** bit field is set, every time the IR register is written, the LENGTH register is decremented by one until it reaches zero. The hardware monitors the transition of the LENGTH register from 1 to 0 to detect the end of the message and proceed with the comparison of the result register RESvalue with the CHECK register value. If the automatic length reload feature is enabled by the CFG.ALR bit field (see

Flexible CRC Engine (FCE)

CFGm (m = 0-3)), the LENGTH register is reinitialized with the previously configured value. This feature is especially suited when the FCE is used in combination with a DMA engine.

In the case the automatic length reload feature is not enabled, if LENGTH is already at zero but software still writes to IR (by mistake) every bit of the LENGTH should be set to 1 and hold this value until software initializes it again for the processing of a new message. In such case the STS.LEF (Length Error Flag) should be set and an interrupt generated if the CFG.LEI (Length Error Interrupt) is set.

Usually, the CRC signature of a message M0 is computed and appended to M0 to form the message M1 which is transmitted. One interesting property of CRCs is that the CRC signature of M1 shall be zero. This property is particularly useful when automatically checking the signature of data blocks of fixed length with the automatic length reload enabled. LENGTH should be loaded with the length of M1 and CHECK with 0.

6.2.3 Register protection and monitoring methods

Register Monitoring: applied to CFG and CHECK registers

Because CFG and CHECK registers are critical to the CRC operation, some mechanisms to detect and log transient errors are provided. Early detection of transient failures enables to improve the failure detection time and assess the severity of the failure. The monitoring mechanisms are implemented using two redundant instances as presented in [Figure 6-4](#).

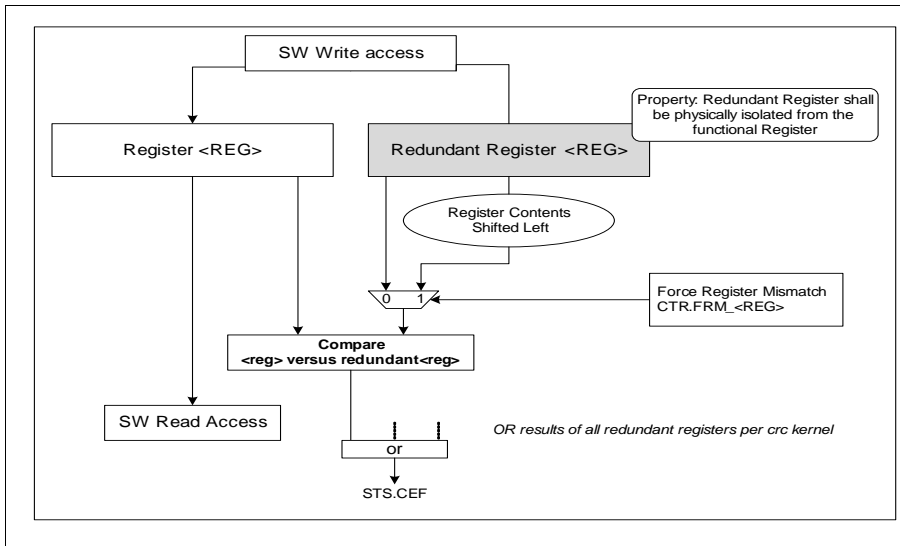


Figure 6-4 Register monitoring scheme

Let <REG> designate either CFG or CHECK registers. When a write to <REG> takes place the redundant register is also updated. **Redundant registers are not visible to software.** Bits of <REG> reserved have no storage and are not used for redundancy. A compare logic continuously compares the two stored values and provides a signal that indicates if the compare is successful or not. The result of all compare blocks are orred together to provide a single flag information. If a mismatch is detected the **STS.CEF** (Configuration Error Flag) bit is set. For run-time validation of the compare logic a Force Register Mismatch bit field (**CTR.FRM_<REG>**) is provided. When set to 1 by software the contents of the redundant register is shifted left by one bit position (redundant bit 0 position is always replaced by a logical 0 value) and is given to the compare logic instead of the redundant register value. This enables to check the compare logic is functional. Using a walking bit pattern, the software can completely check the full operation of the compare logic. Software needs to clear the **CTR.FRM_<REG>** bit to '0' to be able to trigger again a new comparison error interrupt.

Register Access Protection: applies to LENGTH and CHECK registers

In order to reduce the probability of a mis-configuration of the CHECK and LENGTH registers (in the case the automatic check is used), the write access to the CHECK and LENGTH registers must follow a procedure:

Let <REG> designate **CHECK** or **LENGTH** registers. Before being able to configure a new <value> value into the <REG> register of a CRC kernel, software must first write the

Flexible CRC Engine (FCE)

0xFACECAFE value to the <REG> address. The 0xFACECAFE is not written into the <REG> register. The next write access will proceed as a normal bus write access. The write accesses shall use full 32-bit access only. This procedure will then be repeated every time software wants to configure a new <REG> value. If software reads the CHECK register just after writing 0xFACECAFE it returns the current <REG> contents and not 0xFACECAFE. **A read access to <REG> has no effect on the protection mechanism.**

The following C-code shows write accesses to the CHECK and LENGTH registers following this procedure:

```
//set CHECK register
FCE_CHECK0.U = 0xFACECAFE;
FCE_CHECK0.U = 0;

//set LENGTH register
FCE_LENGTH0.U = 0xFACECAFE;
FCE_LENGTH0.U = 256;
```

6.3 Service Request Generation

Each FCE CRC kernel provides one internal interrupt source. The interrupt lines from each CRC kernel are ored together to be sent to the interrupt system. The system interrupt is an active high pulse with the duration of one cycle (of the peripheral clock). The FCE interrupt handler can use the status information located within the **STS** status register of each CRC kernel.

Each CRC kernel provides the following interrupt sources:

- CRC Mismatch Interrupt controlled by **CFG.CMI** bit field and observable via the status bit field **STS.CMF** (CRC Mismatch Flag).
- Configuration Error Interrupt controlled by **CFG.CEI** bit field and observable via the status bit field **STS.CEF** (Configuration Error Flag).
- Length Error Interrupt controlled by **CFG.LEI** bit field and observable via the status bit field **STS.LEF** (Length Error Flag).
- Bus Error Interrupt controlled by **CFG.BEI** bit field and observable via the status bit field **STS.BEF** (Bus Error Flag).

Interrupt generation rules

- A status flag shall be cleared by software by writing a **1** to the corresponding bit position.
- If an status flag is set and a new hardware condition occurs, no new interrupt is generated by the kernel: the STS.<FLAG> bit field masks the generation of a new

Flexible CRC Engine (FCE)

interrupt from the same source. If a SW access to clear the interrupt status bit takes place and in the same cycle the hardware wants to set the bit, the hardware condition wins the arbitration.

As all the interrupts are caused by an error condition, the interrupt shall be handled by a Error Management software layer. The software services using the FCE as acceleration engine may not directly deal with error conditions but let the upper layer using the service to deal with the error handling.

6.4 Debug Behavior

The FCE has no specific debug feature.

6.5 Power, Reset and Clock

The FCE is inside the power core domain, therefore no special considerations about power up or power down sequences need to be taken. For an explanation about the different power domains, please address the SCU (System Control Unit) section.

A power down mode can be achieved by disabling the module using the [Clock Control Register \(CLC\)](#).

The FCE module has one reset source. This reset source is handled at system level and it can be generated independently via a system control register (address SCU section for full description).

After release, the complete IP is set to default configuration. The default configuration for each register field is addressed on [Section 6.7](#).

The FCE uses the CPU clock, fCPU (address SCU section for more details on clocking).

6.6 Initialization and System Dependencies

The FCE may have dependencies regarding the bus clock frequency. This dependencies should be addressed in the SCU and System Architecture sections.

Initialization:

The FCE is enabled by writing 0x0 to the CLC register. Software must first ensure that the CRC kernel is properly configured, especially the initial CRC register value written via the CRC register, the input and result reflection as well as the final xored value via the CFG register. The following source code is an example of initialization for the basic operation of the FCE kernel 0:

```
//enable FCE
FCE_CLC.U = 0x0;
//final result to be xored with 0xFFFFFFFF, no reflection
```

```
FCE_CFG0.U = 0x400;  
//set CRC initial value (seed)  
FCE_CRC0.U = 0xFFFFFFFF;
```

6.7 Registers

Table 6-3 show all registers associated with a FCE CRC-kernel. All FCE kernel register names are described in this section. They should get the prefix “FCE_” when used in the context of a product specification.

The registers are numbered by one index to indicate the related FCE CRC Kernel ($m = 0-3$). Some kernel registers are adapted to the degree of the polynomial implemented by the kernel.

Table 6-2 Registers Address Space - FCE Module

Module	Base Address	End Address	Note
FCE	5002 0000 _H	5002 3FFF _H	

Table 6-3 Registers Overview - CRC Kernel Registers

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Reset Class	Description See
			Read	Write		
System Registers						
CLC	Clock Control Register	00 _H	U, PV	PV	3	Page 6-12
ID	Module Identification Register	08 _H	U, PV	BE	3	Page 6-12
Generic CRC Engine Registers						
IR _m	Input Register m	20 _H + m*20 _H	U, PV	U, PV	3	Page 6-13
RES _m	CRC Result Register m	24 _H + m*20 _H	U, PV	BE	3	Page 6-15
CFG _m	CRC Configuration Register m	28 _H + m*20 _H	U, PV	PV	3	Page 6-17
STS _m	CRC Status Register m	2C _H + m*20 _H	U, PV	U, PV	3	Page 6-19
LENGTH _m	CRC Length Register m	30 _H + m*20 _H	U, PV	U, PV	3	Page 6-20
CHECK _m	CRC Check Register m	34 _H + m*20 _H	U, PV	U, PV	3	Page 6-20
CRC _m	CRC Register m	38 _H + m*20 _H	U, PV	U, PV	3	Page 6-22
CTR _m	CRC Test Register m	3C _H + m*20 _H	U, PV	U, PV	3	Page 6-23

- 1) The absolute register byte address for each CRC kernel m is calculated as follows:
CRC kernel register base Address (Table 6-2) + $m \cdot 20H$, $m = 0-3$

Disabling the FCE

The FCE module can be disabled using the **CLC** register.

When the disable state is requested all pending transactions running on the bus slave interface must be completed before the disabled state is entered. The CLC Register Module Disable Bit Status CLC.DISS indicates whether the module is currently disabled (DISS == 1). Any attempt to write any register with the exception of the CLC Register will generate a bus error. A read operation is allowed and does not generate a bus error.

6.7.1 System Registers description

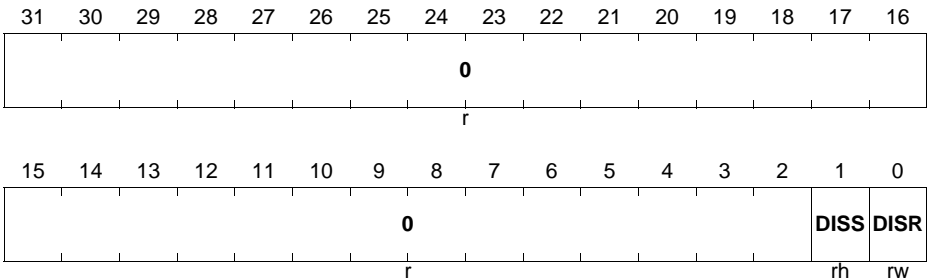
This section describes the registers related to the product system architecture.

Clock Control Register (CLC)

The Clock Control Register allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application.

CLC

Clock Control Register (00_H) **Reset Value: 0000 0003_H**

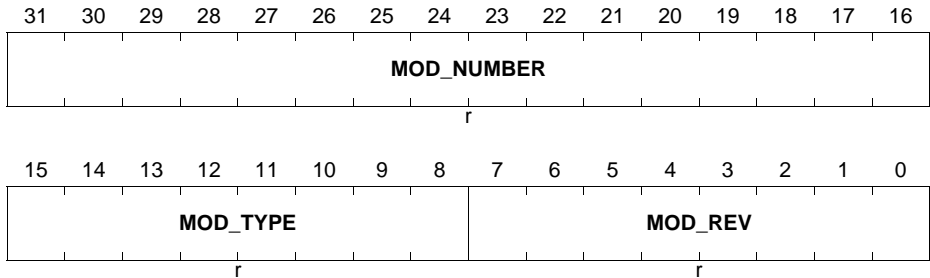


Field	Bits	Type	Description
DISR	0	rw	Module Disable Request Bit Used for enable/disable control of the module.
DISS	1	rh	Module Disable Status Bit Bit indicates the current status of the module.
0	[31:2]	r	Reserved Read as 0; should be written with 0.

Module Identification Register

ID

Module Identification Register (08_H) **Reset Value: 00CA C001_H**



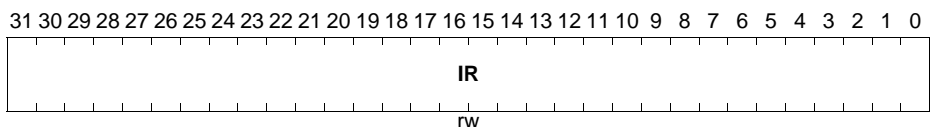
Field	Bits	Type	Description
MOD_REV	[7:0]	r	Module Revision Number This bit field defines the module revision number. The value of a module revision starts with 01 _H (first revision). The current revision number is 01 _H .
MOD_TYPE	[15:8]	r	Module Type The bit field is set to C0 _H which defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	Module Number Value This bit field defines a module identification number. The value for the FCE module is 00CA _H .

6.7.2 CRC Kernel Control/Status Registers

CRC Engine Input Register

IR_m (m = 0-1)

Input Register m (20_H + m*20_H) **Reset Value: 0000 0000_H**



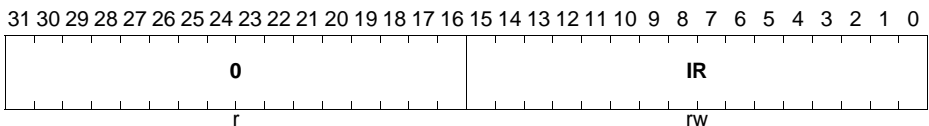
Field	Bits	Type	Description
IR	[31:0]	rw	Input Register This bit field holds the 32-bit data to be computed

A write to IR_m triggers the CRC kernel to update the message checksum according to the IR contents and to the current CRC register contents. Only 32-bit write transactions are allowed to this IR_m registers, any other bus write transaction will lead to a Bus Error.

CRC Engine Input Register

IR_m (m = 2-2)

Input Register m **(20_H + m*20_H)** **Reset Value: 0000 0000_H**



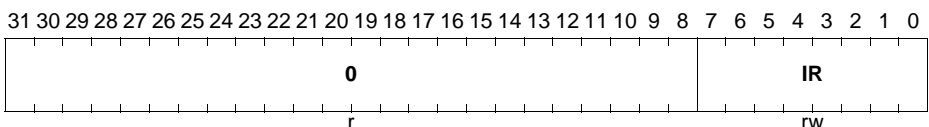
Field	Bits	Type	Description
IR	[15:0]	rw	Input Register This bit field holds the 16-bit data to be computed
0	[31:16]	r	Reserved Read as 0; should be written with 0.

A write to IR_m triggers the CRC kernel to update the message checksum according to the IR contents and to the current CRC register contents. Only 32-bit or 16-bit write transactions are allowed to this IR_m register, any other bus write transaction will lead to a Bus Error. Only the lower 16-bit of the write transactions will be used.

CRC Engine Input Register

IR_m (m = 3-3)

Input Register m **(20_H + m*20_H)** **Reset Value: 0000 0000_H**



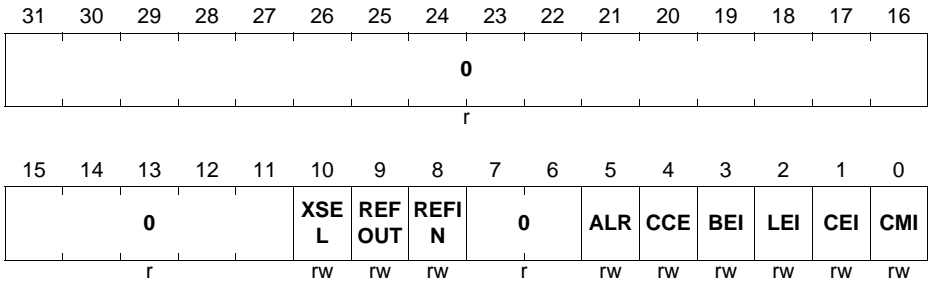
CRC Engine Configuration Register

CFGm (m = 0-3)

CRC Configuration Register m

(28_H + m*20_H)

Reset Value: 0000 0700_H



Field	Bits	Type	Description
CMI	0	rw	CRC Mismatch Interrupt 0 _B CRC Mismatch Interrupt is disabled 1 _B CRC Mismatch Interrupt is enabled
CEI	1	rw	Configuration Error Interrupt When enabled, a Configuration Error Interrupt is generated whenever a mismatch is detected in the CFG and CHECK redundant registers. 0 _B Configuration Error Interrupt is disabled 1 _B Configuration Error Interrupt is enabled
LEI	2	rw	Length Error Interrupt When enabled, a Length Error Interrupt is generated if software writes to IR register with LENGTH equal to 0 and CFG.CCE is set to 1. 0 _B Length Error Interrupt is disabled 1 _B Length Error Interrupt is enabled
BEI	3	rw	Bus Error Interrupt When enabled, an interrupt is generated if a bus write transaction with an access width smaller than the kernel width is issued to the input register. 0 _B Bus Error Interrupt is disabled 1 _B Bus Error Interrupt is enabled

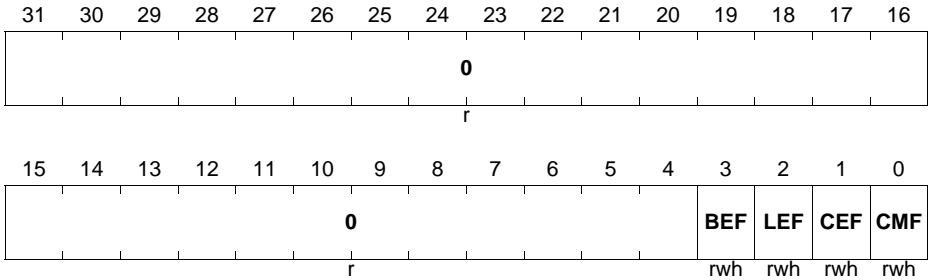
Flexible CRC Engine (FCE)

Field	Bits	Type	Description
CCE	4	rw	CRC Check Comparison 0 _B CRC check comparison at the end of a message is disabled 1 _B CRC check comparison at the end of a message is enabled
ALR	5	rw	Automatic Length Reload 0 _B Disables automatic reload of the LENGTH field. 1 _B Enables automatic reload of the LENGTH field at the end of a message.
REFIN	8	rw	IR Byte Wise Reflection 0 _B IR Byte Wise Reflection is disabled 1 _B IR Byte Wise Reflection is enabled
REFOUT	9	rw	CRC 32-Bit Wise Reflection 0 _B CRC 32-bit wise is disabled 1 _B CRC 32-bit wise is enabled
XSEL	10	rw	Selects the value to be xored with the final CRC 0 _B 0x00000000 1 _B 0xFFFFFFFF
0	[7:6], [31:11]	r	Reserved Read as 0; should be written with 0.

CRC Engine Status Register

STSm (m = 0-3)

CRC Status Register m (2C_H + m*20_H) Reset Value: 0000 0000_H

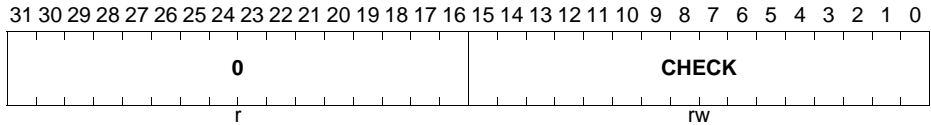


Field	Bits	Type	Description
CMF	0	rwh	CRC Mismatch Flag This bit is set per hardware only. To clear this bit, software must write a 1 to this bit field location. Writing 0 per software has no effect.
CEF	1	rwh	Configuration Error Flag This bit is set per hardware only. To clear this bit, software must write a 1 to this bit field location. Writing 0 per software has no effect.
LEF	2	rwh	Length Error Flag This bit is set per hardware only. To clear this bit, software must write a 1 to this bit field location. Writing 0 per software has no effect.
BEF	3	rwh	Bus Error Flag This bit is set per hardware only. To clear this bit, software must write a 1 to this bit field location. Writing 0 per software has no effect.
0	[31:4]	r	Reserved Read as 0; should be written with 0.

CRC Engine Check Register

CHECK_m (m = 2-2)

CRC Check Register m (34_H + m*20_H) **Reset Value: 0000 0000_H**

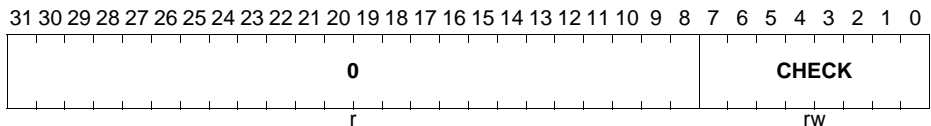


Field	Bits	Type	Description
CHECK	[15:0]	rw	CHECK Register Expected CRC value to be checked by the hardware upon detection of a 1 to 0 transition of the LENGTH register. The comparison is enabled by the CFG.CCE bit field
0	[31:16]	r	Reserved Read as 0; should be written with 0.

CRC Engine Check Register

CHECK_m (m = 3-3)

CRC Check Register m (34_H + m*20_H) **Reset Value: 0000 0000_H**

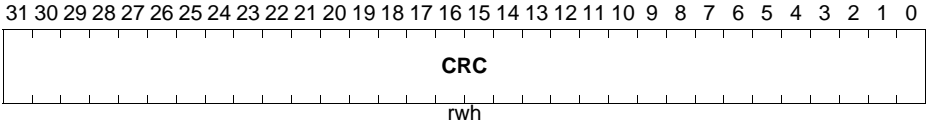


Field	Bits	Type	Description
CHECK	[7:0]	rw	CHECK Register Expected CRC value to be checked by the hardware upon detection of a 1 to 0 transition of the LENGTH register. The comparison is enabled by the CFG.CCE bit field
0	[31:8]	r	Reserved Read as 0; should be written with 0.

CRC Engine Initialization Register

CRCm (m = 0-1)

CRC Register m (38_H + m*20_H) **Reset Value: 0000 0000_H**

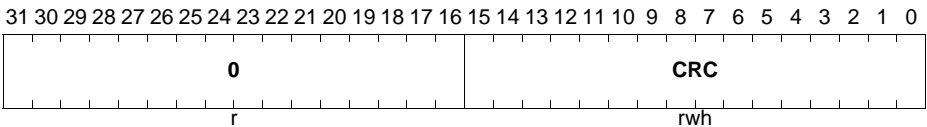


Field	Bits	Type	Description
CRC	[31:0]	rwh	CRC Register This register enables to directly access the internal CRC register

CRC Engine Initialization Register

CRCm (m = 2-2)

CRC Register m (38_H + m*20_H) **Reset Value: 0000 0000_H**

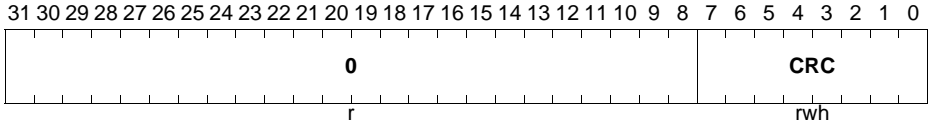


Field	Bits	Type	Description
CRC	[15:0]	rwh	CRC Register This register enables to directly access the internal CRC register
0	[31:16]	r	Reserved Read as 0; should be written with 0.

CRC Engine Initialization Register

CRCm (m = 3-3)

CRC Register m (38_H + m*20_H) **Reset Value: 0000 0000_H**

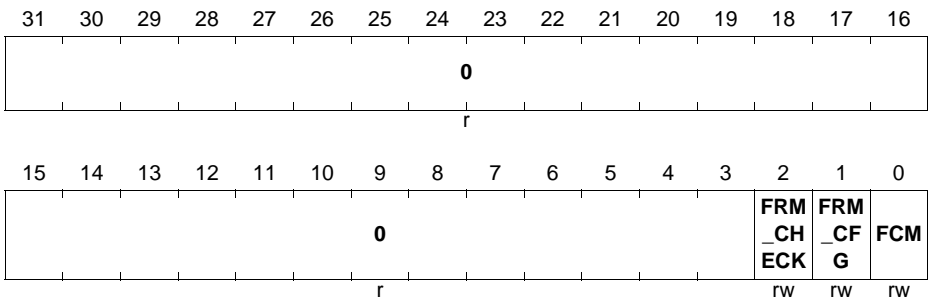


Field	Bits	Type	Description
CRC	[7:0]	rwh	CRC Register This register enables to directly access the internal CRC register
0	[31:8]	r	Reserved Read as 0; should be written with 0.

CRC Test Register

CTRm (m = 0-3)

CRC Test Register m (3C_H + m*20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
FCM	0	rw	Force CRC Mismatch Forces the CRC compare logic to issue an error regardless of the CHECK and CRC values. The hardware detects a 0 to 1 transition of this bit field and triggers a CRC Mismatch interrupt

Field	Bits	Type	Description
FRM_CFG	1	rw	Force CFG Register Mismatch This field is used to control the error injection mechanism used to check the compare logic of the redundant CFG registers. This is a one shot operation. When the hardware detects a 0 to 1 transition of this bit field it triggers a Configuration Mismatch interrupt (if enabled by the corresponding CFGm register).
FRM_CHECK	2	rw	Force Check Register Mismatch This field is used to control the error injection mechanism used to check the compare logic of the redundant CHECK registers. This is a one shot operation. The hardware detects a 0 to 1 transition of this bit field and triggers a Check Register Mismatch interrupt (if enabled by the corresponding CFGm register).
0	[31:3]	r	Reserved Read as 0; should be written with 0.

6.8 Interconnects

The interfaces of the FCE module shall be described in the module design specification. The [Table 6-4](#) shows the services requests of the FCE module.

Table 6-4 FCE Service Requests

Inputs/Outputs	I/O	Connected To	Description
FCE.SR0	O	NVIC	Service request line

6.9 Properties of CRC code

Hamming Distance

The Hamming distance defines the error detection capability of a CRC polynomial. A cyclic code with a Hamming Distance of D can detect all D-1 bit errors. [Table 6-5 “Hamming Distance as a function of message length \(bits\)” on Page 6-25](#) shows the dependency of the Hamming Distance with the length of the message.

Table 6-5 Hamming Distance as a function of message length (bits)¹⁾

Hamming Distance	IEEE-802.3 CRC32	CCITT CRC16	J1850 CRC8
15	8 - 10	Information not available	Information not available
14	8 - 10		
13	8 - 10		
12	11 - 12		
11	13 - 21		
10	22 - 34		
9	35 - 57		
8	58 - 91		
7	92 - 171		
6	172 - 268		
5	269 - 2974		
4	2973 - 91607		
3	91607 - 131072		

1) Data from technical paper "32-Bit Cyclic Redundancy Codes for Internet Applications" by Philip Koopman, Carnegie Mellon University, 2002



On-Chip Memories

7 Memory Organization

This chapter provides description of the system Memory Organization and basic information related to Parity Testing and Parity Error handling.

References

[8] Cortex™-M4 User Guide, ARM DUI 0508B (ID062910)

7.1 Overview

The Memory Map is intended to balance decoding cost at various level of the system bus infrastructure.

7.1.1 Features

The Memory Map implements the following features:

- Compatibility with standard ARM Cortex-M4 CPU [\[8\]](#)
- Compatibility across entire XMC4000 Family
- Optimal functional module address spaces grouping

7.1.2 Cortex-M4 Address Space

The system memory map defines several regions. Address boundaries of each of the regions are determined by the Cortex-M4 core architecture.

Memory Organization

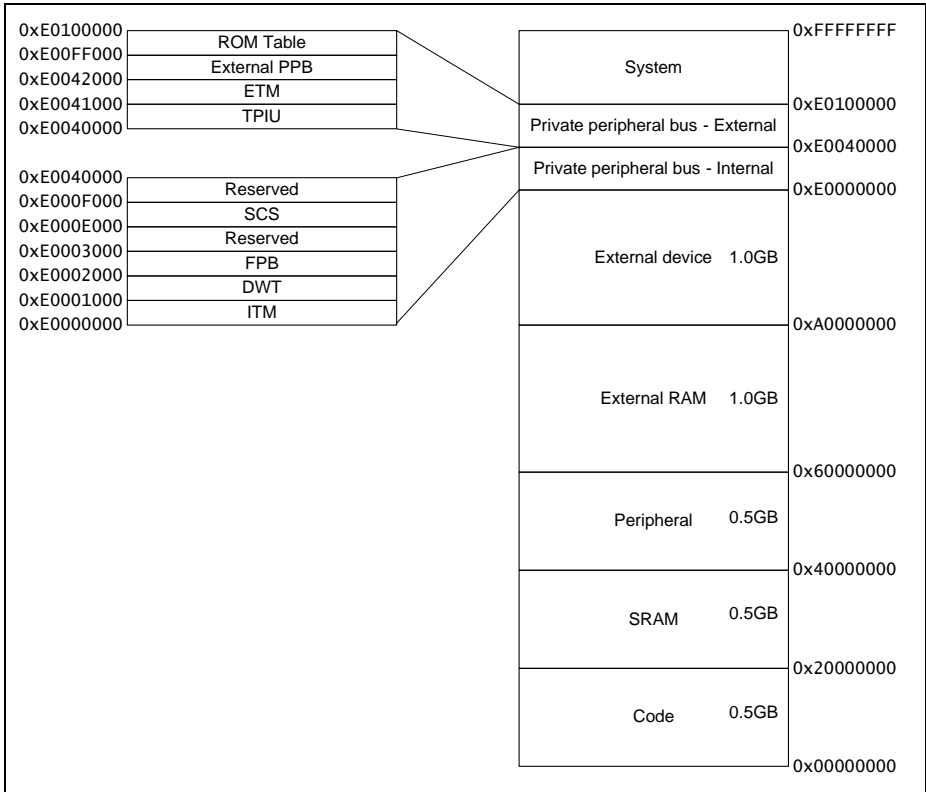


Figure 7-1 Cortex-M4 processor address space

7.2 Memory Regions

The XMC4300 device specific address map assumes presence of internal and external memories and peripherals. The memory regions for XMC4300 are described in [Table 7-1](#).

Table 7-1 Memory Regions

Start	End	Size (hex)	Space name	Usage
00000000	1FFFFFFF	20000000	Code	Boot ROM Flash Program SRAM
20000000	3FFFFFFF	20000000	SRAM	Fast internal SRAMs
40000000	47FFFFFFF	08000000	Peripheral 0	Internal Peripherals group 0
48000000	4FFFFFFF	08000000	Peripheral 1	Internal Peripherals group 1
50000000	57FFFFFFF	08000000	Peripheral 2	Internal Peripherals group 2
58000000	5FFFFFFF	08000000	Peripheral 3	Internal Peripherals group 3
60000000	9FFFFFFF	40000000	External SRAM	External Memories
A0000000	DFFFFFFF	40000000	External Device	External Devices
E0000000	E0FFFFFFF	00100000	Private Peripheral Bus	CPU
E0100000	FFFFFFF	0FF00000	Vedor specific 1	reserved
F0000000	FFFFFFF	10000000	Vedor specific 2	reserved

7.3 Memory Map

[Table 7-2](#) defines detailed system memory map of XMC4300 where each individual peripheral or memory instance implement its own address spaces. For detailed register description of the system components and peripherals please refer to respective chapters of this document.

Table 7-2 Memory Map

Addr space	Start Address (hex)	End Address (hex)	Modules
Code	00000000	00003FFF	BROM (PMU ROM)
	00004000	07FFFFFF	reserved
	08000000	0801FFFF	PMU/FLASH (cached)
		09E1FFFF	reserved
	09E20000	09E23FFF	reserved
	09E24000	0BFFFFFF	reserved
	0C000000	0C01FFFF	PMU/FLASH (uncached)
		0FFFFFFF	reserved
	0DE20000	0DE23FFF	reserved
	0DE24000	0FFFFFFF	reserved
	10000000		reserved
	1FFF0000	1FFFFFFF	PSRAM (code)
SRAM	20000000	2000FFFF	DSRAM1 (system)
			reserved
			reserved
			reserved
	30008000	3FFFFFFF	reserved

Memory Organization

Table 7-2 Memory Map (cont'd)

Addr space	Start Address (hex)	End Address (hex)	Modules
Peripherals 0	40000000	40003FFF	PBA0
	40004000	40007FFF	VADC
	40008000	4000BFFF	reserved
	4000C000	4000FFFF	CCU40
	40010000	40013FFF	CCU41
	40014000	40017FFF	reserved
	40018000	4001BFFF	reserved
	4001C000	4001FFFF	reserved
	40020000	40023FFF	CCU80
	40024000	40027FFF	reserved
	40028000	4002BFFF	POSIF0
	4002C000	4002FFFF	reserved
	40030000	40033FFF	USIC0
	40034000	40037FFF	reserved
	40038000	4003BFFF	reserved
	4003C000	4003FFFF	reserved
	40044000	40047FFF	ERU1
	40048000	47FFFFFF	reserved
Peripherals 1	48000000	48003FFF	PBA1
	48004000	48007FFF	reserved
	48008000	4800BFFF	reserved
	4800C000	4800FFFF	reserved
	48010000	48013FFF	LEDTS0
	48014000	48017FFF	MultiCAN
	48018000	4801BFFF	DAC
	4801C000	4801FFFF	SDMMC
	48020000	48023FFF	USIC1
	48024000	48027FFF	reserved
	48028000	4802BFFF	PORTS
	4802C000	4FFFFFFF	reserved

Memory Organization

Table 7-2 Memory Map (cont'd)

Addr space	Start Address (hex)	End Address (hex)	Modules
Peripherals 2	50000000	50003FFF	PBA2
	50004000	50007FFF	SCU & RTC
	50008000	5000BFFF	WDT
	5000C000	5000FFFF	ETH
	50010000	50013FFF	reserved
	50014000	50017FFF	DMA0
	50018000	5001BFFF	reserved
	5001C000	5001FFFF	reserved
	50020000	50023FFF	FCE
	50024000	5003FFFF	reserved
	50040000	5007FFFF	USB
	50080000	5008FFFF	reserved
	54000000	54003FFF	reserved
	54004000	54007FFF	reserved
	54008000	5400BFFF	reserved
	5400C000	5400FFFF	reserved
	54010000	54013FFF	ECAT0
	54014000	57FFFFFF	reserved
Peripherals 3	58000000	58003FFF	PMU0 registers
	58004000	58007FFF	PMU0 prefetch
	58008000	5800BFFF	reserved
	5800C000	5800FFFF	reserved
	58010000	58013FFF	reserved
	58014000	58017FFF	reserved
	58018000	5FFFFFFF	reserved
External SRAM	60000000	63FFFFFF	reserved
	64000000	67FFFFFF	reserved
	68000000	6BFFFFFF	reserved
	6C000000	6FFFFFFF	reserved
	70000000	9FFFFFFF	reserved

Table 7-2 Memory Map (cont'd)

Addr space	Start Address (hex)	End Address (hex)	Modules
External Device	A0000000	A3FFFFFF	reserved
	A4000000	A7FFFFFF	reserved
	A8000000	ABFFFFFF	reserved
	AC000000	AFFFFFFF	reserved
	B0000000	DFFFFFFF	reserved
Private Peripheral Bus	E0000000	E000FFF	ITM
	E0001000	E0001FFF	DWT
	E0002000	E0002FFF	FPB
	E0003000	E000DFFF	reserved
	E000E000	E000EFFF	SCS
	E000E010	E000E01C	SysTick
	E000EF34	E000EF47	FPU
	E000F000	E003FFFF	reserved
	E0040000	E0040FFF	TPIU
	E0041000	E0041FFF	ETM
	E0042000	E00FEFFF	reserved
	E00FF000	E00FFFFFF	ROM Table
	Vedor specific 1	E0100000	FFFFFFF
Vedor specific 2	F0000000	FFFFFFF	reserved

7.4 Service Request Generation

Memory modules and other system components are capable of generating error responses indicated to the CPU as bus error exceptions or interrupts.

Types of error causes

- Unsupported Access Mode
- Access to Invalid Address
- Parity Error (memories only)
- Bufferable Write Access to Peripheral

Errors that cannot be indicated with bus errors get indicated with service requests that get propagated to the CPU as interrupts. Typically lack of bus error response capability applies to memory modules that lack of direct access from the system bus This applies to memories that serve the purpose of internal FIFOs and local storage buffers.

Unsupported Access Modes

Unsupported access modes can be classified in various ways and are usually specific to the module that access is performed to. The typical examples of unsupported access modes are read access to write-only or write access to read-only type of address mapped resources, unsupported access data widths, protected memory regions. For module specific limitations please refer to individual module chapters.

Invalid Address

Accesses to invalid addresses result in error responses. Invalid addresses are defined as those that do not map to any valid resources. This applies to single addresses and to wider address ranges. Some invalid addresses within valid module address ranges may not produce error responses and this is specific to individual modules.

Parity Errors

Parity test is performed on the XMC4300 memories in normal functional mode. Parity errors are generated in case of failure of parity test performed inside of each of the memory module. The mechanism of parity testing depends on memory data width and access mode, i.e. memory modules that are accessible byte-wise implement parity check for each data byte individually while for memory modules that are accessible double-word-wise it is sufficient to perform joint check for all bits.

The occurrence of a parity error gets signaled to the system with system bus error or an interrupt (parity trap). For details on parity error generation control and handling please refer to the SCU chapter. For more details please refer to [Table 7-3](#).

Table 7-3 Parity Test Enabled Memories and Supported Parity Error Indication

Memory	Number of Parity Bits	Parity Test Granularity	Bus Error	Parity Trap
Program SRAM (PSRAM)	1	4 bytes	yes	yes
System SRAM (DSRAM1)	4	1 byte	yes	yes
USIC 0 Buffer Memory	1	4 bytes	no	yes
USIC 1 Buffer Memory	1	4 bytes	no	yes
MultiCAN Buffer Memory	1	4 bytes	no	yes
PMU Prefetch Buffer Memory	1	4 bytes	no	yes
USB Buffer Memory	1	4 bytes	no	yes
ETH 0 TX Buffer Memory	1	4 bytes	no	yes
ETH 0 RX Buffer Memory	1	4 bytes	no	yes

Table 7-3 Parity Test Enabled Memories and Supported Parity Error Indication
(cont'd)

Memory	Number of Parity Bits	Parity Test Granularity	Bus Error	Parity Trap
ECAT 0 RX Buffer Memory	1	1 byte	yes	yes
SDMMC Buffer Memory 0	1	4 bytes	no	yes
SDMMC Buffer Memory 1	1	4 bytes	no	yes

Bufferable Write Access to Peripheral

Bufferable writes to peripheral may result in error responses as described above. Bus error responses from modules attached to peripheral bridges PBA0 and PBA1 trigger service request from the respective bridge that will result in NMI to the CPU. Error status and access address that caused the service request get stored in dedicated registers of the peripheral bridges. For detail please refer to [Registers](#).

7.5 Debug Behavior

The bus system in debug mode allows debug probe access to all system resources except for the Flash sectors protected with a dedicated protection mechanism (for more details please refer to Flash Memory chapter). No special handling of HALT mode is implemented and all interfaces respond with a valid bus response upon accesses.

7.6 Power, Reset and Clock

The bus system clocking scheme enables stable system operation and accesses to system resources for all valid system clock rates. Some parts of the system may also run at a half of the system clock rate and no special handling is required as appropriate alignment of the bus system protocol is provided on the clock domain boundary (for details please refer to clocking system description in SCU chapter).

7.7 Initialization and System Dependencies

No initialization is required for the memory system from user point of view. All valid memories are available after reset. Some peripherals may need to be initialized (e.g. released from reset state) before accessed. For details please refer to individual peripheral chapters.

7.8 Registers

This section describes registers of the Peripheral Bridges. The purpose of the registers is handling of errors signaled during bufferable accesses to peripherals connected to the respective bridges. Active errors on bufferable writes trigger interrupt requests generated from the Peripheral Bridges that can be monitored and cleared in the register defined in this chapter.

Table 7-4 Registers Address Space

Module	Base Address	End Address	Note
PBA0	4000 0000 _H	4000 3FFF _H	Peripheral Bridge 0
PBA1	4800 0000 _H	4800 3FFF _H	Peripheral Bridge 1

Table 7-5 Registers Overview

Register Short Name	Register Long Name	Offset Address	Access Mode		Description
			Read	Write	
PBA0_STS	PBA 0 Status Register	0000 _H	U, PV	PV	Page 7-10
PBA0_WADDR	PBA 0 Write Error Address	0004 _H	U, PV		Page 7-11
PBA1_STS	PBA 1 Status Register	0000 _H	U, PV	PV	Page 7-12
PBA1_WADDR	PBA 1 Write Error Address	0004 _H	U, PV		Page 7-12

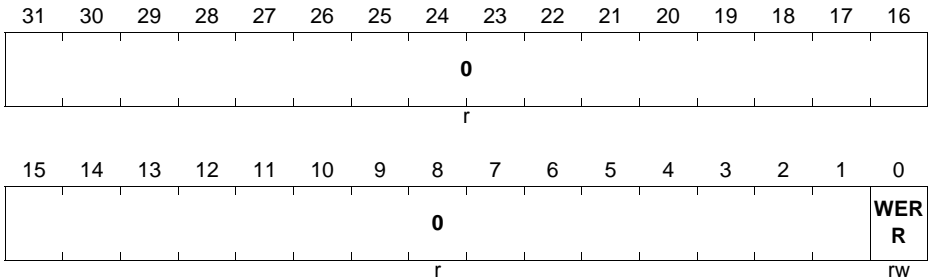
PBA0_STS

The status register of PBA0 bridge indicates bus error occurrence for write access. Is meant to be used for errors triggered upon buffered writes. The bit gets set and interrupt request has been generated to the SCU.

Write one to clear, writing zero has no effect.

PBA0_STS

Peripheral Bridge Status Register (0000_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
WERR	0	rw	Bufferable Write Access Error 0 _B no write error occurred. 1 _B write error occurred, interrupt request is pending.
0	[31:1]	r	Reserved bits. Write zeros

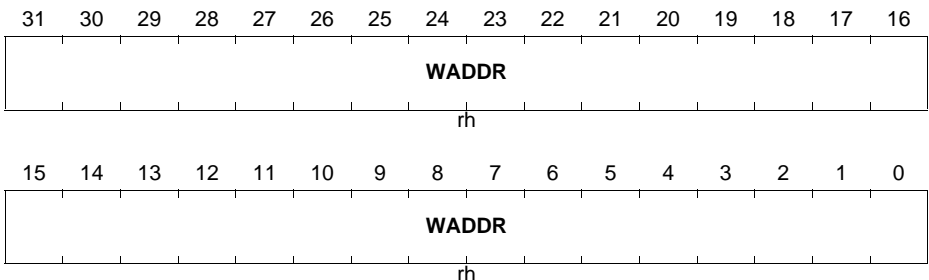
PBA0_WADDR

The Write Error Address Register keeps write access address that caused a bus error upon bufferable write attempt to a peripheral connected to PBA0 bridge. This register store the address that of the bufferable write access attempt that caused error resulting in setting WERR bit of the **PBA0_STS** register.

This register value remains unchanged when WERR bit of **PBA0_STS** register is set.

PBA0_WADDR

PBA Write Error Address Register (0004_H) **Reset Value: 0000 0000_H**

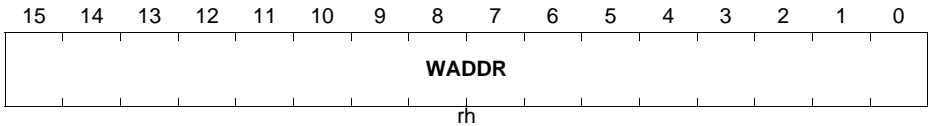
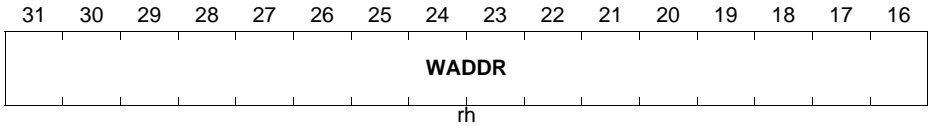


Memory Organization

PBA1_WADDR

PBA Write Error Address Register (0004_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
WADDR	[31:0]	rh	Write Error Address Address of the write access that caused a bus error on the bridge Master port.

8 Flash and Program Memory Unit (PMU)

The Program Memory Unit (PMU) controls the Flash memory and the BROM and connects these to the system. The Prefetch unit maximizes system performance with higher system frequencies, by buffering instruction and data accesses to the Flash.

8.1 Overview

In the XMC4300, the PMU controls the following interfaces:

- The Flash command and fetch control interface for Program Flash
- The Boot ROM interface
- The PMU interfaces via the Prefetch unit to the Bus Matrix

Following memories are controlled by and belong to the PMU:

- 256 kbyte of Program Flash memory (PFLASH)
- 16 Kbyte of BROM (BROM)
- 8 Kbyte of Instruction Cache memory in the Prefetch unit
- 256-bit Data Buffer in the Prefetch unit

8.1.1 Block Diagram

The PMU block diagram is shown in [Figure 8-1](#).

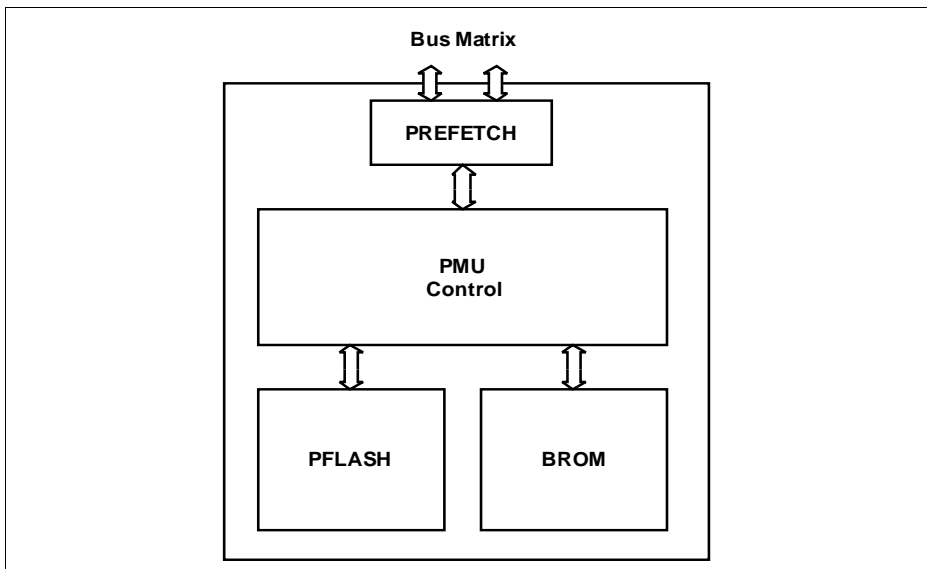


Figure 8-1 PMU Block Diagram

8.2 Boot ROM (BROM)

The Boot ROM in PMU0 has a capacity of 16 KB. The BROM contains the Firmware with:

- startup routines
- bootstrap loading software.

Details on the operations of the BROM are given in the chapter “Startup Modes”.

8.2.1 BROM Addressing

The BROM is visible at one location, as can be seen in the memory map:

- (non-cached space) starting at location 0000 0000_H

After any reset, the hardware-controlled start address is 0000 0000_H. At this location, the startup procedure is stored and started. As no other start location after reset is supported, the startup software within the BROM is always executed first after any reset.

8.3 Prefetch Unit

The purpose of the Prefetch unit is to reduce the Flash latency gap at higher system frequencies to increase the instruction per cycle performance.

8.3.1 Overview

The Prefetch unit separates between instruction and data accesses to the Flash with the following configuration:

- 8 Kbyte Instruction Buffer
 - 2-way set associative
 - Least-Recently-Used (LRU) replacement policy
 - Cache line size: 256 bits
 - Critical word first
 - Streaming¹⁾
 - Line wrap around
 - Parity, 32-bit granularity
 - Buffer can be bypassed
 - Buffer can be globally invalidated
- 256-bit Data Buffer
 - Single line
 - Critical word first
 - Streaming¹⁾
 - Line Wrap around
 - Buffer can be bypassed

1) The first 32-bit data from Flash gets immediately forwarded to the CPU

Flash and Program Memory Unit (PMU)

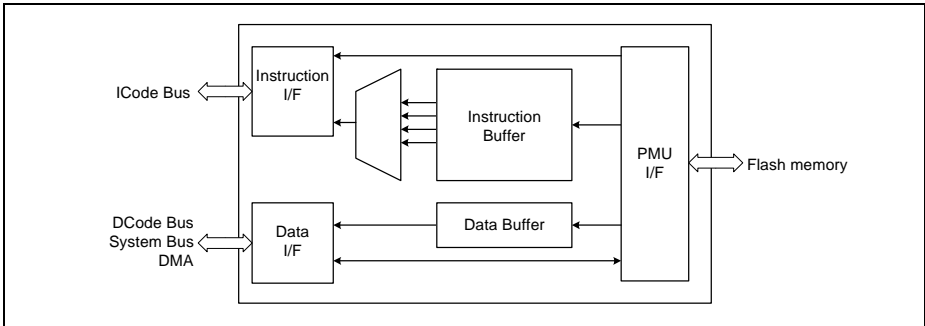


Figure 8-2 Prefetch Unit

8.3.2 Operation

8.3.2.1 Instruction Buffer

The instruction buffer acts like a regular instruction cache with the characteristics described in the overview, optimized for minimum latency via the dedicated instruction interface. Instruction fetches to the non-cacheable address space bypass the instruction buffer. For software development and benchmarking purposes the cacheable accesses can also bypass the instruction buffer by setting `PREF_PCON.IBYP` to `1B`.

Prefetch buffer hits are without any penalty i.e. single cycle access rate. This ensures a minimized latency.

The instruction buffer may be invalidated by writing a `1B` to `PREF_PCON.IINV`. After system reset, the instruction buffer is automatically invalidated.

A parity error during a buffer read operation is automatically turned into a buffer miss, triggering a refill operation of the cache line.

Note: The complete invalidation operation is performed in a single cycle.

Note: The parity information is generated on the fly during the cache refill operation. Parity is checked for each read operation targeting the instruction buffer.

The streaming operation is on the fly - it does not cause any additional latency.

8.3.2.2 Data Buffer

The characteristics of the data buffer are described in the overview. It is used for data read requests from the CPU using the DCode interface and for data read requests from

Flash and Program Memory Unit (PMU)

the DMA. CPU read accesses to the prefetch buffer are without any penalty i.e. single cycle access rate. The miss latency is minimized.

The data interface is shared between DMA requests, CPU DCode bus requests and CPU System bus requests. The CPU System bus is attached to the Prefetch unit to access configuration and status registers within the Prefetch unit and the PMU and Flash. All read requests outside the cacheable address space and all write accesses bypass the data buffer. For software development and benchmarking purposes the cacheable accesses can also bypass the data buffer by setting **PREF_PCON.DBYP** to 1_B .

Note: The streaming operation is on the fly - it does not cause any additional latency.

Note: An erase and/or program operation to the Flash does not clear/invalidate the data buffer and may result in inconsistent data buffer content (no data coherence between data buffer and the updated Flash).

A (dummy) read operation to a cacheable Flash address that has not been updated restores the data coherency.

8.3.2.3 PMU Interface

Each Flash read access returns 256 bits, intermediately stored in a “global read buffer” in the Flash (**Section 8.4.4**). The Prefetch unit reads from this buffer via a 64-bit interface. Cacheable read accesses that are not yet stored in the Prefetch buffer (cache miss) trigger a refill operation by a 4x64-bit burst transfer. By that burst transfer the data from the global buffer is copied, refilling the instruction buffer (code fetch) or data buffer (data fetch) respectively.

Only the initial Flash read access is affected by the Flash latency. The subsequent read accesses of the burst transfer are serviced by the global read buffer with no additional delay. An additional prefetch mechanism in the PFLASH further reduces the latency for linear Flash accesses (**Section 8.4.4**).

Non-cacheable accesses benefit from the global read buffer in the same way, as long as its content is not “trashed” by a new Flash read access (e.g. from a different bus master).

Accesses to the BROM and register address spaces and write operations are ignored by the Prefetch buffers.

8.4 Program Flash (PFLASH)

This chapter describes the embedded Flash module of the XMC4300 and its software interface.

8.4.1 Overview

The embedded Flash module of XMC4300 includes 256 kB of Flash memory for code or constant data (called Program Flash).

8.4.1.1 Features

The following list gives an overview of the features implemented in the Program Flash. Absolute values can be found in the "Data Sheet".

- Consists of one bank.
- Commonly used for instructions and constant data.
- High throughput burst read based on a 256-bit Flash access.
- Application optimized sector structure with sectors ranging from 16 Kbytes to 256 Kbytes.
- High throughput programming of a 256 byte page (see Data Sheet t_{PRP}).
- Sector-wise erase on logical and physical sectors (see Data Sheet t_{ERP}).
- Write protection separately configurable for groups of sectors.
- Hierarchical write protection control with 3 levels of which 2 are password based and 1 is a one-time programmable one.
- Password based read protection combined with write protection for the whole Flash.
- Separate configuration sector containing the protection configuration and boot configuration (BMI).
- All Flash operations initiated by command sequences as protection against unintended operation.
- Erase and program performed by a Flash specific control logic independent of the CPU.
- End of erase and program operations reported by interrupt.
- Dynamic Error Correcting Code (ECC) with Single-bit Error Correction and Double-bit Error Detection ("SEC-DED").
- Error reporting by bus error, interrupts and status flags.
- Margin reads for quality assurance.
- Delivery in the erased state.
- Configurable wait state configuration for optimum read performance depending on CPU frequency (see [FCON.WSPFLASH](#)).
- High endurance and long retention.
- Pad supply voltage used for program and erase.

8.4.2 Definition of Terms

The description of Flash memories uses a specific terminology for operations and the hierarchical structure.

Flash Operation Terms

- **Erasing:** The erased state of a Flash cell is logical '0'. Forcing a cell to this state is called "erasing". Depending on the Flash area and command sequence complete logical or physical sectors are erased. All Flash cells in this area incur one "cycle" that counts for the "endurance".
- **Programming:** The programmed state of a cell is logical '1'. Changing an erased Flash cell to this state is called "programming". The 1-bits of a page are programmed concurrently.
- **Retention:** This is the time during which the data of a Flash cell can be read reliably. The retention time is a statistical figure that depends on the operating conditions of the device (e.g. temperature profile) and is affected by operations on other Flash cells in the same word-line and physical sector. With an increasing number of program/erase cycles (see endurance) the retention is lowered. Figures are documented in the Data Sheet separately for physical sectors (t_{RET}) and UCBs (t_{RTU}).
- **Endurance:** The maximum number of program/erase cycles of each Flash cell is called "endurance". The endurance is a statistical figure that depends on operating conditions and the use of the flash cells and also on the required quality level. The endurance is documented in the Data Sheet as a condition to the retention parameters.

Flash Structure Terms

- **Flash Module:** The PMU contains one "Flash module" with its own operation control logic.
- **Bank:** A "Flash module" may contain separate "banks". "Banks" support concurrent operations (read, program, erase) with some limitations due to common logic.
- **Physical Sector:** A Flash "bank" consists of "physical sectors" ranging from 64 Kbytes to 256 Kbytes. The Flash cells of different "physical sectors" are isolated from each other. Therefore cycling Flash cells in one physical sector does not affect the retention of Flash cells in other physical sectors. A "physical sector" is the largest erase unit.
- **Logical Sector:** A "logical sector" is a group of word-lines of one physical sector. They can be erased with a single operation but other Flash cells in the same physical sector are slightly disturbed.
- **Sector:** The plain term "sector" means "logical sector" when a physical sector is divided in such, else it means the complete physical sector.
- **User Configuration Block "UCB":** A "UCB" is a specific logical sector contained in the configuration sector. It contains the protection settings and other data configured

Flash and Program Memory Unit (PMU)

by the user. The “UCBs” are the only part of the configuration sector that can be programmed and erased by the user.

- **Word-Line:** A “word-line” consists of two pages, an even one and an odd one. In the PFLASH a word-line contains aligned 512 bytes.
- **Page:** A “page” is a part of a word-line that is programmed at once. In PFLASH a page is an aligned group of 256 bytes.

8.4.3 Flash Structure

The PMU contains one PFLASH bank, accessible via the cacheable or non-cacheable address space. The offset address of each sector is relative to the base address of its bank which is given in [Table 8-1](#).

Derived devices (see Data Sheet) can have less Flash memory. The PFLASH bank shrinks by cutting-off higher numbered physical sectors.

Table 8-1 Flash Memory Map

Range Description	Size	Start Address
PMU0 Program Flash Bank non-cached	256 kbyte	0C00 0000 _H
PMU0 Program Flash Bank cached space (different address space for the same physical memory, mapped in the non-cached address space)	256 kbyte	0800 0000 _H
PMU0 UCB User Configuration Blocks	3 Kbyte	0C00 0000 _H
PMU0 Flash Registers	1 Kbyte	5800 2000 _H

PFLASH

All addresses offset to the start addresses given in [Table 8-1](#).

Table 8-2 Sector Structure of PFLASH

Sector	Phys. Sector	Size	Offset Address
S0	PS0	16 KB	00'0000 _H
S1		16 KB	00'4000 _H
S2		16 KB	00'8000 _H
S3		16 KB	00'C000 _H

Flash and Program Memory Unit (PMU)

Table 8-2 Sector Structure of PFLASH (cont'd)

Sector	Phys. Sector	Size	Offset Address
S4	PS4	16 KB	01'0000 _H
S5		16 KB	01'4000 _H
S6		16 KB	01'8000 _H
S7		16 KB	01'C000 _H
S8	–	128 KB	02'0000 _H

UCB

All addresses offset to the start addresses given in [Table 8-1](#). As explained before the UCBx are logical sectors.

Table 8-3 Structure of UCB Area

Sector	Size	Offset Address
UCB0	1 KB	00'0000 _H
UCB1	1 KB	00'0400 _H
UCB2	1 KB	00'0800 _H

8.4.4 Flash Read Access

Flash banks that are active and in read mode can be directly read like a ROM.

The wait cycles for the Flash read access must be configured based on the CPU frequency f_{CPU} (incl. PLL jitter) in relation to the Flash access time t_a defined in the Data Sheet. The following formula applies for **FCON.WSPFLASH** > 0_H¹⁾:

$$WSPFLASH \times (1 / f_{CPU}) \geq t_a \tag{8.1}$$

The PFLASH delivers 256 bits per read access. All read data from the PFLASH passes through a 256-bit “global read buffer”.

The PMU allows 4x64-bit burst accesses to the cached address space and single 32-bit read accesses to the non-cached address space of the PFLASH.

The Prefetch generates the 4x64-bit bursts for code and data fetches from the cached address range in order to fill one cache line or the data buffer respectively. Data reads from the non-cached address range are performed with single 32-bit transfers.

Following an initial Flash access, the PFLASH automatically starts a prefetch of the next linear address (even before it has been requested). Has the content of the global read

1) WSPFLASH = 0_H deviates from this formula and results in the same timing as WSPFLASH = 1_H.

Flash and Program Memory Unit (PMU)

buffer been read completely (e.g. by a burst from the Prefetch unit), the new prefetched data is copied to the read buffer and another prefetch to the PFLASH is started. This significantly reduces the Flash latency for mostly linearly accessed code or data sections. To avoid additional wait states due to these prefetches, they can be aborted in case a new (initial) read access is requested from a different address. For power saving purposes these prefetch operations can be disabled by **FCON.IDLE (Idle Read Path)**. Read accesses from Flash can be blocked by the read protection (see **Section 8.4.8**). ECC errors can be detected and corrected (see **Section 8.4.9**).

8.4.5 Flash Write and Erase Operations

Flash write and erase operations are triggered by **Command Sequences** to avoid harm to the stored data by “accidental” accesses from faulty code. Erase operations are executed on sectors, write operations on pages.

Attention: Flash write and erase operations must be executed to the non-cacheable address space.

8.4.6 Modes of Operation

A Flash module can be in one of the following states:

- Active (normal) mode.
- Sleep mode (see **Section 8.6.2**).

In sleep mode write and read accesses to all Flash ranges of this PMU are refused with a bus error.

When the Flash module is in active mode the Flash bank can be in one of these modes:

- Read mode.
- Command mode.

In read mode a Flash bank can be read and command sequences are interpreted. In read mode a Flash bank can additionally enter page mode which enables it to receive data for programming.

In command mode an operation is performed. During its execution the Flash bank reports BUSY in **FSR**. In this mode read accesses to this Flash bank are refused with a bus error. At the end of an operation the Flash bank returns to read mode and BUSY is cleared. Only operations with a significant duration (shown in the command documentation) set BUSY.

Register read and write accesses are not affected by these modes.

8.4.7 Command Sequences

All Flash operations except read are performed with command sequences. When a Flash bank is in read mode or page mode all write accesses to its reserved address

Flash and Program Memory Unit (PMU)

range are interpreted as command cycle belonging to a command sequence. Write accesses to a busy bank cause a sequence error (SQER).

Attention: *For the proper execution of the command sequences and the triggered operations f_{CPU} must be equal or above 1 MHz.*

Command sequences consist of 1 to 6 command cycles. The command interpreter checks that a command cycle is correct in the current state of command interpretation. Else a SQER is reported.

When the command sequence is accepted the last command cycle finishes read mode and the Flash bank transitions into command mode.

These write accesses must be single transfers and must address the non-cacheable address range.

Generally when the command interpreter detects an error it reports a sequence error by setting **FSR.SQER**. Then the command interpreter is reset and a page mode is left. The next command cycle must be the 1st cycle of a command sequence. The only exception is "Enter Page Mode" when a bank is already in page mode (see below).

8.4.7.1 Command Sequence Definitions

Table 8-4 gives an overview of the supported command sequence, with the following nomenclature:

The parameter "addr" can be one of the following:

- **CCCC_H**: The "addr" must point into the bank that performs the operation. The last 16 address bits must match CCCC_H. It is recommended to use as address the base address of the bank incremented by CCCC_H.
- **PA**: Absolute start address of the Flash page.
- **UCPA**: Absolute start address of a user configuration block page.
- **SA**: Absolute start address of a Flash sector. Allowed are the PFLASH sectors Sx.
- **PSA**: Absolute start address of a physical sector. Allowed are the PFLASH physical sectors PSx.
- **UCBA**: Absolute start address of a user configuration block.

The parameter "data" can be one of the following:

- **WD**: 32-bit write data to be loaded into the page assembly buffer.
- **xxYY**: 8-bit write data as part of a command cycle. Only the byte "YY" is used for command interpretation. The higher order bytes "xx" are ignored.
 - **xx5y**: Specific case for "YY". The "y" can be "0_H" for selecting the PFLASH bank.
- **UL**: User protection level (xxx0_H or xxx1_H for user levels 0 and 1).
- **PWx**: 32-bit password.

Flash and Program Memory Unit (PMU)

Command Sequence Overview Table

The **Table 8-4** summarizes all commands sequences. The following sections describe each command sequence in detail.

Table 8-4 Command Sequences for Flash Control

Command Sequence		1. Cycle	2. Cycle	3. Cycle	4. Cycle	5. Cycle	6. Cycle
Reset to Read	Address Data	.5554 ..xxF0					
Enter Page Mode	Address Data	.5554 ..xx5y					
Load Page	Address Data	.55F0 WD	.55F4 WD				
Write Page	Address Data	.5554 ..xxAA	.AAA8 ..xx55	.5554 ..xxA0	PA ..xxAA		
Write User Configuration Page	Address Data	.5554 ..xxAA	.AAA8 ..xx55	.5554 ..xC0	UCPA ..xxAA		
Erase Sector	Address Data	.5554 ..xxAA	.AAA8 ..xx55	.5554 ..xx80	.5554 ..xxAA	.AAA8 ..xx55	SA ..xx30
Erase Physical Sector ¹⁾	Address Data	.5554 ..xxAA	.AAA8 ..xx55	.5554 ..xx80	.5554 ..xxAA	.AAA8 ..xx55	PSA ..xx40
Physical Sector Repair ¹⁾	Address Data	.5554 ..xxAA	.AAA8 ..xx55	.5554 ..xx80	.5554 ..xxAA	.AAA8 ..xx55	PSA ..xx40
Erase User Configuration Block	Address Data	.5554 ..xxAA	.AAA8 ..xx55	.5554 ..xx80	.5554 ..xxAA	.AAA8 ..xx55	UCBA ..xxC0
Disable Sector Write Protection	Address Data	.5554 ..xxAA	.AAA8 ..xx55	.553C UL	.AAA8 PW 0	.AAA8 PW 1	.5558 ..xx05
Disable Read Protection	Address Data	.5554 ..xxAA	.AAA8 ..xx55	.553C ..xx00	.AAA8 PW 0	.AAA8 PW 1	.5558 ..xx08
Resume Protection	Address Data	.5554 ..xx5E					
Clear Status	Address Data	.5554 ..xxF5					

1) Alternate operation as Erase Physical Sector or Sector Soft Repair, depending on the configuration of PROCON1.PSR.

Flash and Program Memory Unit (PMU)**Reset to Read**

This function resets the command interpreter to its initial state (i.e. the next command cycle must be the 1st cycle of a sequence). A page mode is aborted.

This command is the only one that is accepted without generating a SQER when the command interpreter has already received command cycles of a different sequence but is still not in command mode. Thus “Reset to Read” can cancel every command sequence before its last command cycle has been received.

The error flags of **FSR** (PFOPER, SQER, PROER, PFDBER, ORIER, VER) are cleared. The flags can be also cleared in the status registers without command sequence.

If any Flash bank is busy this command is executed but the flag SQER is set.

Enter Page Mode

The PFLASH enters page mode. The selection of the PFLASH assembly buffer (256 bytes) is additionally done by the parameter “ $y_H = 0_H$ ”.

The write pointer of the page assembly buffer is set to 0, its previous content is maintained.

The page mode is signalled by the flag PAGE_x in the FSR.

If a new “Enter Page Mode” command sequence is received while any Flash bank is already in page mode SQER is set but this sequence is correctly executed (i.e. in this case the command interpreter is not reset).

Load Page

Loads the data “WD” into the page assembly buffer. It is required to transfer 64-bit with two consecutive 32-bit data transfers, first addressing the low word with 55F0_H, followed by the high word with 55F4_H. The 64-bit are then transferred to the assembly buffer and the write pointer is incremented to the next position.

32 “Load Page” operations are required to fill the assembly buffer for one 256 byte page.

The addressed bank must be in page mode, else SQER is issued.

If “Load Page” is called more often than necessary for filling the page SQER is issued and if configured an interrupt is triggered. The overflow data is discarded. The page mode is not left.

Write Page

This function starts the programming process for one page with the data transferred previously by “Load Page” commands. Upon entering command mode the page mode is finished (indicated by clearing the corresponding PAGE flag) and the BUSY flag of the bank is set.

Flash and Program Memory Unit (PMU)

This command is refused with SQER when the addressed Flash bank is not in page mode.

SQER is also issued when PA addresses an unavailable Flash range or when PA does not point to a legal page start address.

If after “Enter Page Mode” too few data or no data was transferred to the assembly buffer with “Load Page” then “Write Page” programs the page but sets SQER. The missing data is programmed with the previous content of the assembly buffer.

When the page “PA” is located in a sector with active write protection or the Flash module has an active global read protection the execution fails and PROER is set.

Write User Configuration Page

As for “Write Page”, except that the page “UCPA” is located in a user configuration block. This changes the Flash module’s protection configuration.

When the page “UCPA” is located in an UCB with active write protection or the Flash module has an active global read protection the execution fails and PROER is set.

When UCPA is not the start address of a page in a valid UCB the command fails with SQER.

Erase Sector

The sector “SA” is erased.

SQER is returned when SA does not point to the base address of a correct sector (as specified at the beginning of this section) or to an unavailable sector.

When SA has an active write protection or the Flash module has an active global read protection the execution fails and PROER is set.

Erase Physical Sector

The physical sector “PSA” is erased.

SQER is returned when PSA does not point to the base address of a correct sector (as specified at the beginning of this section) or an unavailable sector.

When PSA has an active write protection or the Flash module has an active global read protection the execution fails and PROER is set.

*Note: The same command sequence can alternatively be used as Erase Physical Sector or Sector Soft Repair, depending on the configuration of **PROCON1.PSR**.*

Physical Sector Repair

The physical sector “PSA = PS4” is repaired.

This command sequence must be used when an erase operation on a logical sector inside the physical sector PS4 has been aborted by a reset - this may render parts or the

Flash and Program Memory Unit (PMU)

whole physical sector unreadable. This command sequence repairs the corrupted logical sector.

This command sequence is required to run an EEPROM emulation algorithm that cycles the logical sectors S4..S7 of PS4.

The Physical Sector Repair command sequence will be ignored if any other sector than “PSA4” is addressed. No SQER is triggered in this case.

Note: The same command sequence can alternatively be used as Erase Physical Sector or Sector Soft Repair, depending on the configuration of PROCON1.PSR.

Erase User Configuration Block

The addressed user configuration block “UCB” is erased.

When the UCB has an active write protection or the Flash module has an active global read protection the execution fails and PROER is set.

The command fails with SQER when UCBA is not the start address of a valid UCB.

Disable Sector Write Protection

The sector write protection belonging to user level “UL” is temporarily disabled by setting FSR.WPRODIS when the passwords PW0 and PW1 match their configured values in the corresponding UCB.

The command fails by setting PROER when any of PW0 and PW1 does not match. In this case until the next application reset all further calls of “Disable Sector Write Protection” and “Disable Read Protection” fail with PROER independent of the supplied password.

Disable Read Protection

The Flash module read protection including the derived module wide write protection are temporarily disabled by setting FSR.RPRODIS when the passwords PW0 and PW1 match their configured values in the UCB0.

The command fails by setting PROER when any of PW0 and PW1 does not match. In this case until the next application reset all further calls of “Disable Sector Write Protection” and “Disable Read Protection” fail with PROER independent of the supplied password.

Resume Protection

This command clears all FSR.WPRODISx and the FSR.RPRODIS effectively enabling again the Flash protection as it was configured.

A FSR.WPRODISx is not cleared when corresponding UCBx is not in the “confirmed” state (see [Section 8.4.8.1](#)).

Clear Status

The flags FSR.PROG and FSR.ERASE and the error flags of **FSR** (PFOPER, SQER, PROER, PFDBER, ORIER, VER) are cleared. These flags can be also cleared in the status registers without command sequence.

When any Flash bank is busy this command fails by setting additionally SQER.

8.4.7.2 Flash Page Programming Example

Figure 8-3 shows the basic flow of command sequences to program a Flash page. All commands are write accesses to the non-cached Flash address space. E.g. the “Enter Page Mode” command can be executed by a write access to the address 0C00 5554_H with the data 0000 0050_H.

In the first step the Flash bank is switched to Page Mode. Only in this mode the other command sequences are accepted.

Then the data is loaded in the assembly buffer with a series of “Load Page” commands.

The “Write Page” command triggers the actual write operation, transferring the 256 bytes data from the assembly buffer to the addressed page in the Flash.

FSR.PROG is set with the last cycle of the “Write Page” command sequence, indicating that a program operation is started.

While this write operation is processed the Flash bank is not accessible, indicated by the **FSR.PBUSY** flag.

In every step the **FSR.SQER** flag provides a direct feedback on the successful execution of the command sequence.

After the program operation has been completed successfully, the “sticky” status flags like **FSR.PROG** can be cleared by the “Clear Status” sequence, before the next operation is started.

Flash and Program Memory Unit (PMU)

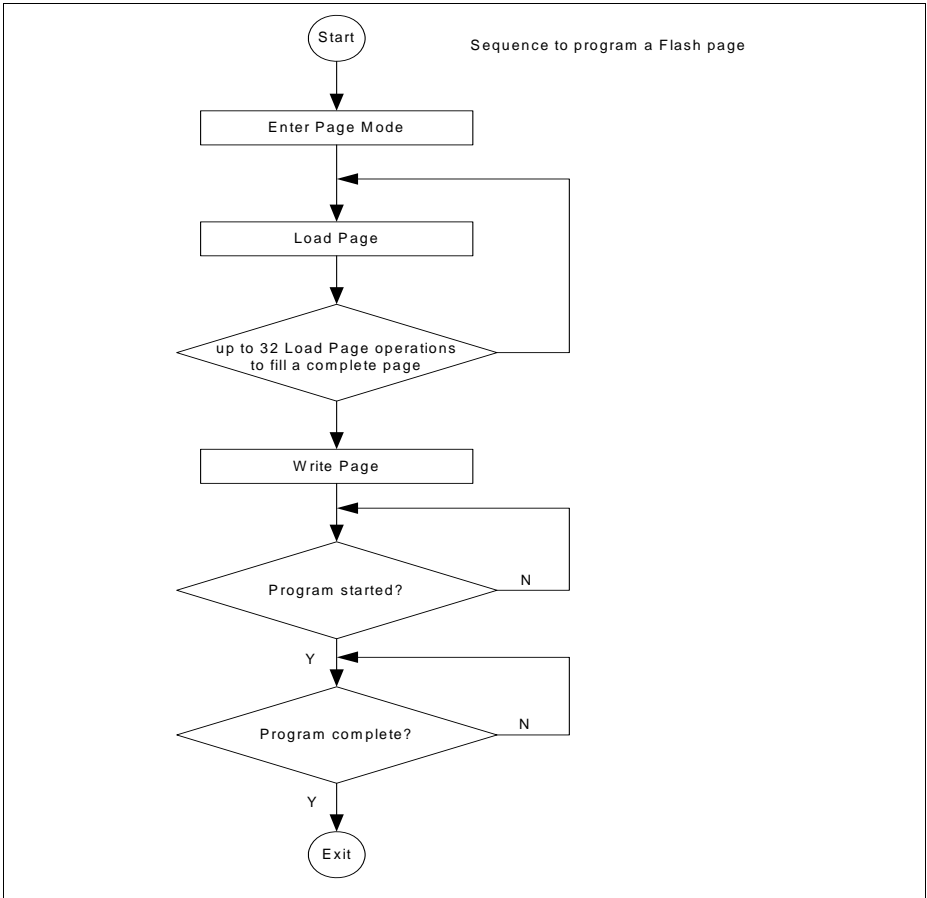


Figure 8-3 Basic Flash Program Sequence

8.4.8 Flash Protection

The Flash memory can be read and write protected. The protection is configured by programming the User Configuration Blocks “UCB”.

For an effective IP protection the Flash read protection must be activated. This ensures system wide that the Flash cannot be read from external or changed without authorization.

8.4.8.1 Configuring Flash Protection in the UCB

As indicated above the effective protection is determined by the content of the **Protection Configuration Indication** PROCON0–2 registers. These are loaded during startup from the UCB0–2. Each UCB comprises 1 Kbyte of Flash organized in 4 UC pages of 256 bytes. The UCBs have the following structure:

Table 8-5 UCB Content

UC Page	Bytes	UCB0	UCB1	UCB2
0	[3:0]	PROCON0	PROCON1	PROCON2
	[7:4]	unused	unused	unused
	[11:8]	PROCON0 (copy)	PROCON1 (copy)	PROCON2 (copy)
	[15:12]	unused	unused	unused
	[19:16]	PW0 of User 0	PW0 of User 1	unused
	[23:20]	PW1 of User 0	PW1 of User 1	unused
	[27:24]	PW0 of User 0 (copy)	PW0 of User 1 (copy)	unused
	[31:28]	PW1 of User 0 (copy)	PW1 of User 1 (copy)	unused
	others	unused	unused	unused
1		unused	unused	BMI and configuration data (details in Startup Mode chapter)
2	[3:0]	confirmation code	confirmation code	confirmation code
	[11:8]	confirmation code (copy)	confirmation code (copy)	confirmation code (copy)
	others	unused	unused	unused
3	unused	unused	unused	unused

Flash and Program Memory Unit (PMU)

If the confirmation code field is programmed with 8AFE 15C3_H, the UCB content is “confirmed” otherwise it is “unconfirmed”. The status flags **FSR.PROIN**, **FSR.RPROIN** and **FSR.WPROIN0–2** indicate this confirmation state:

- **FSR.PROIN**: set when any UCB is in the confirmed state.
- **FSR.RPROIN**: set when **PROCON0.RPRO** is ‘1’ and the UCB0 is in “confirmed” state.
- **FSR.WPROIN0–2**: set when their UCB0–2 is in “confirmed” state.

An UCB can be erased with the command “Erase User Configuration Block”. An UCB page can be programmed with the command “Write User Configuration Page”. These commands fail with PROER when the UCB is write-protected.

An UCB is write-protected if:

- UCB0: (**FSR.RPROIN** and not **FSR.RPRODIS**) or (**FSR.WPROIN0** and not **FSR.WPRODIS0**)
- UCB1: **FSR.WPROIN1** and not **FSR.WPRODIS1**.
- UCB2: **FSR.WPROIN2**

So when the UCB2 is in the “confirmed” state its protection can not be changed anymore. Therefore this realizes a one-time programmable protection.

Changing UCBs

The protection installation is modified by erasing and programming the UCBs with dedicated command sequences, described in [Section 8.4.7.1](#). These operations need to be performed with care as described in the following.

Aborting an “Erase UC Block” operation (e.g. due to reset or power failure) must be avoided at all means, as it can result in an unusable device.

UCBs are logical sectors, and as such the allowed number of program/erase cycles of the UCBs must not be exceeded. Over-cycling the UCBs can also lead to an unusable device.

The installation of the protection and its confirmation on different pages of the UCB offers the possibility to check the installation before programming the confirmation. First the protection needs to be programmed, then an application reset must be triggered to trigger the reading of the UCBs by the PMU and after that the protection can be verified (e.g. “Disable ... Protection” to check the password and by checking **PROCONs** and **FCON**). The application reset is inevitable because the PMU reads the UCBs only during the startup phase.

8.4.8.2 Flash Read Protection

Read protection can be activated for the whole Flash module.

Read Protection Status

A read access to PFLASH fails with bus error under the following conditions:

- Code fetch: **FCON.DCF** and **FCON.RPA**.
- Data read: **FCON.DDF** and **FCON.RPA**.

The read protection bit **FCON.RPA** is determined during startup by the protection configuration of UCBO. It can be temporarily modified by the command sequences “Disable Read Protection” and “Resume Protection” which modify **FSR.RPRODIS**. **FCON.RPA** is determined by the following equation:

- **FCON.RPA** = **PROCON0.RPRO** and not **FSR.RPRODIS**.

The bits **FCON.DDF** and **FCON.DCF** are initialized by the startup software depending on the configured protection and the startup mode. They can also be directly modified by the user software under conditions noted in the description of **FCON**.

Initializing Read Protection

Installation of read protection is performed with the “Write User Configuration Page” operation, controlled by the user 0. With this command, user 0 writes the protection configuration bits RPRO, and the two 32-bit keywords into the UCBO page 0. Additionally, with a second “Write User Configuration Page” command, a special 32-bit confirmation (lock-) code is written into the UCBO page 2. Only this confirmation code enables the protection and thus the keywords. The confirmation write operation to the second wordline of the User Configuration Block shall be executed only after check of keyword-correctness (with command “Disable Read Protection” after next reset). The confirmed state and thus the installation of protection is indicated with the FSR-bit PROIN in Flash Status Register FSR and for read protection with bit RPROIN in FSR. If read protection is not correctly confirmed and thus not enabled, the bits PROIN and RPROIN in the FSR are not set. The configured read protection as fetched from UCBO is indicated in the protection configuration register **PROCON0**.

For safety of the information stored in the UCB pages, all keywords, lock bits and the confirmation code are stored two-times in the two wordlines. In case of a disturbed original data detected during ramp up, its copy is used **FSR**. Layout of the four UC pages belonging to the user’s UC block is shown in **Table 8-5**, the command “Write User Configuration Page” is described in **Section 8.4.7.1**.

Disabling Read Protection

With the command sequence “Disable Read Protection” short-term disabling of read protection is possible. This command disables the Flash protection (latest until next reset) for user controlled erase and re-program operations as well as for clearing of DCF and DDF control bits after external program execution. The “Disable Read Protection” command sequence is a protected command, which is only processed by the command state machine, if the included two passwords are identical to the two keywords of user 0.

Flash and Program Memory Unit (PMU)

The disabled state of read protection is controlled with the **FCON.RPA**=‘0’ and indicated in the Flash Status Register **FSR** with the RPRODIS bit (see [Section 8.7.3.1](#)). As long as read protection is disabled (and thus not active), the **FCON**-bits DDF and DCF can be cleared.

Resumption of read protection after disablement is performed with the “Resume Read/Write Protection” command. After execution of this single cycle command, read protection (if installed) is again active, indicated by the **FCON** bit RPA=‘1’.

Generally, Flash read protection will remain installed as long as it is confirmed (locked) in the User Configuration Block 0. Erase of UC block and re-program of UC pages can be performed up to 4 times. But note, after execution of the Erase UC block command (which is protected and therefore requires the preceding disable command with the user’s specific passwords), all keywords and all protection installations of user 0 are erased; thus, the Flash is no more read protected (beginning with next reset) until re-programming the UC pages. But the division and separation of the protection configuration data and of the confirmation data into two different UCB-wordlines guarantees, that a disturb of keywords can be discovered and corrected before the protection is confirmed. For this reason, the command sequence “Disable Read Protection” can also be used when protection is programmed (configured) but not confirmed; wrong keywords are then indicated by the error flag PROER.

Read protection can be combined with sector specific write protection. In this case, after execution of the command ‘Disable Read Protection’ only those sectors are unlocked for write accesses, which are not separately write protected.

8.4.8.3 Flash Write and OTP Protection

A range of Flash can be write protected by several means:

- The complete PFLASH can be write protected by the read protection.
- Groups of sectors of PFLASH can be write-protected by three different “users”, i.e. UCBs:
 - UCB0: Write protection that can be disabled with the password of UCB0.
 - UCB1: Write protection that can be disabled with the password of UCB1.
 - UCB2: Write protection that can not be disabled anymore (ROM or OTP function: “One-Time Programmable”).

Write and OTP Protection Status

An active write protection is indicated by WPROIN bits in **FSR** register. It causes the program and erase command sequences to fail with a PROER.

A range “x” (i.e. a group of sectors, see [PROCON0](#)) of the PFLASH is write protected if any of the following conditions is true:

- **FCON.RPA**
- **PROCON2.SxROM**

Flash and Program Memory Unit (PMU)

- **PROCON0**.SxL and not(**FSR**.WPRODIS0)
- **PROCON1**.SxL and not(**PROCON0**.SxL) and not(**FSR**.WPRODIS1)

Thus with the password of UCB0 the write protection of sectors protected by user 0 and user 1 can be disabled, however with the password of UCB1 only those sectors that are only protected by user 1. The write protection of user 2 (OTP) can be obviously not disabled. The global write protection caused by the read protection can be disabled as described above by using the password of UCB0 to disable the read protection.

Initialization of Write and OTP Protection

Installation of write protection is performed with the “Write User Configuration Page” operation, controlled by the user. With this command, the user defines and writes into the UCBx page 0 the write protection configuration bits for all sectors, which shall be locked by the specific user, and the user-specific two keywords (not necessary for user 2). The position of sector lock bits is identical as defined for the PROCON registers ([Section 8.7.3.5](#)). The correctness of keywords shall then (after next reset) be checked with the command ‘Disable Sector Write Protection’, which delivers a protection error PROER in case of wrong passwords. Only if the keywords are correct, the special 32-bit confirmation code must be written into the page 2 of UCBx with a second “Write User Configuration Page” command. Only this confirmation code enables the write protection of the User Control Block UCBx, and only in this case the installation bit(s) in **FSR** is (are) set during ramp up.

Note: If the write protection is configured in the user’s UCB page 0 but not confirmed via page 2 (necessary for check of keywords), the state after next reset is as follows:

- The selected sector(s) are protected (good for testing of protection, but not OTP!)
- The respective PROCON register is set accordingly (also for OTP!)
- The UCBx is not protected, thus it can be erased without passwords
- The related WPROINx bit in **FSR** is not set
- The Disable Write Protection command sets the WPRODISx bit
- The Resume command does not clear the WPRODISx bit.

The structure and layout of the three UC blocks is shown in [Table 8-3](#) below, the command “Write User Configuration Page” is described in [Section 8.4.7.1](#).

Disabling Write Protection (not applicable to OTP)

With the command sequence “Disable Sector Write Protection” short-term disabling of write protection for user 0 or user 1 is possible. This command unlocks temporarily all locked sectors belonging to the user. The “Disable Sector Write Protection” command sequence is a protected command, which is only processed by the command state machine, if the included two passwords are correct. The disabled state of sector protection is indicated in the Flash Status Register **FSR** with the WPRODIS bit of the user 0 or/and user 1 (see [Section 8.7.3.1](#)). For user 2 who owns the sectors with ROM functionality, a disablement of write protection and thus re-programming is not possible.

Flash and Program Memory Unit (PMU)

Resumption of write protection after disablement is performed with the “Resume Read/Write Protection” command, which is identical for user 0 and user 1.

Generally, sector write protection will remain installed as long as it is configured and confirmed in the User Configuration Block belonging to the user. Erase of UC block and re-program of UC pages can be performed up to 4 times, for user 0 and user 1 only. But note, after execution of the Erase UC block command (which is still protected and therefore requires the preceding disablement of write protection with the user’s passwords), the complete protection configuration including the keywords of the specific user (not user 2) is erased; thus, the sectors belonging to the user are unprotected until the user’s UC pages are re-programmed. Only exception: sectors protected by user 2 are locked for ever because the UCB2 can no more be erased after installation of write protection in UCB2.

8.4.8.4 System Wide Effects of Flash Protection

An active Flash read protection needs to be respected in the complete system.

The startup software (SSW) checks if the Flash read protection is active in the PMU, if yes:

- If the selected boot mode executes from internal PFLASH.
 - The SSW clears the DCF and DDF.
 - The SSW leaves the debug interface locked.
- If the selected boot mode does not execute from internal PFLASH:
 - The SSW either leaves DCF and DDF set or actively sets them again in the PMU after evaluating the configuration sector.
 - The debug interface is unlocked.

If the read protection is inactive in the PMU the DCF and DDF flags are cleared by the SSW and the debug interface is unlocked.

Note: Full Flash analysis of an FAR device is only possible when the customer has removed all installed protections or delivers the necessary passwords with the device. As the removal of an OTP protection in UCB2 is not possible the OTP protection inevitably limits analysis capabilities.

8.4.9 Data Integrity and Safety

The data in Flash is stored with error correcting codes “ECC” in order to protect against data corruption. The healthiness of Flash data can be checked with margin checks.

8.4.9.1 Error-Correcting Code (ECC)

The data in the PFLASH is stored with ECC codes. These are automatically generated when the data is programmed. When data is read these codes are evaluated. Data in

Flash and Program Memory Unit (PMU)

PFLASH uses an ECC code with SEC-DED (Single Error Correction, Double Error Detection) capabilities. Each block of 64 data bits is accompanied with 8 ECC bits.

Standard PFLASH ECC

In the standard PFLASH ECC the 8-bit ECC value is calculated over 64 data bits. An erased data block (all bits '0') has an ECC value of 00_H. Therefore an erased sector is free of ECC errors. A data block with all bits '1' has an ECC value of FF_H.

The ECC is automatically generated when programming the PFLASH.

The ECC is automatically evaluated when reading data.

This algorithm has the following capabilities:

- Single-bit error:
 - Is noted in **FSR**.PFSBER.
 - Data and ECC value are corrected.
 - Interrupt is triggered if enabled with **FCON**.PFSBERM.
- Double-bit error:
 - Is noted in **FSR**.PFDBER.
 - Causes a bus error if not disabled by **MARP**.TRAPDIS.
 - Interrupt is triggered if enabled with **FCON**.PFDBERM. This interrupt shall only be used for margin check, when the bus error is disabled.

8.4.9.2 Margin Checks

The Flash memory offers a “margin check feature”: the limit which defines if a Flash cell is read as logic '0' or logic '1' can be shifted. This is controlled by the register **MARP**. The Margin Control Register **MARP** is used to change the margin levels for read operations to find problematic array bits. The array area to be checked is read with more restrictive margins. “Problematic” bits will result in a single or double-bit error that is reported to the CPU by an error interrupt or a bus error trap. The double-bit error trap can be disabled for margin checks and also redirected to an error interrupt.

After changing the read margin at least $t_{FL_MarginDel}$ have to be waited before reading the affected Flash module. During erase or program operation only the standard (default) margins are allowed.

8.5 Service Request Generation

Access and/or operational errors (e.g. wrong command sequences) may be reported to the user by interrupts, and they are indicated by flags in the Flash Status Register **FSR**. Additionally, bus errors may be generated resulting in CPU traps.

8.5.1 Interrupt Control

The PMU and Flash module supports immediate error and status information to the user by interrupt generation. One CPU interrupt request is provided by the Flash module.

The Flash interrupt can be issued because of following events:

- End of busy state: program or erase operation finished
- Operational error (OPER): program or erase operation aborted
- Verify error (VER): program or erase operation not correctly finished
- Protection error
- Sequence error
- Single-bit error: corrected read data from PFLASH delivered
- Double-bit error in Program Flash.

Note: In case of an OPER or VER error, the error interrupt is issued not before the busy state of the Flash is deactivated.

The source of interrupt is indicated in the Flash Status Register **FSR** by the error flags or by the PROG or ERASE flag in case of end of busy interrupt. An interrupt is also generated for a new error event, even if the related error flag is still set from a previous error interrupt.

Every interrupt source is masked (disabled) after reset and can be enabled via dedicated mask bits in the Flash Configuration Register **FCON**.

8.5.2 Trap Control

CPU traps are triggered because of bus errors, generated by the PMU in case of erroneous Flash accesses. Bus errors are generated synchronously to the bus cycle requesting the not allowed Flash access or the disturbed Flash read data. Bus errors are issued because of following events:

- Not correctable double-bit error of 64-bit read data from PFLASH (if not disabled for margin check)
- Not allowed write access to read only register (see [Table 8-11](#))
- Not allowed write access to Privileged Mode protected register (see [Table 8-11](#))
- Not allowed data or instruction read access in case of active read protection
- Access to not implemented addresses within the register or array space.
- Read-modify-write access to the Flash array.

Write accesses to the Flash array address space are interpreted as command cycles and initiate not a bus error but a sequence error if the address or data pattern is not correct. However, command sequence cycles, which address a busy Flash bank, are serviced with busy cycles, not with a sequence error.

If the trap event is a double-bit error in PFLASH, it is indicated in the **FSR**. With exception of this error trap event, all other trap sources cannot be disabled within the PMU.

Flash and Program Memory Unit (PMU)

*Note: A double-bit error trap during margin check can be disabled (via **MARP** register) and redirected to an interrupt request.*

8.5.3 Handling Errors During Operation

The previous sections described shortly the functionality of “error indicating” bits in the flash status register **FSR**. This section elaborates on this with more in-depth explanation of the error conditions and recommendations how these should be handled by customer software. This first part handles error conditions occurring during operation (i.e. after issuing command sequences) and the second part (**Section 8.5.3.6**) error conditions detected during startup.

8.5.3.1 SQER “Sequence Error”

Fault conditions:

- Improper command cycle address or data, i.e. incorrect command sequence.
- New “Enter Page” in Page Mode.
- “Load Page” and not in Page Mode.
- “Load Page” results in buffer overflow.
- First “Load Page” addresses 2. word.
- “Write Page” with buffer underflow.
- “Write Page” and not in Page Mode.
- “Write Page” to wrong Flash type.
- Byte transfer to password or data.
- “Clear Status” or “Reset to Read” while busy¹⁾.
- Erase UCB with wrong UCBA.

New state:

Read mode is entered with following exceptions:

- “Enter Page” in Page Mode re-enters Page Mode.
- “Write Page” with buffer underflow is executed.
- After “Load Page” causing a buffer overflow the Page Mode is not left, a following “Write Page” is executed.

Proposed handling by software:

Usually this bit is only set due to a bug in the software. Therefore in development code the responsible error tracer should be notified. In production code this error should not occur. It is however possible to clear this flag with “Clear Status” or “Reset to Read” and simply issue the corrected command sequence again.

With a SQER after the “Write Page” sequence it is possible to verify the written data in the Flash. It is sufficient to clear the flag with the “Clear Status” command if the written

1) When the command addresses the busy Flash bank, the access is serviced with busy cycles.

Flash and Program Memory Unit (PMU)

data is correct. If the written data is wrong, the whole sector must be erased and reprogrammed.

8.5.3.2 PFOPER “Operation Error”Fault conditions:

ECC double-bit error detected in Flash module internal SRAM during a program or erase operation in PFLASH. This can be a transient event due to alpha-particles or illegal operating conditions or it is a permanent error due to a hardware defect. This situation will practically not occur.

Attention: these bits can also be set during startup (see [Section 8.5.3.6](#)).

New state:

The Flash operation is aborted, the BUSY flag is cleared and read mode is entered.

Proposed handling by software:

The flag should be cleared with “Clear Status”. The last operation can be determined from the PROG and ERASE flags. In case of an erase operation the affected physical sector must be assumed to be in an invalid state, in case of a program operation only the affected page. Other physical sectors can still be read. New program or erase commands must not be issued before the next reset.

Consequently a reset must be performed. This performs a new Flash ramp up with initialization of the microcode SRAM. The application must determine from the context which operation failed and react accordingly. Mostly erasing the addressed sector and re-programming its data is most appropriate. If a “Program Page” command was affected and the sector can not be erased the wordline could be invalidated if needed by marking it with all-one data and the data could be programmed to another empty wordline.

Only in case of a defective microcode SRAM the next program or erase operation will incur again this error.

Note: Although this error indicates a failed operation it is possible to ignore it and rely on a data verification step to determine if the Flash memory has correct data. Before re-programming the Flash the flow must ensure that a new reset is applied.

Note: Even when the flag is ignored it is recommended to clear it. Otherwise all following operations — including “sleep” — could trigger an interrupt even when they are successful (see [Section 8.5.1](#), interrupt because of operational error).

8.5.3.3 PROER “Protection Error”Fault conditions:

- Password failure.
- Erase/Write to protected sector.
- Erase UCB and protection active.

Flash and Program Memory Unit (PMU)

- Write UC-Page to protected UCB.

Attention: a protection violation can even occur when a protection was not explicitly installed by the user. This is the case when the Flash startup detects an error and starts the user software with read-only Flash (see [Section 8.5.3.6](#)). Trying to change the Flash memory will then cause a PROER.

New state:

Read mode is entered. The protection violating command is not executed.

Proposed handling by software:

Usually this bit is only set during runtime due to a bug in the software. In case of a password failure a reset must be performed in the other cases the flag can be cleared with “Clear Status” or “Reset to Read”. After that the corrected sequence can be executed.

8.5.3.4 VER “Verification Error”

Fault conditions:

This flag is a warning indication and not an error. It is set when a program or erase operation was completed but with a suboptimal result. This bit is already set when only a single bit is left over-erased or weakly programmed which would be corrected by the ECC anyhow.

However, excessive VER occurrence can be caused by operating the Flash out of the specified limits, e.g. incorrect voltage or temperature. A VER after programming can also be caused by programming a page whose sector was not erased correctly (e.g. aborted erase due to power failure).

Under correct operating conditions a VER after programming will practically not occur. A VER after erasing is not unusual.

Attention: this bit can also be set during startup (see [Section 8.5.3.6](#)).

New state:

No state change. Just the bit is set.

Proposed handling by software:

This bit can be ignored. It should be cleared with “Clear Status” or “Reset to Read”. In-spec operation of the Flash memory must be ensured.

If the application allows (timing and data logistics), a more elaborate procedure can be used to get rid of the VER situation:

- VER after program: erase the sector and program the data again. This is only recommended when there are more than 3 program VERs in the same sector. When programming the Flash in field ignoring program VER is normally the best solution because its most likely cause are violated operating conditions. Take care that never a sector is programmed in which the erase was aborted.

Flash and Program Memory Unit (PMU)

- VER after erase: the erase operation can be repeated until VER disappears. Repeating the erase more than 3 times consecutively for the same sector is not recommended. After that it is better to ignore the VER, program the data and check its readability. Again its most likely cause are violated operating conditions. Therefore it is recommended to repeat the erase at most once or ignore it altogether.

For optimizing the quality of Flash programming see the following section about handling single-bit ECC errors.

Note: Even when this flag is ignored it is recommended to clear it. Otherwise all following operations — including “sleep” — could trigger an interrupt even when they are successful (see [Section 8.5.1](#), interrupt because of verify error).

8.5.3.5 PFSBER/DFSBER “Single-Bit Error”Fault conditions:

When reading data or fetching code from PFLASH the ECC evaluation detected a single-bit error (“SBE”) which was corrected.

This flag is a warning indication and not an error. A certain amount of single-bit errors must be expected because of known physical effects.

New state:

No state change. Just the bit is set.

Proposed handling by software:

This flag can be used to analyze the state of the Flash memory. During normal operation it should be ignored. In order to count single-bit errors it must be cleared by “Clear Status” or “Reset to Read” after each occurrence¹⁾.

Usually it is sufficient after programming data to compare the programmed data with its reference values ignoring the SBE bits. When there is a comparison error the sector is erased and programmed again.

When programming the PFLASH (end-of-line programming or SW updates) customers can further reduce the probability of future read errors by performing the following check after programming:

- Change the read margin to “high margin 0”.
- Verify the data and count the number of SBEs.
- When the number of SBEs exceeds a certain limit (e.g. 10 in 2 Mbyte) the affected sectors could be erased and programmed again.
- Repeat the check for “high margin 1”.

1) Further advice: remember that the ECC is evaluated when the data is read from the PMU. When counting single-bit errors use always the non-cached address range otherwise the error count can depend on cache hit or miss and it refers to the complete cache line. As the ECC covers a block of 64 data bits take care to evaluate the **FSR** only once per 64-bit block.

Flash and Program Memory Unit (PMU)

- Each sector should be reprogrammed at most once, afterwards SBEs can be ignored.

Due to the specificity of each application the appropriate usage and implementation of these measures (together with the more elaborate VER handling) must be chosen according to the context of the application.

8.5.3.6 Handling Flash Errors During Startup

During startup, a fatal error during Flash ramp up forces the Firmware to terminate the startup process and to end in the Debug Monitor Mode (see Firmware chapter).

The reason for a failed Flash startup can be a hardware error or damaged configuration data.

FSR bits set after startup are of informative warning nature.

FSR.PFOPER can indicate a problem of a program/erase operation before the last system reset or an error when restoring the Flash module internal SRAM content after the last reset. In both cases it is advised to clear the flag with the command sequence “Clear Status” and trigger a system reset. If the error shows up again it is an indication for a permanent fault which will limit the Flash operation to read accesses. Under this condition program and erase operations are forbidden (but not blocked by hardware!).

8.6 Power, Reset and Clock

The following chapters describe the required power supplies, the power consumption and its possible reduction, the control of Flash Sleep Mode and the basic control of Reset.

8.6.1 Power Supply

The Flash module uses the standard V_{DDP} I/O power supply to generate the voltages for both read and write functions. Internally generated and regulated voltages are provided for the program and erase operations as well as for read operations. The standard V_{DDC} is used for all digital control functions.

8.6.2 Power Reduction

The “Flash Sleep Mode” can be used to drastically reduce power consumption while the Flash is not accessed for longer periods of time.

The “Idle Read Path” slightly reduces the dynamic power consumption during normal operation with marginal impact on the Flash read performance.

Flash and Program Memory Unit (PMU)**Flash Sleep Mode**

As power reduction feature, the Flash module provides the Flash Sleep mode which can be selected by the user individually for the Flash. The Sleep mode can be requested by:

- Programming 1_B to the bit **FCON.SLEEP**.
- “External” sleep mode by the SCU (see “Flash Power Control” in the SCU). Only executed by the Flash when **FCON.ESLDIS** = 0_B .

Attention: *f_{CPU} must be equal or above 1 MHz when Sleep mode is requested until the Sleep mode is indicated in **FSR.SLM**, and when a wake-up request is triggered, until **FSR.PBUSY** is cleared.*

The requested Sleep mode is only taken if the Flash is in idle state and when all pending or active requests are processed and terminated. Only then, the Flash array performs the ramp down into the Sleep mode: the sense amplifiers are switched off and the voltages are ramped down.

During ramp down to Sleep mode **FSR.PBUSY** is set.

As long as the Flash is in Sleep mode, this state is indicated by the bit **FSR.SLM**. The **FSR.PBUSY** stays set as well.

Wake-up from sleep is controlled with clearing of bit **FCON.SLEEP**, if selected via this bit, or wake-up is initiated by releasing the “external” sleep signal from SCU. After wake-up, the Flash enters read mode and is available again after the wake-up time t_{WU} . During the wake-up phase the **FSR.PBUSY** is set until the wake-up process is completed.

Note: During sleep and wake-up, the Flash is reported to be busy. Thus, read and write accesses to the Flash in Sleep mode are acknowledged with busy’ and should therefore be avoided; those accesses make sense only during wake-up, when waiting for the Flash read mode.

3. The wake-up time t_{WU} is documented in the Data Sheet. This time may fully delay the interrupt response time in Sleep mode.
4. Note: A wake-up is only accepted by the Flash if it is in Sleep mode. The Flash will first complete the ramp down to Sleep mode before reacting to a wake-up trigger.

Idle Read Path

An additional power saving feature is enabled by setting **FCON.IDLE**. In this case the PFLASH read path (**Flash Read Access**) is switched off when no read access is pending. System performance for sequential accesses is slightly reduced because internal linear prefetches of the PFLASH are disabled. Non-sequential read accesses requested by the CPU or any other bus master see no additional delay.

8.6.3 Reset Control

All PMU and Flash functionality is reset with the system reset with the exception of the register bits: **FSR.PROG**, **FSR.ERASE**, **FSR.PFOPER**. These bits are reset with the power-on reset.

The flash will be automatically reset to the read mode after every reset.

8.6.3.1 Resets During Flash Operation

A reset or power failure during an ongoing Flash operation (i.e. program or erase) must be considered as violation of stable operating conditions. However the Flash was designed to prevent damage to non-addressed Flash ranges when the reset is applied as defined in the data sheet. The exceptions are erasing logical sectors and UCBs. Aborting an erase process of a logical sector can leave the complete physical sector unreadable. When an UCB erase is aborted the complete Flash can become unusable. So UCBs must be only erased in a controlled environment. The addressed Flash range is left in an undefined state.

When an erase operation is aborted the addressed logical or physical sector can contain any data. It can even be in a state that doesn't allow this range to be programmed.

When a page programming operation is aborted the page can still appear as erased (but contain slightly programmed bits), it can appear as being correctly programmed (but the data has a lowered retention) or the page contains garbage data. It is also possible that the read data is instable so that depending on the operating conditions different data is read.

For the detection of an aborted Flash process the flags **FSR.PROG** and **FSR.ERASE** could be used as indicator but only when the reset was a System Reset. Power-on resets can not be determined from any flags. It is not possible to detect an aborted operation simply by reading the Flash range. Even the margin reads don't offer a reliable indication.

When erasing or programming the PFLASH usually an external instance can notice the reset and simply restart the operation by erasing the Flash range and programming it again.

However, in cases where this external instance is not existing, a common solution is detecting an abort by performing two operations in sequence and determine after reset from the correctness of the second the completeness of the first operation.

E.g. after erasing a sector a page is programmed. After reset the existence of this page proves that the erase process was performed completely.

The detection of aborted programming processes can be handled similarly. After programming a block of data an additional page is programmed as marker. When after reset the block of data is readable and the marker is existent it is ensured that the block of data was programmed without interruption.

Flash and Program Memory Unit (PMU)

If a complete page can be spent as marker, the following recipe allows to reduce the marker size to 8 bytes. This recipe violates the rule that a page may be programmed only once. This violation is only allowed for this purpose and only when the algorithm is robust against disturbed pages (see also recommendations for handling single-bit errors) by repeating a programming step when it detects a failure.

Robust programming of a page of data with an 8 byte marker:

1. After reset program preferably always first to an even page ("Target Page").
2. If the Other Page on the same wordline contains active data save it to SRAM (the page can become disturbed because of the 4 programming operations per wordline).
3. Program the data to the Target Page.
4. Perform strict check of the Target Page (see below).
5. Program 8 byte marker to Target Page.
6. Perform strict check of the Target Page.
7. In case of any error of the strict check go to the next wordline and program the saved data and the target data again following the same steps.
8. Ensure that the algorithm doesn't repeat unlimited in case of a violation of operating conditions.

Strict checking of programmed data:

1. Ignore single-bit errors and the VER flag.
2. Switch to tight margin 0.
3. If the data (check the complete page) is not equal to the expected data report an error.
4. If a double-bit error is detected report an error.

After reset the algorithm has to check the last programmed page if it was programmed completely:

1. Read with normal read level. Ignore single-bit errors.
2. Read 8-byte marker and check for double-bit error.
3. Read data part and verify its consistency (e.g. by evaluating a CRC). Check for double-bit error.
4. If the data part is defective don't use it (e.g. by invalidating the page).
5. If the data part is ok:
 - a) If the marker is erased the data part could have been programmed incompletely. Therefore the data part should not be used or alternatively it could be programmed again to a following page.
 - b) If the marker contains incorrect data the data part was most likely programmed correctly but the marker was programmed incompletely. The page could be used as is or alternatively the data could be programmed again to a following page.
 - c) If the marker is ok the data part was programmed completely and has the full retention. However this is not ensured for the marker part itself. Therefore the algorithm must be robust against the case that the marker becomes unreadable later.

Flash and Program Memory Unit (PMU)

8.6.4 Clock

The Flash interface is operating at the same clock speed as the CPU, f_{CPU} . Depending on the frequency, wait states must be inserted in the Flash accesses. Further details on the wait states configuration are given in [Section 8.4.4](#).

For proper operation of command sequences and when entering or waking up from Sleep mode, f_{CPU} must be equal or above 1 MHz.

8.7 Registers

The register set consists of the PMU ID register ([Section 8.7.1](#)), the Prefetch Control register ([Section 8.7.2](#)). The other registers control Flash functionality ([Section 8.7.3](#)).

All accesses prevented due to access mode restrictions fail with a bus error.

Also accesses to unoccupied register addresses fail with a bus error.

8.7.1 PMU Registers

The PMU only contains the ID register.

Table 8-6 Registers Address Space

Module	Base Address	End Address	Note
reserved	5800 0000 _H	5800 04FF _H	Bus Error
PMU0	5800 0500 _H	5800 05FF _H	
reserved	5800 0600 _H	5800 0FFF _H	Bus Error
reserved	5800 2400 _H	5800 3FFF _H	Bus Error

Table 8-7 Registers Overview

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Reset Class	Page Number
			Read	Write		
ID	Module Identification	08 _H	U, PV	BE	System Reset	8-34

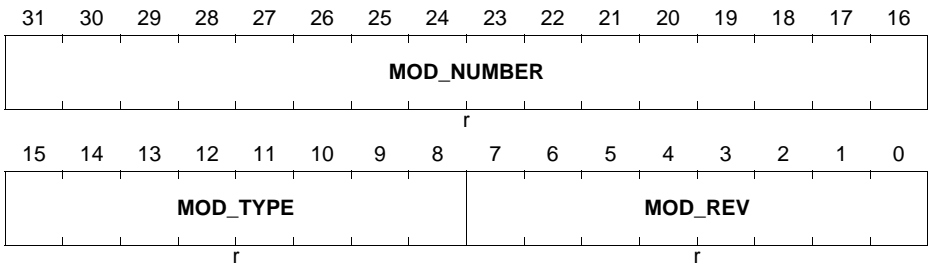
1) The absolute register address is calculated as follows:
Module Base Address ([Table 8-6](#)) + Offset Address (shown in this column)

Flash and Program Memory Unit (PMU)

8.7.1.1 PMU ID Register

The PMU_ID register is a read-only register, thus write accesses lead to a bus error trap. Read accesses are permitted in Privileged Mode PV and in User Mode. The PMU_ID register is defined as follows:

PMU_ID
PMU0 Identification Register (5800 0508_H) **Reset Value: 0091 C0XX_H**



Field	Bits	Type	Description
MOD_REV	[7:0]	r	Module Revision Number MOD_REV defines the module revision number. The value of a module revision starts with 01 _H (first rev.).
MOD_TYPE	[15:8]	r	Module Type This bit field is C0 _H . It defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	Module Number Value This bit field defines the module identification number for PMU0.

8.7.2 Prefetch Registers

This section describes the register of the Prefetch unit.

Table 8-8 Registers Address Space

Module	Base Address	End Address	Note
PREF	5800 4000 _H	5800 7FFF _H	Prefetch Module Registers

Table 8-9 Registers Overview

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Reset Class	Page Number
			Read	Write		
PCON	Prefetch Configuration Register	0 _H	U, PV	U, PV	System Reset	Page 8-35

1) The absolute register address is calculated as follows:
Module Base Address ([Table 8-6](#)) + Offset Address (shown in this column)

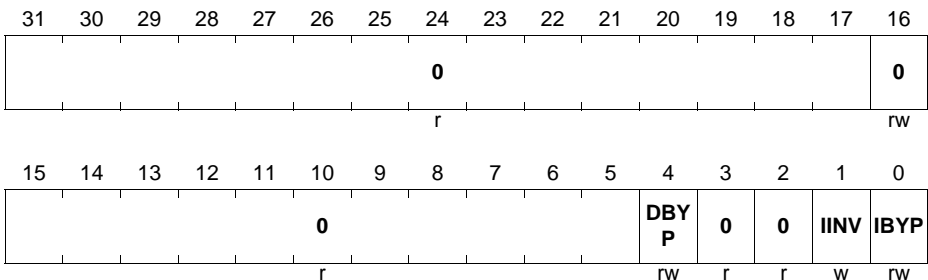
8.7.2.1 Prefetch Configuration Register

This register provides control bits for instruction buffer invalidation and bypass.

PREF_PCON

Prefetch Configuration Register (5800 4000_H)

Reset Value: 0000 0000_H



Flash and Program Memory Unit (PMU)

Field	Bits	Type	Description
IBYP	0	rw	Instruction Prefetch Buffer Bypass 0 _B Instruction prefetch buffer not bypassed. 1 _B Instruction prefetch buffer bypassed.
IINV	1	w	Instruction Prefetch Buffer Invalidate Write Operation: 0 _B No effect. 1 _B Initiate invalidation of entire instruction cache.
DBYP	4	rw	Data Buffer Bypass 0 _B Prefetch Data buffer not bypassed. 1 _B Prefetch Data buffer bypassed.
0	16	rw	Reserved Must be written with 0.
0	[15:5], 3, 2, [31:17]	r	Reserved returns 0 if read; should be written with 0.

Flash and Program Memory Unit (PMU)

8.7.3 Flash Registers

All register addresses are word aligned, independently of the register width. Besides word-read/write accesses, also byte or half-word read/write accesses are supported.

The absolute address of a Flash register is calculated by the base address from [Table 8-10](#) plus the offset address of this register from [Table 8-11](#).

Table 8-10 Registers Address Space

Module	Base Address	End Address	Note
FLASH0	5800 1000 _H	5800 23FF _H	Flash registers of PMU0

The following table shows the addresses, the access modes and reset types for the Flash registers in PMU0:

Table 8-11 Addresses of Flash0 Registers

Short Name	Description	Address	Access Mode		Reset	See
			Read	Write		
–	Reserved	5800 2000 _H – 5800 2004 _H	BE	BE	–	–
FLASH0_ID	Flash Module Identification Register	5800 2008 _H	U, PV	BE	System Reset	Page 8-48
–	Reserved	5800 200C _H	BE	BE	–	–
FLASH0_FSR	Flash Status Register	5800 2010 _H	U, PV	BE	System + PORST	Page 8-38
FLASH0_FCON	Flash Configuration Register	5800 2014 _H	U, PV	PV	System Reset	Page 8-44
FLASH0_MARP	Flash Margin Control Register PFLASH	5800 2018 _H	U, PV	PV	System Reset	Page 8-49
FLASH0_PROCON0	Flash Protection Configuration User 0	5800 2020 _H	U, PV	BE	System Reset	Page 8-50
FLASH0_PROCON1	Flash Protection Configuration User 1	5800 2024 _H	U, PV	BE	System Reset	Page 8-51
FLASH0_PROCON2	Flash Protection Configuration User 2	5800 2028 _H	U, PV	BE	System Reset	Page 8-52
–	Reserved	5800 202C _H – 5800 23FC _H	BE	BE	–	

Flash and Program Memory Unit (PMU)

8.7.3.1 Flash Status Definition

The Flash Status Register FSR reflects the overall status of the Flash module after Reset and after reception of the different commands. Sector specific protection states are not indicated in the FSR, but in the registers PROCON0, PROCON1 and PROCON2. The status register is a read-only register. Only the error flags and the two status flags (PROG, ERASE) are affected with the “Clear Status” command. The error flags are also cleared with the “Reset to Read” command.

The FSR is defined as follows:

FSR
Flash Status Register (1010_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VER	X	0	SLM	0	W PRO DIS1	W PRO DIS0	0	W PRO IN2	W PRO IN1	W PRO IN0	0	R PRO DIS	R PRO IN	0	PRO IN
rh	rh	r	rh	r	rh	rh	r	rh	rh	rh	r	rh	rh	r	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PF DB ER	0	PF SB ER	PRO ER	SQ ER	0	PF OP ER	0	PF PAG E	ERA SE	PRO G	0	0	FA BUS Y	P BUS Y
r	rh	r	rh	rh	rh	r	rh	r	rh	rh	rh	r	r	rh	rh

Field	Bits	Type	Description
PBUSY ¹⁾	0	rh	<p>Program Flash Busy HW-controlled status flag.</p> <p>0_B PFLASH ready, not busy; PFLASH in read mode.</p> <p>1_B PFLASH busy; PFLASH not in read mode.</p> <p>Indication of busy state of PFLASH because of active execution of program or erase operation; PFLASH busy state is also indicated during Flash recovery (after reset) and in power ramp-up state or in sleep mode; while in busy state, the PFLASH is not in read mode.</p>
FABUSY ¹⁾	1	rh	<p>Flash Array Busy Internal busy flag for testing purposes. Must be ignored by application software, which must use PBUSY instead.</p>

Flash and Program Memory Unit (PMU)

Field	Bits	Type	Description
PROG ³⁾⁴⁾	4	rh	<p>Programming State HW-controlled status flag.</p> <p>0_B There is no program operation requested or in progress or just finished.</p> <p>1_B Programming operation (write page) requested (from FIM) or in action or finished.</p> <p>Set with last cycle of Write Page command sequence, cleared with Clear Status command (if not busy) or with power-on reset. If one BUSY flag is coincidentally set, PROG indicates the type of busy state. If xOPER is coincidentally set, PROG indicates the type of erroneous operation. Otherwise, PROG indicates, that operation is still requested or finished.</p>
ERASE ³⁾⁴⁾	5	rh	<p>Erase State HW-controlled status flag.</p> <p>0_B There is no erase operation requested or in progress or just finished</p> <p>1_B Erase operation requested (from FIM) or in action or finished.</p> <p>Set with last cycle of Erase command sequence, cleared with Clear Status command (if not busy) or with power-on reset. Indications are analogous to PROG flag.</p>
PPPAGE ¹⁾²⁾	6	rh	<p>Program Flash in Page Mode HW-controlled status flag.</p> <p>0_B Program Flash not in page mode</p> <p>1_B Program Flash in page mode; assembly buffer of PFLASH (256 byte) is in use (being filled up)</p> <p>Set with Enter Page Mode for PFLASH, cleared with Write Page command</p> <p><i>Note: Concurrent page and read modes are allowed</i></p>
PFOPER ²⁾³⁾⁴⁾	8	rh	<p>Program Flash Operation Error</p> <p>0_B No operation error reported by Program Flash</p> <p>1_B Flash array operation aborted, because of a Flash array failure, e.g. an ECC error in microcode.</p> <p>This bit is not cleared with System Reset, but with power-on reset.</p> <p>Registered status bit; must be cleared per command</p>

Flash and Program Memory Unit (PMU)

Field	Bits	Type	Description
SQER ¹⁾²⁾³⁾	10	rh	<p>Command Sequence Error</p> <p>0_B No sequence error</p> <p>1_B Command state machine operation unsuccessful because of improper address or command sequence.</p> <p>A sequence error is not indicated if the Reset to Read command aborts a command sequence.</p> <p>Registered status bit; must be cleared per command</p>
PROER ¹⁾²⁾³⁾	11	rh	<p>Protection Error</p> <p>0_B No protection error</p> <p>1_B Protection error.</p> <p>A Protection Error is reported e.g. because of a not allowed command, for example an Erase or Write Page command addressing a locked sector, or because of wrong password(s) in a protected command sequence such as “Disable Read Protection”</p> <p>Registered status bit; must be cleared per command</p>
PFSBER ¹⁾²⁾³⁾	12	rh	<p>PFLASH Single-Bit Error and Correction</p> <p>0_B No Single-Bit Error detected during read access to PFLASH</p> <p>1_B Single-Bit Error detected and corrected</p> <p>Registered status bit; must be cleared per command</p>
PFDBER ¹⁾²⁾³⁾	14	rh	<p>PFLASH Double-Bit Error</p> <p>0_B No Double-Bit Error detected during read access to PFLASH</p> <p>1_B Double-Bit Error detected in PFLASH</p> <p>Registered status bit; must be cleared per command</p>
PROIN	16	rh	<p>Protection Installed</p> <p>0_B No protection is installed</p> <p>1_B Read or/and write protection for one or more users is configured and correctly confirmed in the User Configuration Block(s).</p> <p>HW-controlled status flag</p>

Flash and Program Memory Unit (PMU)

Field	Bits	Type	Description
RPROIN	18	rh	<p>Read Protection Installed</p> <p>0_B No read protection installed</p> <p>1_B Read protection and global write protection is configured and correctly confirmed in the User Configuration Block 0.</p> <p>Supported only for the master user (user zero). HW-controlled status flag</p>
RPRODIS¹⁾⁵⁾	19	rh	<p>Read Protection Disable State</p> <p>0_B Read protection (if installed) is not disabled</p> <p>1_B Read and global write protection is temporarily disabled.</p> <p>Flash read with instructions from other memory, as well as program or erase on not separately write protected sectors is possible. HW-controlled status flag</p>
WPROIN0	21	rh	<p>Sector Write Protection Installed for User 0</p> <p>0_B No write protection installed for user 0</p> <p>1_B Sector write protection for user 0 is configured and correctly confirmed in the User Configuration Block 0.</p> <p>HW-controlled status flag</p>
WPROIN1	22	rh	<p>Sector Write Protection Installed for User 1</p> <p>0_B No write protection installed for user 1</p> <p>1_B Sector write protection for user 1 is configured and correctly confirmed in the User Configuration Block 1.</p> <p>HW-controlled status flag</p>
WPROIN2	23	rh	<p>Sector OTP Protection Installed for User 2</p> <p>0_B No OTP write protection installed for user 2</p> <p>1_B Sector OTP write protection with ROM functionality is configured and correctly confirmed in the UCB2. The protection is locked for ever.</p> <p>HW-controlled status flag</p>

Flash and Program Memory Unit (PMU)

Field	Bits	Type	Description
WPRODISO ¹⁾⁵⁾	25	rh	<p>Sector Write Protection Disabled for User 0</p> <p>0_B All protected sectors of user 0 are locked if write protection is installed</p> <p>1_B All write-protected sectors of user 0 are temporarily unlocked, if not coincidentally locked by user 2 or via read protection.</p> <p>Hierarchical protection control: User-0 sectors are also unlocked, if coincidentally protected by user 1. But not vice versa.</p> <p>HW-controlled status flag</p>
WPRODISE ¹⁾⁵⁾	26	rh	<p>Sector Write Protection Disabled for User 1</p> <p>0_B All protected sectors of user 1 are locked if write protection is installed</p> <p>1_B All write-protected sectors of user 1 are temporarily unlocked, if not coincidentally locked by user 0 or user 2 or via read protection.</p> <p>HW-controlled status flag</p>
SLM ¹⁾	28	rh	<p>Flash Sleep Mode</p> <p>HW-controlled status flag. Indication of Flash sleep mode taken because of global or individual sleep request; additionally indicates when the Flash is in shut down mode.</p> <p>0_B Flash not in sleep mode</p> <p>1_B Flash is in sleep or shut down mode</p>
X	30	rh	<p>Reserved</p> <p>Value undefined</p>
VER ¹⁾³⁾	31	rh	<p>Verify Error</p> <p>0_B The page is correctly programmed or the sector correctly erased. All programmed or erased bits have full expected quality.</p> <p>1_B A program verify error or an erase verify error has been detected. Full quality (retention time) of all programmed ("1") or erased ("0") bits cannot be guaranteed.</p> <p>See Section 8.5.3 and Section 8.5.3.6 for proper reaction.</p> <p>Registered status bit; must be cleared per command</p>

Flash and Program Memory Unit (PMU)

Field	Bits	Type	Description
0	2,3,7, 9,13, 15,17, 20,24, 27, 29	r	Reserved Read zero, no write

Note: The footnote numbers of FSR bits describe the specific reset conditions:

- 1)Cleared with System Reset
- 2)Cleared with command "Reset to Read"
- 3)Cleared with command "Clear Status"
- 4)Cleared with power-on reset (PORST)
- 5)Cleared with command "Resume Protection"

Note: The xBUSY flags as well as the protection flags cannot be cleared with the "Clear Status" command or with the "Reset to Read" command. These flags are controlled by HW.

Note: The reset value above is indicated after correct execution of Flash ramp up. Additionally, errors are possible after ramp up (see [Section 8.5.3.6](#)).

8.7.3.2 Flash Configuration Control

The Flash Configuration Register FCON reflects and controls the following general Flash configuration functions:

- Number of wait states for Flash accesses.
- Indication of installed and active read protection.
- Instruction and data access control for read protection.
- Interrupt mask bits.
- Power reduction and shut down control.

FCON is a Privileged Mode protected register. It is defined as follows:

Flash and Program Memory Unit (PMU)

FCON

Flash Configuration Register

(1014_H)

Reset value: 000X 0006_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EOB M	0	PF DB ERM	0	PF SB ERM	PRO ERM	SQ ERM	VOP ERM	0	0	0	0	0	DDF	DCF	RPA
rw	r	rw	r	rw	rw	rw	rw	r	r	r	r	r	rwh	rwh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SL EEP	ESL DIS	IDLE					0					WS EC PF	WSPFLASH		
rw	rw	rw	r			r			r			rw	rw		

Field	Bits	Type	Description
WSPFLASH	[3:0]	rw	<p>Wait States for read access to PFLASH</p> <p>This bit field defines the number of wait states n, which are used for an initial read access to the Program Flash memory area, with $WSPFLASH \times (1 / f_{CPU}) \geq t_a^{(1)}$.</p> <p>0000_B PFLASH access in one clock cycle 0001_B PFLASH access in one clock cycle 0010_B PFLASH access in two clock cycles 0011_B PFLASH access in three clock cycles PFLASH access in four up to fourteen clock cycles. 1111_B PFLASH access in fifteen clock cycles.</p>
WSECPF	4	rw	<p>Wait State for Error Correction of PFLASH</p> <p>0_B No additional wait state for error correction 1_B One additional wait state for error correction during read access to Program Flash. If enabled, this wait state is only used for the first transfer of a burst transfer. Set this bit only when requested by Infineon.</p>
IDLE	13	rw	<p>Dynamic Flash Idle</p> <p>0_B Normal/standard Flash read operation 1_B Dynamic idle of Program Flash enabled for power saving; static prefetching disabled</p>

Flash and Program Memory Unit (PMU)

Field	Bits	Type	Description
ESLDIS	14	rw	<p>External Sleep Request Disable</p> <p>0_B External sleep request signal input is enabled</p> <p>1_B Externally requested Flash sleep is disabled</p> <p>The 'external' signal input is connected with a global power-down/sleep request signal from SCU.</p>
SLEEP	15	rw	<p>Flash SLEEP</p> <p>0_B Normal state or wake-up</p> <p>1_B Flash sleep mode is requested</p> <p>Wake-up from sleep is started with clearing of the SLEEP-bit.</p>
RPA	16	rh	<p>Read Protection Activated</p> <p>This bit monitors the status of the Flash-internal read protection. This bit can only be '0' when read protection is not installed or while the read protection is temporarily disabled with password sequence.</p> <p>0_B The Flash-internal read protection is not activated. Bits DCF, DDF are not taken into account. Bits DCF, DDFx can be cleared</p> <p>1_B The Flash-internal read protection is activated. Bits DCF, DDF are enabled and evaluated.</p>
DCF	17	rwh	<p>Disable Code Fetch from Flash Memory</p> <p>This bit enables/disables the code fetch from the internal Flash memory area. Once set, this bit can only be cleared when RPA='0'.</p> <p>This bit is automatically set with reset and is cleared during ramp up, if no RP installed, and during startup (BROM) in case of internal start out of Flash.</p> <p>0_B Code fetching from the Flash memory area is allowed.</p> <p>1_B Code fetching from the Flash memory area is not allowed. This bit is not taken into account while RPA='0'.</p>

Flash and Program Memory Unit (PMU)

Field	Bits	Type	Description
DDF	18	rwh	<p>Disable Any Data Fetch from Flash</p> <p>This bit enables/disables the data read access to the Flash memory area (Program Flash and Data Flash). Once set, this bit can only be cleared when RPA='0'. This bit is automatically set with reset and is cleared during ramp up, if no RP installed, and during startup (BROM) in case of internal start out of Flash.</p> <p>0_B Data read access to the Flash memory area is allowed.</p> <p>1_B Data read access to the Flash memory area is not allowed. This bit is not taken into account while RPA='0'.</p>
VOPERM	24	rw	<p>Verify and Operation Error Interrupt Mask</p> <p>0_B Interrupt not enabled</p> <p>1_B Flash interrupt because of Verify Error or Operation Error in Flash array (FSI) is enabled</p>
SQERM	25	rw	<p>Command Sequence Error Interrupt Mask</p> <p>0_B Interrupt not enabled</p> <p>1_B Flash interrupt because of Sequence Error is enabled</p>
PROERM	26	rw	<p>Protection Error Interrupt Mask</p> <p>0_B Interrupt not enabled</p> <p>1_B Flash interrupt because of Protection Error is enabled</p>
PFSBERM	27	rw	<p>PFLASH Single-Bit Error Interrupt Mask</p> <p>0_B No Single-Bit Error interrupt enabled</p> <p>1_B Single-Bit Error interrupt enabled for PFLASH</p>
PFDBERM	29	rw	<p>PFLASH Double-Bit Error Interrupt Mask</p> <p>0_B Double-Bit Error interrupt for PFLASH not enabled</p> <p>1_B Double-Bit Error interrupt for PFLASH enabled. Especially intended for margin check</p>
EOBM	31	rw	<p>End of Busy Interrupt Mask</p> <p>0_B Interrupt not enabled</p> <p>1_B EOB interrupt is enabled</p>
0	[12:5], [23:19], 28, 30	r	<p>Reserved</p> <p>Always read/write zero</p>

Flash and Program Memory Unit (PMU)

1) WSPFLASH = 0_H deviates from this formula and results in the same timing as WSPFLASH = 1_H.

Note: The default numbers of wait states represent the slow cases. This is a general proceeding and additionally opens the possibility to execute higher frequencies without changing the configuration.

Note: After reset and execution of Firmware, the read protection control bits are coded as follows:

DDF, DCF, RPA = "110": No read protection installed

DDF, DCF, RPA = "001": Read protection installed; start in internal Flash

DDF, DCF, RPA = "111": Read protection installed; start not in internal Flash.

Flash and Program Memory Unit (PMU)

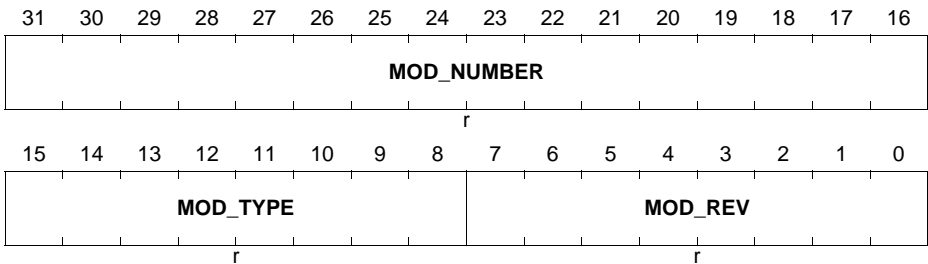
8.7.3.3 Flash Identification Register

The module identification register of Flash module is directly accessible by the CPU via PMU access. This register is mapped into the space of the Flash Interface Module's registers (see [Table 8-11](#)).

FLASH0_ID

Flash Module Identification Register (1008_H)

Reset Value: 0092 C0XX_H



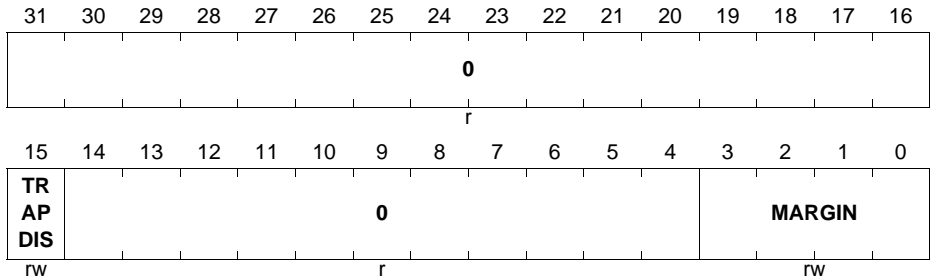
Field	Bits	Type	Description
MOD_REV	[7:0]	r	Module Revision Number MOD_REV defines the module revision number. The value of a module revision starts with 01 _H (first revision).
MOD_TYPE	[15:8]	r	Module Type This bit field is C0 _H . It defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	Module Number Value This bit field defines a module identification number. For the XMC4300 Flash0 this number is 0092 _H .

Flash and Program Memory Unit (PMU)

8.7.3.4 Margin Check Control Register

MARP

Margin Control Register PFLASH (1018_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
MARGIN	[3:0]	rw	PFLASH Margin Selection 0000 _B Default , Standard (default) margin. 0001 _B Tight0 , Tight margin for 0 (low) level. Suboptimal 0-bits are read as 1s. 0100 _B Tight1 , Tight margin for 1 (high) level. Suboptimal 1-bits are read as 0s. – Reserved.
TRAPDIS	15	rw	PFLASH Double-Bit Error Trap Disable 0 _B If a double-bit error occurs in PFLASH, a bus error trap is generated ¹⁾ . 1 _B The double-bit error trap is disabled. Shall be used only during margin check
0	[14:4], [31:16]	r	Reserved Always read as 0; should be written with 0.

1) After Boot ROM exit, double-bit error traps are enabled (TRAPDIS = 0).

8.7.3.5 Protection Configuration Indication

The configuration of read/write/OTP protection is indicated with registers PROCON0, PROCON1 and PROCON2, thus separately for every user, and it is generally indicated in the status register FSR.

If write protection is installed for user 0 or 1 or OTP protection for user 2, for each sector of the Program Flash it is indicated in the user-specific Protection Configuration register PROCONx, if it is locked or unlocked for program or erase operations.

Flash and Program Memory Unit (PMU)

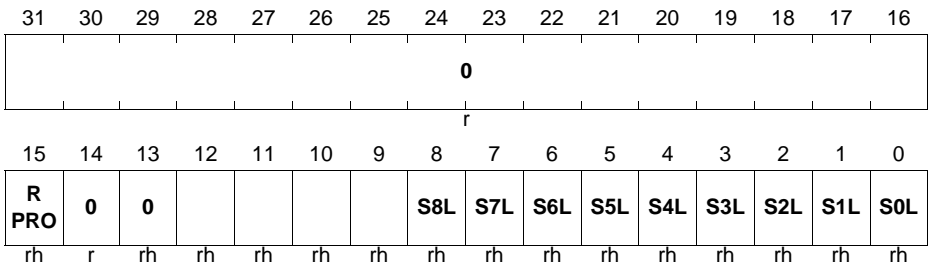
The Flash Protection Configuration registers PROCONx are loaded out of the user's configuration block directly after reset during ramp up. For software the three PROCONx registers are read-only registers.

PROCON0

Flash Protection Configuration Register User 0

(1020_H)

Reset Value: 0000 XXXX_H



Field	Bits	Type	Description
SnL (n=0-)	n	rh	Sector n Locked for Write Protection by User 0 These bits indicate whether PFLASH sector n is write-protected by user 0 or not. 0 _B No write protection is configured for sector n. 1 _B Write protection is configured for sector n.
RPRO	15	rh	Read Protection Configuration This bit indicates whether read protection is configured for PFLASH by user 0. 0 _B No read protection configured 1 _B Read protection and global write protection is configured by user 0 (master user)
0	13	rh	Reserved deliver the corresponding UCB0 entry. Shall be configured to 0.
0	[31:16], 14	r	Reserved Always reads as 0.

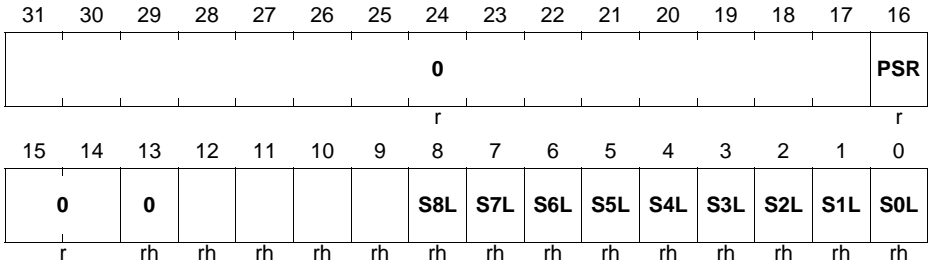
Flash and Program Memory Unit (PMU)

PROCON1

Flash Protection Configuration Register User 1

(1024_H)

Reset Value: 0000 XXXX_H



Field	Bits	Type	Description
SnL (n=0-)	n	rh	<p>Sector n Locked for Write Protection by User 1</p> <p>These bits indicate whether PFLASH sector n is write-protected by user 1 or not.</p> <p>0_B No write protection is configured for sector n.</p> <p>1_B Write protection is configured for sector n.</p>
PSR	16	r	<p>Physical Sector Repair</p> <p>Enables the Physical Sector Repair command sequence, to repair physical sector PS4 after a logical sector erase has been aborted by a reset. If enabled, it replaces the Erase Physical Sector command sequence.</p> <p>0_B Physical Sector Repair command disabled; Erase Physical Sector command sequence available.</p> <p>1_B Physical Sector Repair command sequence enabled; Erase Physical Sector command sequence disabled.</p> <p><i>Note: This bit is never set in the PROCON1 register and will always read as 0_B, but its configuration is evaluated according to the UCB1 content.</i></p>
0	13	rh	<p>Reserved</p> <p>Deliver the corresponding UCB1 entry. Shall be configured to 0.</p>

Flash and Program Memory Unit (PMU)

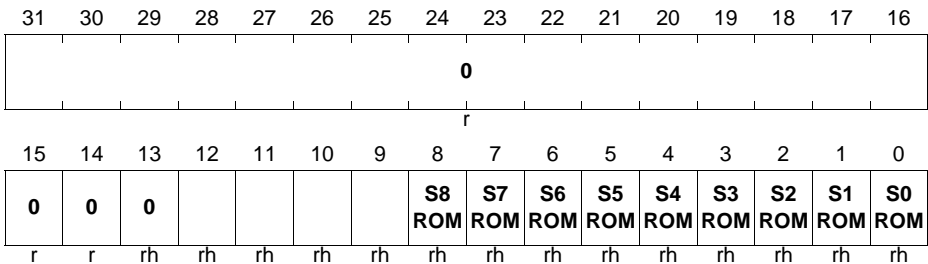
Field	Bits	Type	Description
0	[31:16], 15, 14	r	Reserved Always reads as 0.

PROCON2

Flash Protection Configuration Register User 2

(1028_H)

Reset Value: 0000 XXXX_H



Field	Bits	Type	Description
SnROM (n=0-)	n	rh	Sector n Locked Forever by User 2 These bits indicate whether PFLASH sector n is an OTP protected sector with read-only functionality, thus if it is locked for ever. 0 _B No ROM functionality configured for sector n. 1 _B ROM functionality is configured for sector n. Re-programming of this sector is no longer possible.
0	13	r	Reserved Deliver the corresponding UCB2 entry. Shall be configured to 0.
0	[31:16], 15, 14	r	Reserved Always reads as 0.

System Control

9 Window Watchdog Timer (WDT)

Purpose of the Window Watchdog Timer module is improvement of system integrity. WDT triggers the system reset or other corrective action like e.g. non-maskable interrupt if the main program, due to some fault condition, neglects to regularly service the watchdog (also referred to as “kicking the dog”, “petting the dog”, “feeding the watchdog” or “waking the watchdog”). The intention is to bring the system back from unresponsive state into normal operation.

References

[9] Cortex-M4 User Guide, ARM DUI 0508B (ID062910)

9.1 Overview

A successful servicing of the WDT results in a pulse on the signal `wdt_service`. The signal is offered also as an alternate function output. It can be used to show to an external watchdog that the system is alive.

The WDT timer is a 32-bit counter, which counts up from 0_H . It can be serviced while the counter value is within the window boundary, i.e. between the lower and the upper boundary value. Correct servicing results in a reset of the counter to 0_H . A so called “Bad Service” attempt results in a system reset request.

The timer block is running on the f_{WDT} clock which is independent from the bus clock. The timer value is updated in the corresponding register **TIM**, whenever the timer value increments. This mechanism enables immediate response on a read access from the bus.

The WDT module provides register interface for configuration. A write to writable registers is only allowed, when the access is in privileged mode. A write access in user mode results in a bus error response.

9.1.1 Features

The watchdog timer (WDT) is an independent window watchdog timer.

The features are:

- Triggers system reset when not serviced on time or serviced in a wrong way
- Servicing restricted to be within boundaries of a user definable refresh window
- Can run from an independent clock
- Provides service indication to an external pin
- Can be suspended in HALT mode
- Provides optional pre-warning alarm before reset

Table 9-1 Application Features

Feature	Purpose/Application
System reset upon Bad Servicing	Triggered to restore system stable operation and ensure system integrity
Servicing restricted to be within defined boundaries of refresh window	Allows to consider minimum and maximum software timing
Independent clocks	To ensure that WDT counts even in case of the system clock failure
Service indication on external pin	For dual-channel watchdog solution, additional external control of system integrity
Suspending in HALT mode	Enables safe debugging with productive code
Pre-warning alarm	Software recovery to allow corrective action via software recovery routine bringing system back from the unresponsive state into normal operation

9.1.2 Block Diagram

The WDT block diagram is shown in [Figure 9-1](#).

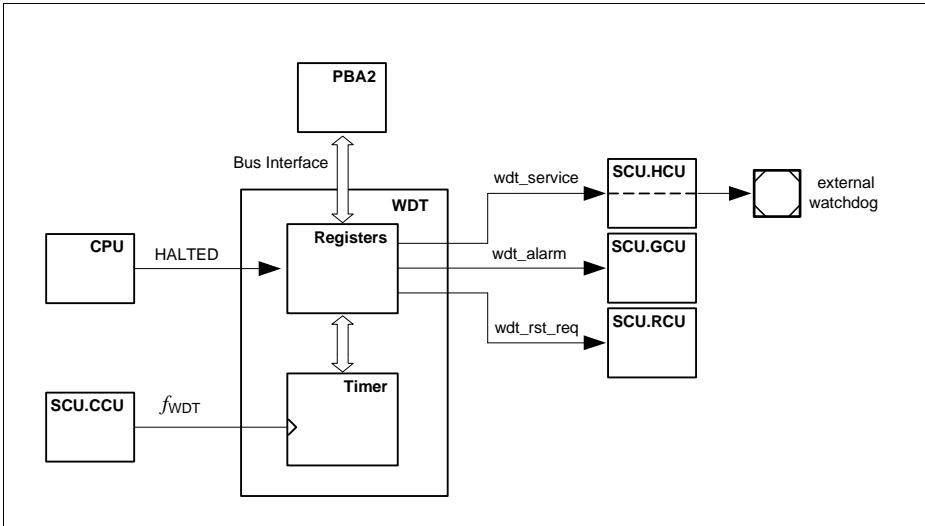


Figure 9-1 Watchdog Timer Block Diagram

9.2 Time-Out Mode

An overflow results in an immediate reset request going to the RCU of the SCU via the signal **wdt_rst_req** whenever the counter crosses the upper boundary it triggers an overflow event pre-warning is not enabled with **CTR** register. A successful servicing performed with writing a unique value, referred to as “Magic Word” to the **SRV** register of the WDT within the valid servicing window, results in a pulse on the signal **wdt_service** and reset of the timer counter.

Window Watchdog Timer (WDT)

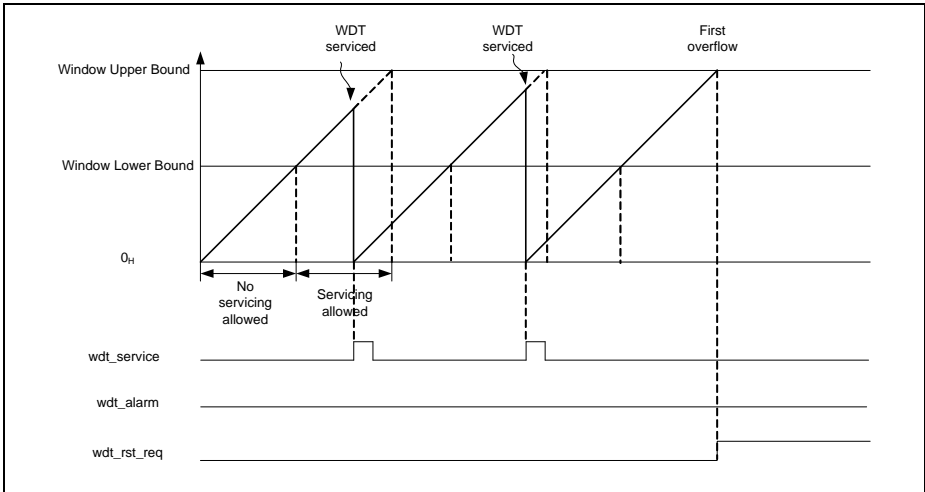


Figure 9-2 Reset without pre-warning

The example scenario depicted in [Figure 9-2](#) shows two consecutive service pulses generated from WDT module as the result of successful servicing within valid time windows. The situation where no service has been performed immediately triggers generation of reset request on the wdt_rst_req output after the counter value has exceeded window upper bound value.

9.3 Pre-warning Mode

While in pre-warning mode the effect of the overflow event is different with and without pre-warning enabled. The first crossing of the upper bound triggers the outgoing alarm signal wdt_alarm when pre-warning is enabled. Only the next overflow results a reset request. The alarm status is shown via register [WDTSTS](#) and can be cleared via register [WDTCLR](#). A clear of the alarm status will bring the WDT back to normal state. The alarm signal is routed as request to the SCU, where it can be promoted to NMI.

Window Watchdog Timer (WDT)

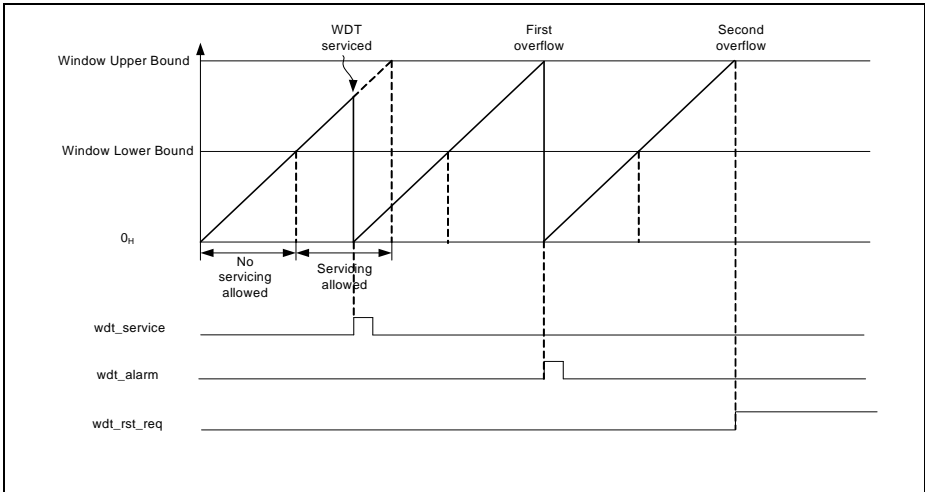


Figure 9-3 Reset after pre-warning

The example scenario depicted in [Figure 9-3](#) shows service pulse generated from WDT module as the result of successful servicing within valid time window. WDT generates alarm pulse on `wdt_alarm` upon first missing servicing. The alarm signal is routed as interrupt request to the SCU, where it can be promoted to NMI. Within this alarm service request the user can clear the WDT status bit and give a proper WDT service before it overflows next time. Otherwise WDT generates reset request on `wdt_rst_req` upon the second missing service.

9.4 Bad Service Operation

A bad service attempt results in a reset request. A bad service attempt can be due to servicing outside the window boundaries or servicing with an invalid Magic Word.

Window Watchdog Timer (WDT)

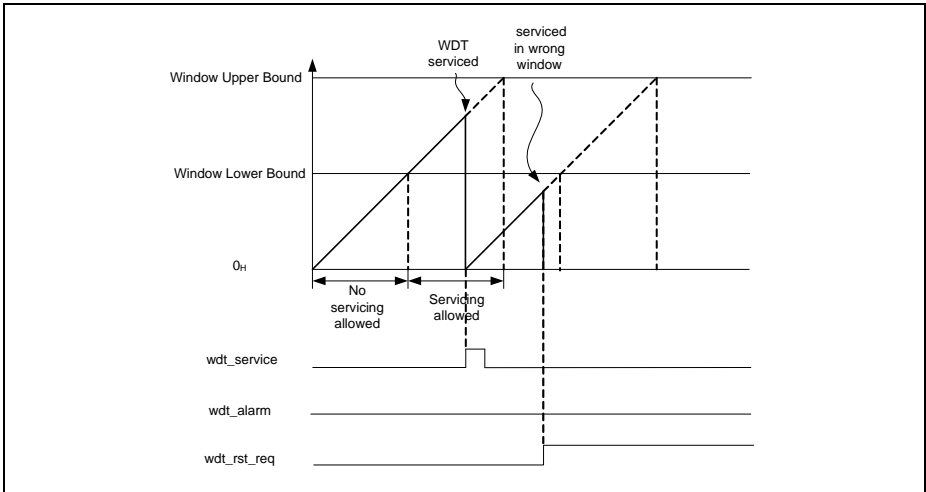


Figure 9-4 Reset upon servicing in a wrong window

The example in **Figure 9-4** shows servicing performed outside of valid servicing window. Attempt to service WDT while counter value remains below the Window Lower Bound results in immediate reset request on `wdt_rst_req` signal.

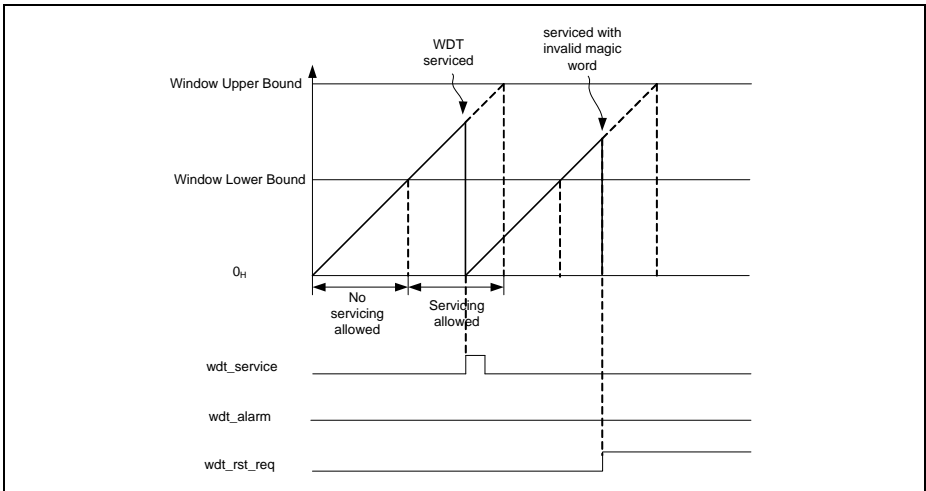


Figure 9-5 Reset upon servicing with a wrong magic word

Window Watchdog Timer (WDT)

The example in **Figure 9-5** shows servicing performed within a valid servicing window but with an invalid Magic Word. Attempt to write a wrong word to the **SRV** register results in immediate reset request on `wdt_rst_req` signal.

9.5 Service Request Processing

The WDT generates watchdog alarm service requests via `wdt_alarm` output signal upon first counter overflow over Watchdog Upper Bound when pre-warning mode is enabled. The alarm service request may be promoted by the SCU in two alternative modes:

- service request
- trap request causing NMI interrupt

Service requests can be disabled i SCU with service request mask or trap request disable registers respectively.

9.6 Debug Behavior

The WDT function can be suspended when the CPU enters HALT mode. WDT debug function is controlled by DSP bit field in **CTR** register.

9.7 Power, Reset and Clock

The WDT module is a part of the core domain and supplied with VDDC voltage.

All WDT registers get reset with the system reset.

A sticky bit in the RSSTAT register of SCU/RCU module indicates whether the last system reset has been triggered by the WDT module. This bit does not get reset with system reset.

The input clock of the WDT counter can be selected by the user between system PLL output, direct output of the internal system oscillator or 32kHz clock of hibernate domain, independently from the AHB interface clock. Selection of the WDT input clock is performed in SCU using WDTCLKCR register (for details please refer to the SCU/CCU chapter).

9.8 Initialization and Control Sequence

The programming model of the WDT module assumes several scenarios where different control sequences apply.

Note: Some of the scenarios described in this chapter require operations on system level the that are not in the scope of the WDT module description, therefore for detailed information please refer to relevant chapters of this document.

9.8.1 Initialization & Start of Operation

Complete WDT module initialization is required after system reset.

- check reason for last system reset in order to determine power state
 - read out SCU_RSTSTAT.RSTSTAT register bit field to determine last system reset cause
 - perform appropriate operations dependent on the last system reset cause
- WDT software initialization sequence
 - enable WDT clock with SCU_CLKSET.WDTCEN register bit field
 - release WDT reset with SCU_PRCLR2.WDTRS register bit field
 - set lower window bound with WDT_WLB register
 - set upper window bound with WDT_WUB register
 - configure external watchdog service indication (optional, please refer to SCU/HCU chapter)
 - select and enable WDT input clock with SCU_WDTCLKCR register
 - enable system trap for pre-warning alarm on system level with SCU_NMIREQEN register (optional, used in WDT pre-warning mode only)
- software start sequence
 - select mode (Time-Out or Pre-warning) and enable WDT module with WDT_CTR register
- service the watchdog
 - check current timer value in WDT_TIM register against programmed time window
 - write magic word to WDT_SRV register within valid time window

9.8.2 Reconfiguration & Restart of Operation

Reset and initialization of the WDT module is required in order to update its settings.

- software initialization sequence
 - assert WDT reset with SCU_PRSET2.WDTCEN register bit field
 - release WDT reset with SCU_PRCLR2.WDTRS register bit field register
 - set lower window bound with WDT_WLB register
 - set upper window bound with WDT_WUB register
 - configure external watchdog service indication (optional, please refer to SCU/HCU chapter)
 - select and enable WDT input clock (if change of the clock settings required) with SCU_WDTCLKCR register
 - enable system trap for pre-warning alarm on system level with SCU_NMIREQEN register (optional, used in WDT pre-warning mode only)
- software start sequence
 - select mode (Time-Out or Pre-warning) and enable WDT module with WDT_CTR register

Window Watchdog Timer (WDT)

- service the watchdog
 - check current timer value in WDT_TIM register against programmed time window
 - write magic word to WDT_SRV register within valid time window

9.8.3 Software Stop & Resume Operation

The WDT module can be stopped and re-started at any point of time for e.g. debug purpose using software sequence.

- software stop sequence
 - disable WDT module with WDT_CTR register
- perform any user operations
- software start (resume) sequence
 - enable WDT module with WDT_CTR register with WDT_CTR register
- service the watchdog
 - check current timer value in WDT_TIM register against programmed time window
 - write magic word to WDT_SRV register within valid time window

9.8.4 Enter Sleep/Deep Sleep & Resume Operation

The WDT counter clock can be configured to stop while in sleep or deep-sleep mode. No direct software interaction with the WDT is required in those modes and no watchdog time-out will fire if the WDT clock is configured to stop while CPU is sleeping.

- software configuration sequence for sleep/deep-sleep mode
 - configure WDT behavior with SCU register SLEEPCLR or DSLEEPCLR
- enter sleep/deep-sleep mode software sequence
 - select sleep or deep-sleep mode in CPU (for details please refer to Cortex-M4 documentation [9])
 - enter selected mode (for details please refer to Cortex-M4 documentation [9])
- wait for a wake-up event (no software interaction, CPU stopped)
- resume operation (CPU clock restarted automatically on an event)
- service the watchdog
 - check current timer value in WDT_TIM register against programmed time window
 - write magic word to WDT_SRV register within valid time window

9.8.5 Prewarning Alarm Handling

The WDT will fire prewarning alarm before requesting system reset while in pre-warning mode and not serviced within valid time window. The WDT status register indicating alarm must be cleared before the timer counter value crosses the upper bound for the second time after firing the alarm. After clearing of the alarm status regular watchdog servicing must be performed within valid time window.

Window Watchdog Timer (WDT)

- alarm event
 - exception routine (system trap or service request) clearing WDT_WDTSTAT register with WDT_WDTCLR register
- service the watchdog
 - check current timer value in WDT_TIM register against programmed time window
 - write magic word to WDT_SRV register within valid time window

Window Watchdog Timer (WDT)

9.9 Registers

Registers Overview

All these registers can be read in User Mode, but can only be written in Supervisor Mode.

The absolute register address is calculated by adding:

Module Base Address + Offset Address

Table 9-2 Registers Address Space

Module	Base Address	End Address	Note
WDT	5000 8000 _H	5000 BFFF _H	Watchdog Timer Registers

Table 9-3 Register Overview

Short Name	Register Long Name	Offset Addr.	Access Mode		Description
			Read	Write	
WDT Kernel Registers					
ID	Module ID Register	00 _H	U, PV	PV	Page 9-11
CTR	Control Register	04 _H	U, PV	PV	Page 9-12
SRV	Service Register	08 _H	BE	PV	Page 9-13
TIM	Timer Register	0C _H	U, PV	BE	Page 9-14
WLB	Window Lower Bound	10 _H	U, PV	PV	Page 9-14
WUB	Window Upper Bound	14 _H	U, PV	PV	Page 9-14
WDTSTS	Watchdog Status Register	18 _H	U, PV	PV	Page 9-15
WDTCLR	Watchdog Status Clear Register	1C _H	U, PV	PV	Page 9-16

9.9.1 Registers Description

ID

The module ID register.

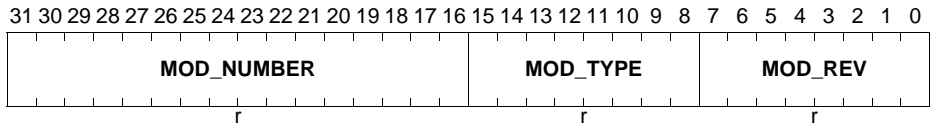
Window Watchdog Timer (WDT)

ID

WDT ID Register

(00_H)

Reset Value: 00AD C0XX_H



Field	Bits	Type	Description
MOD_REV	[7:0]	r	Module Revision Indicates the revision number of the implementation. This information depends on the design step.
MOD_TYPE	[15:8]	r	Module Type This internal marker is fixed to C0 _H .
MOD_NUMBER	[31:16]	r	Module Number Indicates the module identification number

CTR

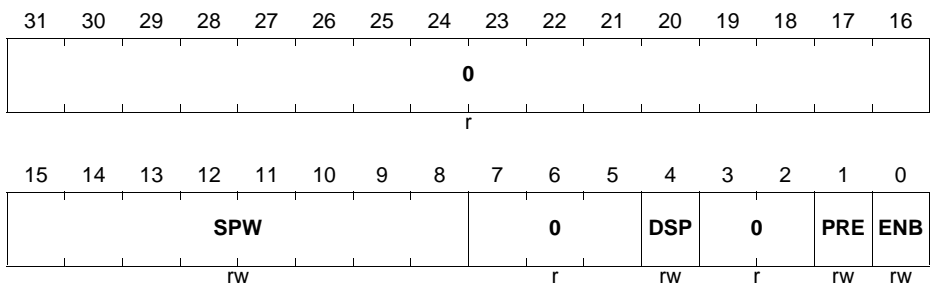
The operation mode control register.

CTR

WDT Control Register

(04_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ENB	0	rw	Enable 0 _B disables watchdog timer, 1 _B enables watchdog timer

Window Watchdog Timer (WDT)

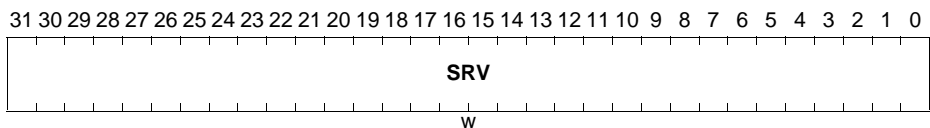
Field	Bits	Type	Description
PRE	1	rw	Pre-warning 0 _B disables pre-warning 1 _B enables pre-warning,
DSP	4	rw	Debug Suspend 0 _B watchdog timer is stopped during halting mode debug, 1 _B watchdog timer is not stopped during halting mode debug
SPW	[15:8]	rw	Service Indication Pulse Width Pulse width (SPW+1) of service indication in f _{WDT} cycles
0	[3:2], [7:5], [31:16]	r	Reserved

SRV

The WDT service register. Software must write a magic word while the timer value is within the valid window boundary. Writing the magic word while the timer value is within the window boundary will service the watchdog and result a reload of the timer with 0H. Upon writing data different than the magic word within valid time window or writing even correct Magic Word but outside of the valid time window no servicing will be performed. Instead will request an immediate system reset request.

SRV

WDT Service Register (08_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
SRV	[31:0]	w	Service Writing the magic word ABADCAFE _H while the timer value is within the window boundary will service the watchdog.

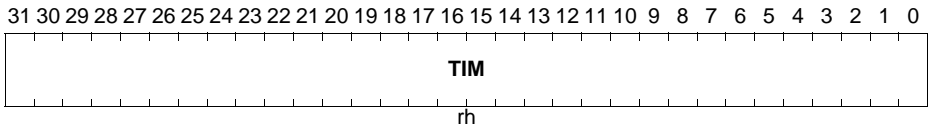
Window Watchdog Timer (WDT)

TIM

The actual watchdog timer register count value. This register can be read by software in order to determine current position in the WDT time window.

TIM

WDT Timer Register (0C_H) **Reset Value: 0000 0000_H**



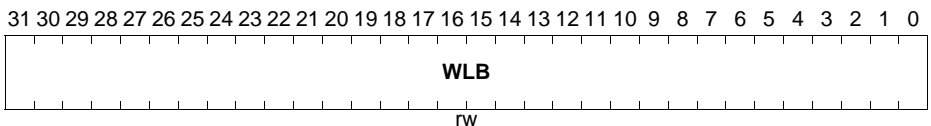
Field	Bits	Type	Description
TIM	[31:0]	rh	Timer Value Actual value of watchdog timer value.

WLB

The Window Lower Bound register defines the lower bound for servicing window. Servicing of the watchdog has only effect within the window boundary

WLB

WDT Window Lower Bound Register (10_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
WLB	[31:0]	rw	Window Lower Bound Lower bound for servicing window. Setting the lower bound to 0 _H disables the window mechanism.

WUB

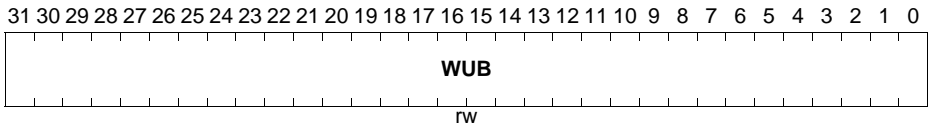
The Window Upper Bound register defines the upper bound for servicing window. Servicing of the watchdog has only effect within the window boundary.

Window Watchdog Timer (WDT)

WUB

WDT Window Upper Bound Register (14_H)

Reset Value: FFFF FFFF_H



Field	Bits	Type	Description
WUB	[31:0]	rw	<p>Window Upper Bound Upper Bound for servicing window. The WDT triggers an reset request when the timer is crossing the upper bound value without pre-warning enabled. With pre-warning enabled the first crossing triggers a watchdog alarm and the second crossing triggers a system reset.</p>

WDTSTS

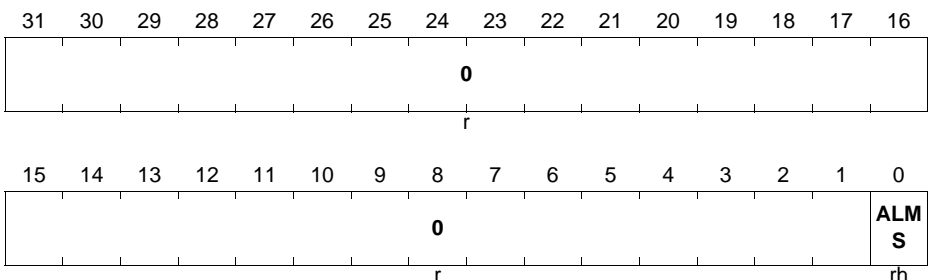
The status register contains sticky bit indicating occurrence of alarm condition.

WDTSTS

WDT Status Register

(0018_H)

Reset Value: 00000000_H



Field	Bits	Type	Description
ALMS	0	rh	<p>Pre-warning Alarm 1_B pre-warning alarm occurred, 0_B no pre-warning alarm occurred</p>

Window Watchdog Timer (WDT)

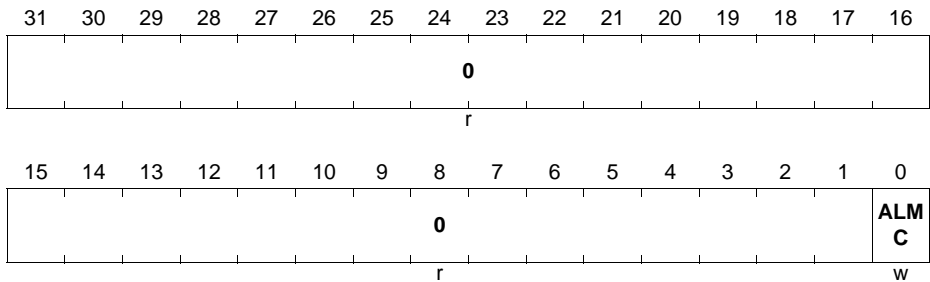
Field	Bits	Type	Description
0	[31:1]	r	Reserved

WDTCLR

The status register contains sticky bitfield indicating occurrence of alarm condition.

WDTCLR

WDT Clear Register (001C_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
ALMC	0	w	Pre-warning Alarm 1 _B clears pre-warning alarm 0 _B no-action
0	[31:1]	r	Reserved

9.10 Interconnects

Table 9-4 Pin Table

Input/Output	I/O	Connected To	Description
Clock and Reset Signals			
f_{WDT}	I	SCU.CCU	timer clock
Timer Signals			
wdt_service	O	SCU.HCU	service indication to external watchdog

Window Watchdog Timer (WDT)

Table 9-4 Pin Table (cont'd)

Input/Output	I/O	Connected To	Description
HALTED	I	CPU	In halting mode debug. HALTED remains asserted while the core is in debug.
Service Request Connectivity			
wdt_alarm	O	SCU.GCU	pre-warning alarm
wdt_rst_req	O	SCU.RCU	reset request



10 Real Time Clock (RTC)

Real-time clock (RTC) is a clock that keeps track of the current time. RTCs are present in almost any electronic device which needs to keep accurate time in a digital format for clock displays and real-time actions.

10.1 Overview

The RTC module tracks time with separate registers for hours, minutes, and seconds. The calendar registers track date, day of the week, month and year with automatic leap year correction.

The RTC is capable of running from an alternate source of power, so it can continue to keep time while the primary source of power is off or unavailable. The timer remains operational when the core domain is in power-down. The kernel part of the RTC keeps running as long as the hibernate domain is powered with an alternate supply source. The alternate source can be for example a lithium battery or a supercapacitor.

10.1.1 Features

The features of the Real Time Clock (RTC) module are:

- Precise real time keeping with
 - 32.768 kHz external crystal clock
 - 32.768 kHz high precision internal clock
- Periodic time-based interrupt
- Programmable alarm interrupt on time match
- Supports wake-up mechanism from hibernate state

Table 10-1 Application Features

Feature	Purpose/Application
Precise real-time keeping	Reduced need for time adjustments
Periodic time-based interrupt	Scheduling of operations performed on precisely defined intervals
Programmable alarm interrupt on time match	Scheduling of operations performed on precisely defined times
Supports wake-up mechanism from hibernate state	Autonomous wake up from hibernate for system state control and maintenance routine operations

10.1.2 Block Diagram

The RTC block diagram is shown in [Figure 10-1](#).

Real Time Clock (RTC)

The main building blocks of the RTC is Time Counter implementing real time counter and RTC registers containing multi-field registers for the time counter and alarm programming register. Dedicated fields represent values for elapsing second, minutes, hours, days, days of week, months and years.

The kernel of the RTC module is instantiated in the hibernate domain.

The RTC registers are instantiated in hibernate domain and mirrored in SCU. Access to the RTC registers is performed via register mirror updated over serial interface.

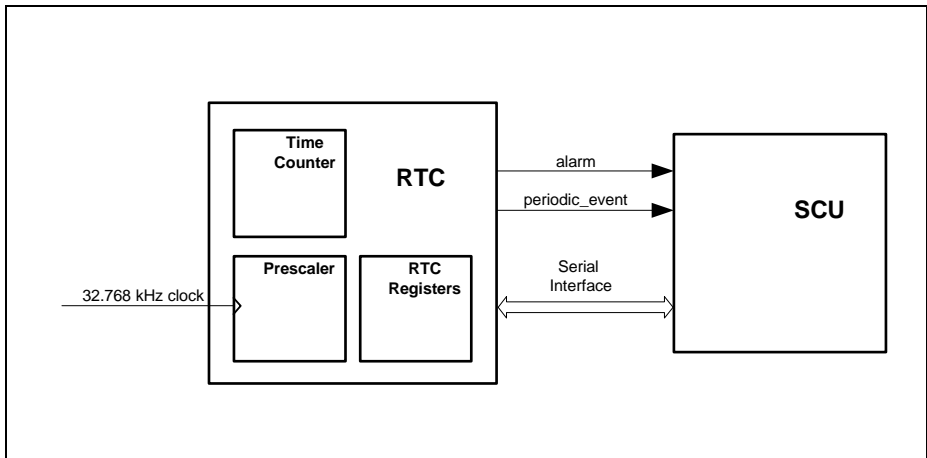


Figure 10-1 Real-Time Clock Block Diagram Structure

10.2 RTC Operation

The RTC timer counts seconds, minutes, hours, days of month, days of week, months and years each in a separate field (see [Figure 10-2](#)). Individual bit fields of the RTC counter can be programmed and read with software over serial interface via mirror registers in SCU module. For details of the serial communication please refer to SCU chapter.

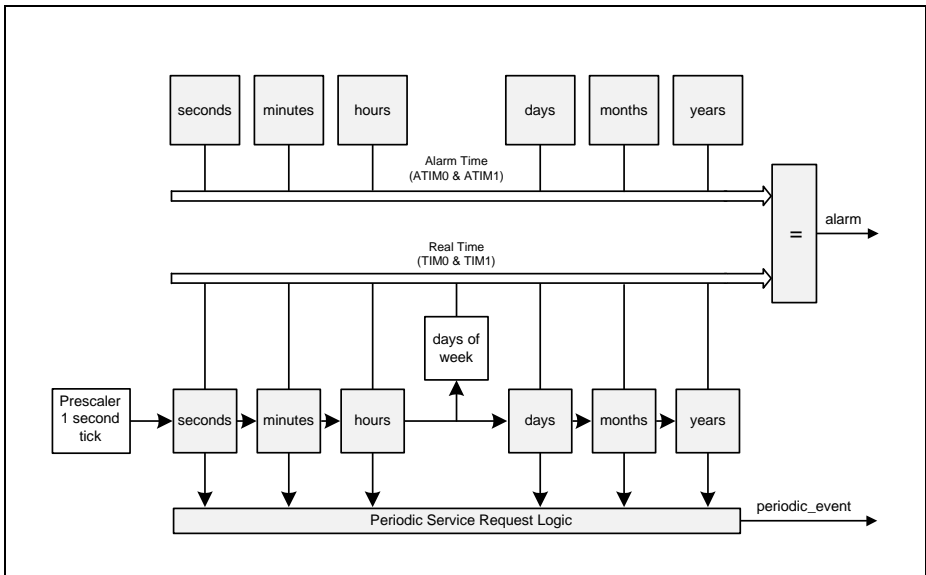


Figure 10-2 Block Diagram of RTC Time Counter

Occurrence of an internal timer event is stored in the service request raw status register **RAWSTAT**. The values of the status register **RAWSTAT** drive the outgoing service request lines alarm and periodic_event.

10.3 Register Access Operations

The RTC module is a part of SCU from programming model perspective and shares register address space for configuration with other sub-modules of SCU. RTC registers are instantiated in hibernate domain and are mirrored in SCU. The registers get updated in both clock domains over serial interface running at 32kHz clock rate.

Any update of the registers is performed with some delay required for data to propagate to and from the mirror registers over serial interface. Accesses to the RTC registers in core domain must not block the bus interface of SCU module. For details of the register mirror and serial communication handling please refer to SCU chapter.

A write to writable registers is only allowed, when the access is in privileged mode. A write access in user mode results in a bus error response.

For consistent write to the timer registers **TIM0** and **TIM1**, the register **TIM0** has to be written before the register **TIM1**. Transfer of the new values from the register mirror starts only after both registers have been written.

After wake-up from hibernate state the content of the mirror registers **TIMO** and **TIM1** is undefined until the first update of the corresponding RTC timers occurs and is propagated to the registers.

10.4 Service Request Processing

The RTC generates service requests upon:

- periodic timer events
- configured alarm condition

The service requests can be processed in the core domain as regular service requests.

10.4.1 Periodic Service Request

The periodic timer service request is raised whenever a non-masked field of the timer counter gets updated. The Periodic Service requests can be enabled/disabled with the **MSKSR** register.

10.4.2 Timer Alarm Service Request

The alarm interrupt is triggered when **TIMO** and **TIM1** bit fields values match all corresponding bit fields values of **ATIMO**, **ATIM1** registers. The Timer Alarm Service requests can be enabled/disabled with the **MSKSR** register.

10.5 Wake-up From Hibernation Trigger

The RTC generates wake-up triggers upon:

- periodic timer events
- configured alarm condition

The timer events can be processed in the hibernate domain as wake-up triggers from hibernate mode, in the HCU module of hibernate domain (for more details please refer to hibernate control description in SCU chapter).

10.5.1 Periodic Wake-up Trigger Generation

The periodic timer wake-up trigger gets generated whenever a non-masked field of the timer counter gets updated. The Periodic Wake-up Trigger generation can be enabled/disabled with the **CTR** register.

10.5.2 Timer Alarm Wake-up Trigger Generation

The alarm wake-up gets trigger gets generated when **TIMO** and **TIM1** bit fields values match all corresponding bit fields values of **ATIMO**, **ATIM1** registers. Timer Alarm Wake-up Trigger generation can be enabled/disabled with the **CTR** register.

10.6 Debug behavior

The RTC clock does not implement dedicated debug mechanisms.

10.7 Power, Reset and Clock

RTC is instantiated entirely in hibernate domain and remains powered up when hibernate domain is powered up. Supply voltage is passed via power switch either from VDDP or VBAT pin as specified in the SCU chapter.

The RTC module remains in reset state along with entire hibernate domain after initial power up of hibernate domain until reset released with software.

The RTC timer is running from either internal or external 32.768 kHz clock selectable with HDCR control register of SCU/HCU module. The prescaler setting of 7FFF_H results in an once per second update of the RTC timer.

10.8 Initialization and Control Sequence

Programming model of the RTC module assumes several scenarios where different control sequences apply.

Note: Some of the scenarios described in this chapter require operations on system level that are not in the scope of the RTC module description, therefore for detailed information please refer to relevant chapters of this document.

10.8.1 Initialization & Start of Operation

Complete RTC module initialization is required upon hibernate domain reset. The hibernate domain needs to be enabled before any programming of RTC registers takes place. Accesses to RTC registers are performed via dedicated mirror registers (for more details please refer to SCU chapter)

- enable hibernate domain (if not disabled)
 - write one to SCU_PWRSET.HIB
- release reset of hibernate domain reset (if asserted)
 - write one to SCU_RSTCLR.HIBRS
- enable RTC module to start counting time
 - write one to RTC_CTR.ENB
- program RTC_TIM0 and RTC_TIM1 registers with current time
 - check SCU_MIRRSTS to ensure that no transfer over serial interface is pending to the RTC_TIM0 register
 - write a new value to the RTC_TIM0 register
 - check SCU_MIRRSTS to ensure that no transfer over serial interface is pending to the RTC_TIM1 register
 - write a new value to the RTC_TIM1 register

10.8.2 Re-configuration & Re-start of Operation

Reset and re-initialization of the RTC module may be required without complete power up sequence of the hibernate domain.

- apply and release reset of hibernate domain reset
 - write one to SCU_RSTSET.HIBRS
 - write one to SCU_RSTCLR.HIBRS
- enable RTC module to start counting time
 - write one to RTC_CTR.ENB
- program RTC_TIM0 and RTC_TIM1 registers with current time
 - check SCU_MIRRSTS to ensure that no transfer over serial interface is pending to the RTC_TIM0 register
 - write a new value to the RTC_TIM1 register
 - check SCU_MIRRSTS to ensure that no transfer over serial interface is pending to the RTC_TIM1 register
 - write a new value to the RTC_TIM1 register

10.8.3 Configure and Enable Periodic Event

The RTC periodic event configuration require programming in order to enable interrupt request generation out upon a change of value in the corresponding bit fields.

- enable service request for periodic timer events in RTC module
 - check SCU_MIRRSTS to ensure that no transfer over serial interface is pending to the RTC_MSKSR register
 - set MAI bit field of RTC_MSKSR register in order enable individual periodic timer events
- enable service request for periodic timer events in RTC module
 - set PI bit field of SCU_SRMSK register in order enable generation of interrupts upon periodic timer events

10.8.4 Configure and Enable Timer Event

The RTC periodic event configuration require programming in order to enable interrupt request generation out upon compare match of values in the corresponding bit fields of TIM0 and TIM1 against ATIM0 and ATIM1 respectively.

- program compare values in individual bit fields of ATIM0 and ATIM1 in RTC module
 - check SCU_MIRRSTS to ensure that no transfer over serial interface is pending to the RTC_ATIM0 register
 - write to RTC_ATIM0 register bit fields
 - check SCU_MIRRSTS to ensure that no transfer over serial interface is pending to the RTC_ATIM1 register
 - write to RTC_ATIM1 register bit fields
- enable service request for timer alarm events in RTC module

Real Time Clock (RTC)

- check SCU_MIRRSTS to ensure that no transfer over serial interface is pending to the RTC_CTR register
- set TAE bit field of RTC_CTR register in order enable individual periodic timer events
- enable service request for timer alarm events in RTC module
 - write one to AI bit field of SCU_SRMSK register in order enable generation of interrupts upon periodic timer events

10.9 Registers

Registers Overview

The absolute register address is calculated by adding:
Module Base Address + Offset Address

Table 10-2 Registers Address Space

Module	Base Address	End Address	Note
RTC	5000 4A00 _H	5000 4BFF _H	Accessible via Mirror Registers

Table 10-3 Register Overview

Short Name	Register Long Name	Offset Addr.	Access Mode		Description
			Read	Write	
RTC Kernel Registers					
ID	ID Register	0000 _H	U, PV	BE	Page 10-8
CTR	Control Register	0004 _H	U, PV	PV	Page 10-9
RAWSTAT	Raw Service Request Register	0008 _H	U, PV	BE	Page 10-11
STSSR	Status Service Request Register	000C _H	U, PV	BE	Page 10-12
MSKSR	Mask Service Request Register	0010 _H	U, PV	PV	Page 10-13
CLRSR	Clear Service Request Register	0014 _H	BE	PV	Page 10-14
ATIM0	Alarm Time Register 0	0018 _H	U,PV	PV	Page 10-15
ATIM1	Alarm Time Register 1	001C _H	U,PV	PV	Page 10-16
TIM0	Time Register 0	0020 _H	U, PV	PV	Page 10-17
TIM1	Time Register 1	0024 _H	U, PV	PV	Page 10-19

10.9.1 Registers Description

ID

Read-only ID register of the RTC module containing unique identification code of the RTC module.

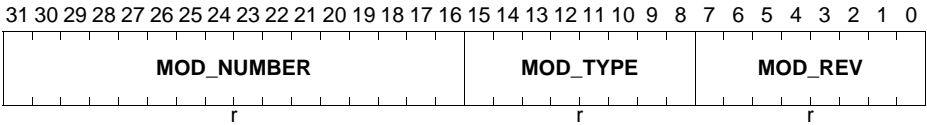
Real Time Clock (RTC)

ID

RTC ID Register

(00_H)

Reset Value: 00A3 C0XX_H



Field	Bits	Type	Description
MOD_REV	[7:0]	r	Module Revision Indicates the revision number of the implementation. This information depends on the design step.
MOD_TYPE	[15:8]	r	Module Type This internal marker is fixed to C0 _H .
MOD_NUMBER	[31:16]	r	Module Number Indicates the module identification number

CTR

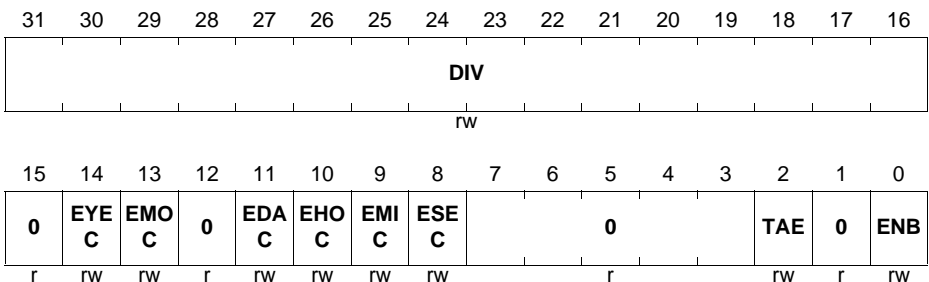
RTC Control Register providing control means of the operation mode of the module.

CTR

RTC Control Register

(04_H)

Reset Value: 7FFF 0000_H



Field	Bits	Type	Description
ENB	0	rw	RTC Module Enable 0 _B disables RTC module 1 _B enables RTC module

Real Time Clock (RTC)

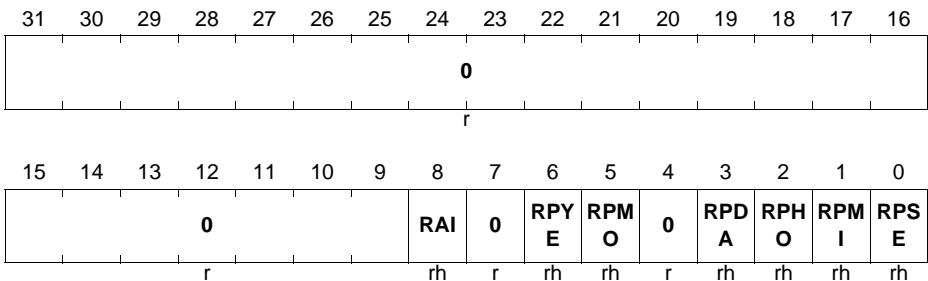
Field	Bits	Type	Description
TAE	2	rw	Timer Alarm Enable for Hibernation Wake-up 0 _B disable timer alarm 1 _B enable timer alarm
ESEC	8	rw	Enable Seconds Comparison for Hibernation Wake-up 0 _B disabled 1 _B enabled
EMIC	9	rw	Enable Minutes Comparison for Hibernation Wake-up 0 _B disabled 1 _B enabled
EHOc	10	rw	Enable Hours Comparison for Hibernation Wake-up 0 _B disabled 1 _B enabled
EDAC	11	rw	Enable Days Comparison for Hibernation Wake-up 0 _B disabled 1 _B enabled
EMOC	13	rw	Enable Months Comparison for Hibernation Wake-up 0 _B disabled 1 _B enabled
EYEC	14	rw	Enable Years Comparison for Hibernation Wake-up 0 _B disabled 1 _B enabled
DIV	[31:16]	rw	RTC Clock Divider Value reload value of RTC prescaler. Clock is divided by DIV+1. 7FFF _H is default value for RTC mode with 32.768 kHz crystal or internal clock
0	1,[7:3], 12,15	r	Reserved Read as 0; should be written with 0

RAWSTAT

RTC Raw Service Request Register contains raw status info i.e. before status mask takes effect on generation of service requests. This register serves debug purpose but can be also used for polling of the status without generating service requests.

RAWSTAT

RTC Raw Service Request Register (08_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RPSE	0	rh	Raw Periodic Seconds Service Request Set whenever seconds count increments
RPMI	1	rh	Raw Periodic Minutes Service Request Set whenever minutes count increments
RPHO	2	rh	Raw Periodic Hours Service Request Set whenever hours count increments
RPDA	3	rh	Raw Periodic Days Service Request Set whenever days count increments
RPMO	5	rh	Raw Periodic Months Service Request Set whenever months count increments
RPYE	6	rh	Raw Periodic Years Service Request Set whenever years count increments
RAI	8	rh	Raw Alarm Service Request Set whenever count value matches compare value
0	4, 7, [31:9]	r	Reserved

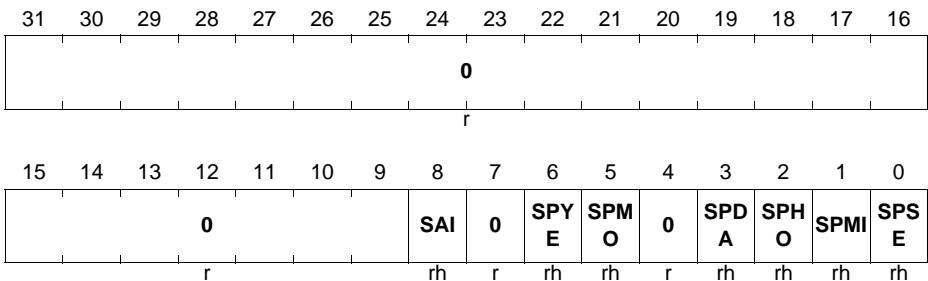
STSSR

RTC Service Request Status Register contains status info reflecting status mask effect on generation of service requests. This register needs to be accessed by software in order to determine the actual cause of an event.

STSSR

RTC Service Request Status Register (0C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SPSE	0	rh	Periodic Seconds Service Request Status after Masking
SPMI	1	rh	Periodic Minutes Service Request Status after Masking
SPHO	2	rh	Periodic Hours Service Request Status after Masking
SPDA	3	rh	Periodic Days Service Request Status after Masking
SPMO	5	rh	Periodic Months Service Request Status after Masking
SPYE	6	rh	Periodic Years Service Request Status after Masking
SAI	8	rh	Alarm Service Request Status after Masking
0	4, 7, [31:9]	r	reserved

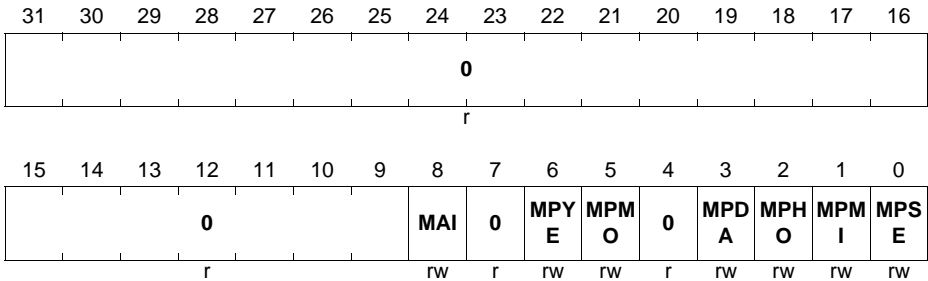
MSKSR

RTC Service Request Mask Register contains masking value for generation control of service requests or interrupts.

MSKSR

RTC Service Request Mask Register (10_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MPSE	0	rw	Periodic Seconds Interrupt Mask 0 _B disable 1 _B enable
MPMI	1	rw	Periodic Minutes Interrupt Mask 0 _B disable 1 _B enable
MPHO	2	rw	Periodic Hours Interrupt Mask 0 _B disable 1 _B enable
MPDA	3	rw	Periodic Days Interrupt Mask 0 _B disable 1 _B enable
MPMO	5	rw	Periodic Months Interrupt Mask 0 _B disable 1 _B enable
MPYE	6	rw	Periodic Years Interrupt Mask 0 _B disable 1 _B enable
MAI	8	rw	Alarm Interrupt Mask 0 _B disable 1 _B enable

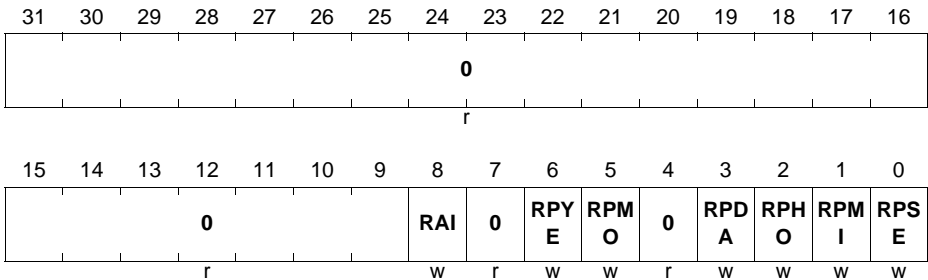
Field	Bits	Type	Description
0	4, 7, [31:9]	r	Reserved

CLRSR

RTC Clear Service Request Register serves purpose of clearing sticky bits of **RAWSTAT** and **STSSR** registers. Write one to a bit in order to clear it is set. Writing zero has no effect on the set nor reset bits.

CLRSR

RTC Clear Service Request Register (14_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RPSE	0	w	Periodic Seconds Interrupt Clear 0 _B no effect 1 _B clear status bit
RPMI	1	w	Periodic Minutes Interrupt Clear 0 _B no effect 1 _B clear status bit
RPHO	2	w	Periodic Hours Interrupt Clear 0 _B no effect 1 _B clear status bit
RPDA	3	w	Periodic Days Interrupt Clear 0 _B no effect 1 _B clear status bit
RPMO	5	w	Periodic Months Interrupt Clear 0 _B no effect 1 _B clear status bit

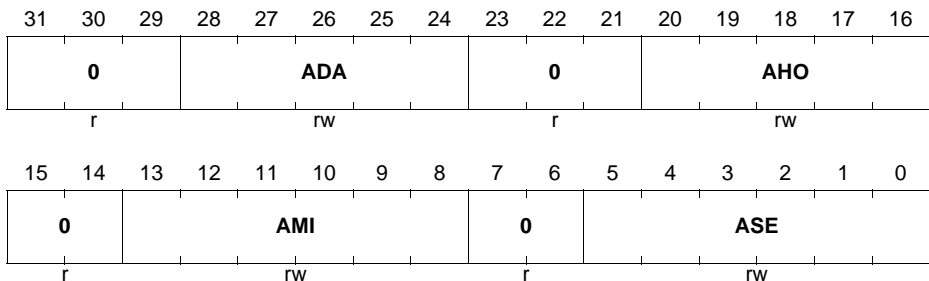
Field	Bits	Type	Description
RPYE	6	w	Periodic Years Interrupt Clear 0 _B no effect 1 _B clear status bit
RAI	8	w	Alarm Interrupt Clear 0 _B no effect 1 _B clear status bit
0	4, 7, [31:9]	r	Reserved

ATIMO

RTC Alarm Time Register 0 serves purpose of programming single alarm time at a desired point of time reflecting comparison configuration in the **CTR** for individual fields against **TIMO** register. The register contains portion of bit fields for seconds, minutes, hours and days. Upon attempts to write an invalid value to a bit field e.g. exceeding maximum value default value gets programmed as described for each individual bit fields.

ATIMO

RTC Alarm Time Register 0 (18_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ASE	[5:0]	rw	Alarm Seconds Compare Value Match of seconds timer count to this value triggers alarm seconds interrupt. Setting value equal or above 3C _H results in setting the field value to 0 _H

Real Time Clock (RTC)

Field	Bits	Type	Description
AMI	[13:8]	rw	Alarm Minutes Compare Value Match of minutes timer count to this value triggers alarm minutes interrupt. Setting value equal or above 3C _H results in setting the field value to 0 _H
AHO	[20:16]	rw	Alarm Hours Compare Value Match of hours timer count to this value triggers alarm hours interrupt. Setting value equal or above 18 _H results in setting the field value to 0 _H
ADA	[28:24]	rw	Alarm Days Compare Value Match of days timer count to this value triggers alarm days interrupt. Setting value equal above 1F _H results in setting the field value to 0 _H
0	[7:6], [15:14], [23:21], [31:29]	r	Reserved

ATIM1

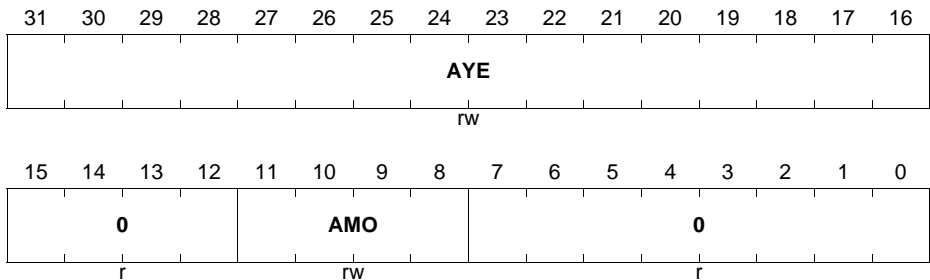
RTC Alarm Time Register 1 serves purpose of programming single alarm time at a desired point of time reflecting comparison configuration in the **CTR** for individual fields against **TIM1** register. The ATM1 register contains portion of bit fields for days of week, months and years. Upon attempts to write an invalid value to a bit field e.g. exceeding maximum value default value gets programmed as described for each individual bit fields.

ATIM1

RTC Alarm Time Register 1

(1C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
AMO	[11:8]	rw	Alarm Month Compare Value Match of months timer count to this value triggers alarm month interrupt. Setting value equal or above the number of days of the actual month count results in setting the field value to 0 _H
AYE	[31:16]	rw	Alarm Year Compare Value Match of years timer count to this value triggers alarm years interrupt.
0	[7:0], [15:12]	r	Reserved

TIM0

RTC Time Register 0 contains current time value for seconds, minutes, hours and days. The bit fields get updated in intervals corresponding with their meaning accordingly. The register needs to be programmed to reflect actual time after initial power up and will continue counting time also while in hibernate mode. Upon attempts to write an invalid value to a bit field e.g. exceeding maximum value a default value gets programmed as described for each individual bit fields.

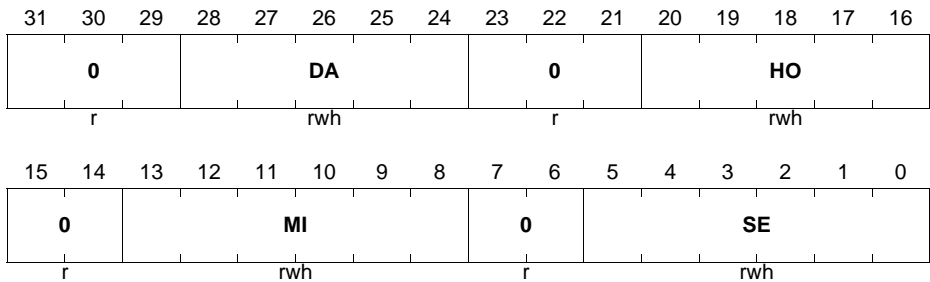
Real Time Clock (RTC)

TIM0

RTC Time Register 0

(20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SE	[5:0]	rwh	<p>Seconds Time Value</p> <p>Setting value equal or above 3C_H results in setting the field value to 0_H. Value can only be written, when RTC is disabled. After wake-up from hibernate, value is undefined until first update of RTC.</p>
MI	[13:8]	rwh	<p>Minutes Time Value</p> <p>Setting value equal or above 3C_H results in setting the field value to 0_H. Value can only be written, when RTC is disabled. After wake-up from hibernate, value is undefined until first update of RTC.</p>
HO	[20:16]	rwh	<p>Hours Time Value</p> <p>Setting value equal or above 18_H results in setting the field value to 0_H. Value can only be written, when RTC is disabled. After wake-up from hibernate, value is undefined until first update of RTC.</p>

Real Time Clock (RTC)

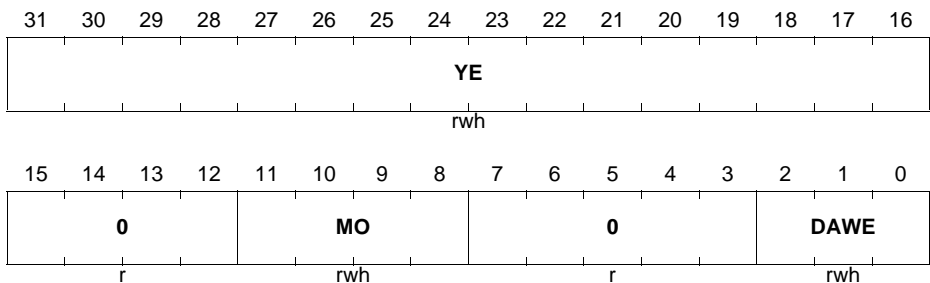
Field	Bits	Type	Description
DA	[28:24]	rwh	Days Time Value Setting value equal or above the number of days of the actual month count results in setting the field value to 0 _H Value can only be written, when RTC is disabled. After wake-up from hibernate, value is undefined until first update of RTC. Days counter starts with value 0 for the first day of month.
0	[7:6], [15:14], [23:21], [31:29]	r	Reserved

TIM1

RTC Time Register 1 contains current time value for days of week, months and years. The bit fields get updated in intervals corresponding with their meaning accordingly. The register needs to be programmed to reflect actual time after initial power up and will continue counting time also while in hibernate mode. Upon attempts to write an invalid value to a bit field e.g. exceeding maximum value a default value gets programmed as described for each individual bit fields.

TIM1

RTC Time Register 1 (24_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
DAWE	[2:0]	rwh	Days of Week Time Value Setting value above 6 _H results in setting the field value to 0 _H . Value can only be written, when RTC is disabled. After wake-up from hibernate, value is undefined until first update of RTC. Days counter starts with value 0 for the first day of week.
MO	[11:8]	rwh	Month Time Value Setting value equal or above C _H results in setting the field value to 0 _H . Value can only be written, when RTC is disabled. After wake-up from hibernate, value is undefined until first update of RTC. Months counter starts with value 0 for the first month of year.
YE	[31:16]	rwh	Year Time Value Value can only be written, when RTC is disabled. After wake-up from hibernate, value is undefined until first update of RTC.
0	[7:3], [15:12]	r	Reserved

10.10 Interconnects

Table 10-4 Pin Connections

Input/Output	I/O	Connected To	Description
Clock Signals			
f_{RTC}	I	SCU.HCU	32.768 kHz clock selected in hibernate domain
Service Request Connectivity			
periodic_event	O	SCU.GCU	Timer periodic service request
alarm	O	SCU.GCU	Alarm service request

11 System Control Unit (SCU)

The SCU is the SoC power, reset and a clock manager with additional responsibility of providing system stability protection and other auxiliary functions.

11.1 Overview

The functionality of the SCU described in this chapter is organized in the following sub-chapters, representing different aspects of system control:

- Miscellaneous control functions, [Chapter 11.2](#)
- Power Control, [Chapter 11.3](#)
- Hibernate Control, [Chapter 11.4](#)
- Reset Control, [Chapter 11.5](#)
- Clock Control, [Chapter 11.6](#)

11.1.1 Features

The following features are provided for monitoring and controlling the system:

- General Control
 - Boot Mode Detection
 - Memory Parity Protection
 - Trap Generation
 - Die Temperature Measurement
 - Retention Memory Support
- Power Control
 - Power Sequencing
 - EVR Control
 - Supply Watchdog
 - Voltage Monitoring
 - Power Validation
 - Power State Indication
 - Flash Power Control
- Hibernate Control
 - Hibernate Mode Control
 - Wake-up from Hibernate Mode
 - Hibernate Domain Control
- Reset Control
 - Reset assertion on various reset request sources
 - System Reset Generation
 - Inspection of Reset Sources After Reset
 - Selective Module reset
- Clock Control

- Individual peripheral clock gating
- Input clock selection
- Clock Generation
- Clock Distribution
- Clock Supervision
- Power Management
- RTC Clock

11.1.2 Block Diagram

The block diagram shown in [Figure 11-1](#) reflects logical organization of the System Control Unit.

- Power Control Unit (PCU)
- Hibernate Control Unit (HCU)
- Reset Control Unit (RCU)
- General Control Unit (GCU)
- Clock Control Unit (CCU)

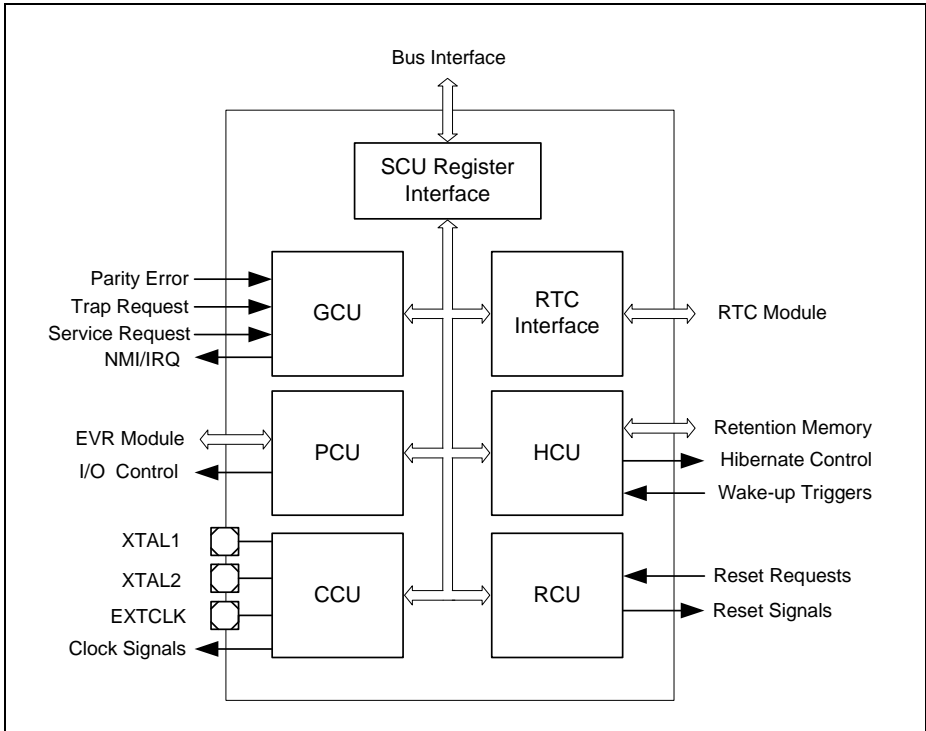


Figure 11-1 SCU Block Diagram

Interface of General Control Unit

The General Control Unit GCU has a memory fault interface to the memory validation logic of each on-chip SRAM and the Flash to receive memory fault events, as parity errors. NMI request are routed to the unit to be gated and combined. The GCU provides the start up protection to all units, which require an additional level of protection.

Interface of Power Control Unit

The Power Control Unit PCU has an interface to the Embedded Voltage Regulator (EVR) and an interface to the PORTS module. The PCU related signals are described in more detail in [Chapter 11.3](#).

Interface of Reset Control Unit

The Reset Control Unit RCU has an interface to the Embedded Voltage Regulator (EVR). The RCU receives from the EVR the power-on reset and the reset information for

System Control Unit (SCU)

each power related reset. Reset requests are coming to the unit from the watchdog, the CPU and the test control unit. The RCU is providing the reset signals to all other units of the chip in the Core power domain. The RCU related signals are described in more detail in [Chapter 11.5](#).

Interface of Clock Control Unit

The Clock Control Unit (CCU) receives the external clock source via the crystal pins XTAL1 and XTAL2. As further clock source the CCU receives the standby clock f_{STDBY} from the Hibernate domain. The CCU drives an external clock output, where internal clocks can be routed out. The CCU provides the clock signals to all other units of the chip.

Interface of Hibernate Power Domain

The interface to the Hibernate domain provides mirror registers updated via a shared serial interface to the Retention Memory, RTC module registers and Hibernate domain control registers. Update of the mirror registers over the serial is controlled using [MIRRSTS](#), [RMDATA](#) and [RMACR](#) registers. End of update can also trigger service requests via [SRSTAT](#) register. Refresh of the Hibernate domain registers in the register mirror is performed continuously, in order to instantly reflect any register state change on both sides. The serial interface is inactivated while in hibernate mode in order to reduce current consumption.

Interface of Retention Memory

Access to the Retention Memory is served over shared serial interface identical to the one used to access Hibernate domain registers, for detail please refer to [“Interface of Hibernate Power Domain” on Page 11-4](#).

Interface of RTC

Access to the RTC module is served over shared serial interface identical to the one used to access Hibernate domain registers, for detail please refer to [“Interface of Hibernate Power Domain” on Page 11-4](#). The RTC module functionality is described in separate RTC chapter.

11.2 Miscellaneous Control Functions

System Control implements system management functions accessible via GCU registers. General system control including various auxiliary function is performed in General Control Unit (GCU).

11.2.1 Startup Software Support

Externally driven boot mode pins determine the boot mode after a power on reset. It also possible for applications to decide the boot mode. Ability to determine boot mode values on boot mode pins and user application desired boot modes is provided by SCU.

11.2.2 Service Requests

Service request events listed in [Table 11-1](#) can result in assertion of a regular interrupt or an NMI. Please refer to [SRMSK](#) and [NMIREQEN](#) register description.

The interrupt structure is shown in [Figure 11-2](#). The interrupt request or the corresponding interrupt set bit (in register [SRSET](#)) can trigger the interrupt generation at the selected interrupt node x. The service request pulse is generated independently from the interrupt flag in register [SRSTAT](#). The interrupt flag can be cleared by software by writing to the corresponding bit in register [SRCLR](#). In addition several service requests can be promoted to NMI trigger level using [NMIREQEN](#) register

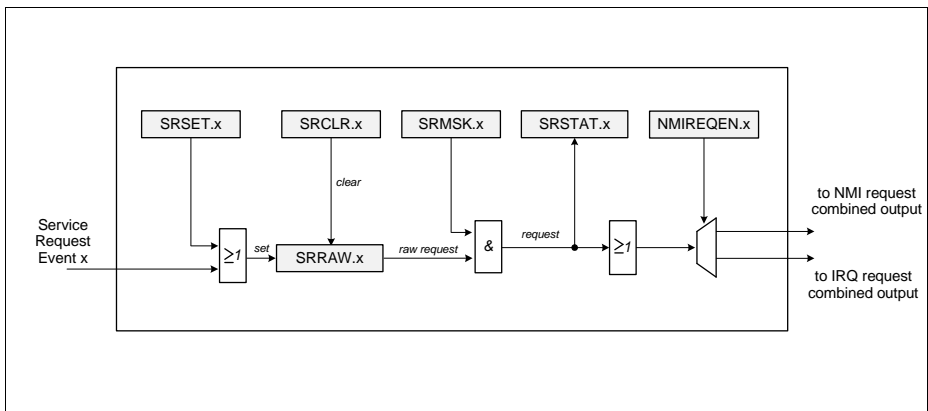


Figure 11-2 Service Request Subsystem

The service request flag in register [SRSTAT](#) can be cleared by software by writing to the corresponding bit in register [SRCLR](#). All service requests are combined to one common line and connected to a regular interrupt node or NMI node of NVIC.

The service requests have a sticky flag in register [SRRAW](#).

Note: When servicing an SCU service request, make sure that all related request flags are cleared after the identified request has been handled.

11.2.2.1 Service Request Sources

The SCU supports service request sources listed in [Table 11-2](#) and reflected in the [SRSTAT](#), [SRRAW](#), [SRMSK](#), [SRCLR](#) and [SRSET](#) registers.

Table 11-1 Service Requests

Service Request Name	Service Request Short Name
WDT pre-warning	PRWARN
RTC Periodic Event	PI
RTC Alarm	AI
DLR Request Overrun	DLROVR
HDCLR Mirror Register Updated	HDCLR
HDSET Mirror Register Updated	HDSET
HDCR Mirror Register Updated	HDCR
OSCSICTRL Mirror Register Updated	OSCSICTRL
OSCULCTRL Mirror Register Updated	OSCULCTRL
RTC CTR Mirror Register Updated	RTC_CTR
RTC ATIM0 Mirror Register Updated	RTC_ATIM0
RTC ATIM1 Mirror Register Updated	RTC_ATIM1
RTC TIM0 Mirror Register Updated	RTC_TIM0
RTC TIM1 Mirror Register Updated	RTC_TIM1
Retention Memory Mirror Register Updated	RMX

11.2.3 Memory Parity Protection

For supervising the content of the on-chip memories, the following mechanism is provided:

All on-chip SRAMs provide protection of integrity via parity checking. The parity logic generates additional parity bits which are stored along with each data word at a write operation. A read operation implies checking of the previous stored parity information.

An occurrence of a parity error is observable at the memory error raw status register. It is configurable whether a memory error should trigger an NMI or System Reset.

11.2.3.1 Parity Error Handling

The on-chip RAM modules check parity information during read accesses and in case of an error a signal can be generated if enabled with **PEEN** register. Two modes of parity error signalling are implemented:

- bus error
- parity error trap (NMI)

The bus error generation applies to memories that can be accessed directly from the bus system level. Apart from that, all memories, including those that are not accessible

System Control Unit (SCU)

directly and are internal to peripherals are capable of generating system traps resulting in NMI. Parity trap requests get enabled with **PETE** register implementing individual control for each memory. Parity error signalling with trap generation is not recommended to be used for memories capable of bus error generation and therefore should be disabled.

Parity error trap generation mechanism can be also used to generate System Reset if enabled with **PERSTEN** register in conjunction with the **PETE** register configuration. For more details of the parity error generation scheme please refer to **Figure 11-3** .

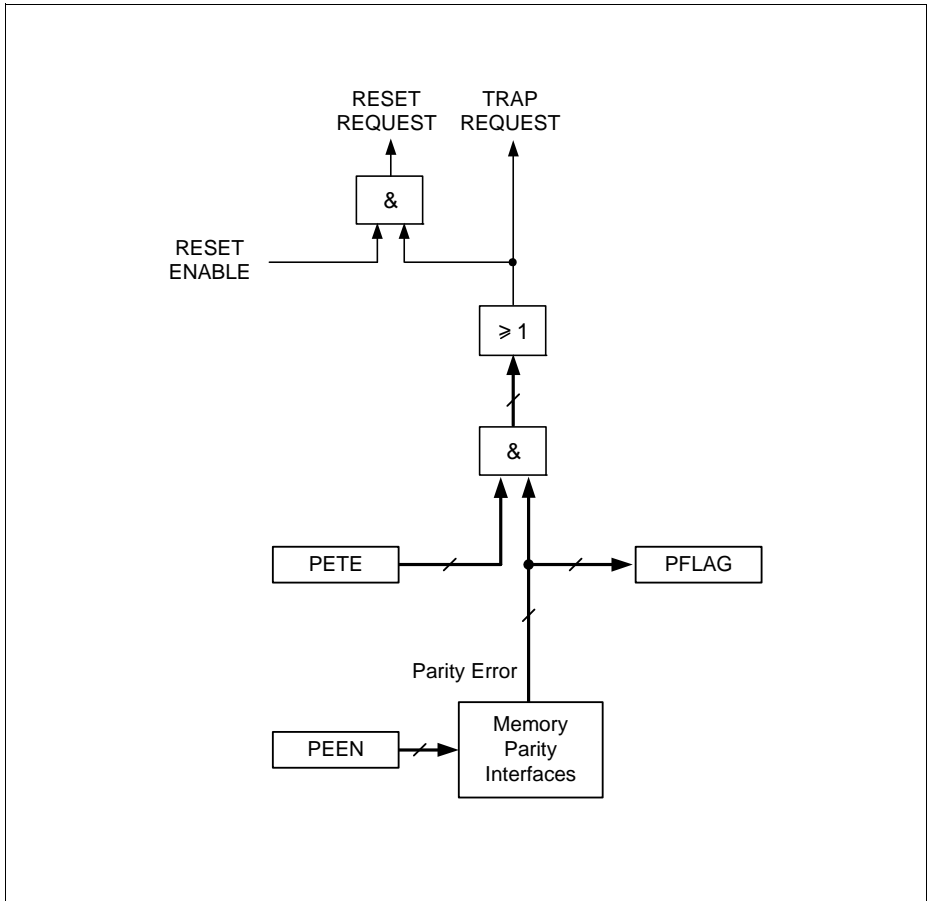


Figure 11-3 Parity Error Control Logic

System Control Unit (SCU)

The logic is controlled by registers **PMTSR** and **PMTPR**. Via bit field **PMTPR.PWR** a parity value can be written to any address of every memory for software driven testing purpose. The parity control software test update has to be enabled with bit **PMTSR** for each memory individually. Otherwise a write to the parity control has no effect. With each read access to a memory the parity from the memory parity control is stored in register **PMTPR** and accessible with software.

11.2.4 Trap Generation

Several abnormal events listed in **Table 11-2** can result in assertion of NMI. Please refer to **TRAPDIS** register description.

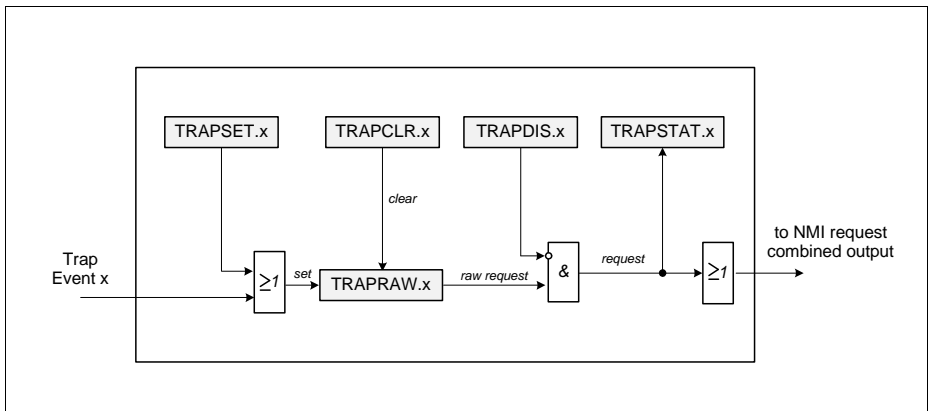


Figure 11-4 Trap Subsystem

The trap flag in register **TRAPSTAT** can be cleared by software by writing to the corresponding bit in register **TRAPCLR**. All trap requests are combined to one common line and connected to NMI node of NVIC.

The trap requests have a sticky flag in register **TRAPRAW**.

Note: When servicing an SCU trap request, make sure that all related request flags are cleared after the identified request has been handled.

11.2.4.1 Trap Sources

The SCU supports trap sources listed in **Table 11-2** and reflected in the **TRAPSTAT**, **TRAPRAW**, **TRAPDIS**, **TRAPCLR** and **TRAPSET** registers.

Table 11-2 SCU Trap Request Overview

Source of Trap	Short Trap Name
OSC_HP Oscillator Watchdog Trap	SOSCWDT
USB VCO Lock Trap	UVCOLCKT
System VCO Lock Trap	SVCOLCKT
Parity Error Trap	PET
Brownout Trap	BRWNT
OSC_ULP Oscillator Watchdog Trap	ULPWDGT
Peripheral Bus 0 Write Error Trap	BWERR0T
Peripheral Bus 1 Write Error Trap	BWERR1T

11.2.5 Die Temperature Measurement

The Die Temperature Sensor (DTS) generates a measurement result that indicates directly the current temperature. The result of the measurement is displayed via bit field **DTSSTAT.RESULT**. In order to start one measurement bit **DTSCON.START** needs to be set.

The DTS has to be enabled before it can be used via bit **DTSCON.PWD**. When the DTS is powered temperature measurement can be started.

In order to adjust production variations of temperature measurement accuracy bit field **DTSCON.BGTRIM** is provided. **DTSCON.BGTRIM** can be programmed by the user software.

Measurement data is available certain time after measurement started. Register **DTSSTAT.RDY** bit indicated that the DTS is ready to start a measurement. If a started measurement is finished or still in progress is indicated via the status bit **DTSSTAT.BUSY**.

The formula to calculate the die temperature is defined in the Target Data Sheet.

Note: The first measurement after the DTS was powered delivers a result without calibration adjustment and should be ignored.

11.2.5.1 Temperature Measurement

The default formula can be applied in order to calculate the die temperature expressed in °C:

$$T_{DTS} = (\text{RESULT} - 605) / 2.05 \text{ [}^\circ\text{C]}$$

Where RESULT is a value of RESULT bit field in **DTSSTAT** register.

System Control Unit (SCU)

The accuracy of the measurement can be improved with adjustment of the OFFSET and GAIN bit fields in the **DTSCON** register.

The following formula can be applied in order to reflect relation between the actual adjustment settings and the resulting value in the **DTSSTAT** register:

$$\text{RESULT}_{\text{CALIB}} = \text{RESULT}_{\text{UNCALIB}} * (1023 / (1023 - \text{GAIN})) + \text{OFFSET}$$

Where the $\text{RESULT}_{\text{CALIB}}$ is the value read from the RESULT bit field of the **DTSSTAT** register, reflecting the effect of gain and offset calibration.

Hence, the default formula above is a special case with the default calibration values and can be substituted with:

$$T_{\text{DTS}} = (\text{RESULT}_{\text{CALIB}} - 605) / 2.05 \text{ [}^\circ\text{C]}$$

*Note: Please always apply the following values to the bit fields of the **DTSCON** register:*

5. **DTSCON.REFTRIM** = 4_H
6. **DTSCON.BGTRIM** = 8_H

11.2.5.2 Offset Adjustment

Offset Adjustment is defined as a shift of the result. The range of the offset adjustment is 7 bits with a $\frac{1}{2}$ LSB resolution, which corresponds to ± 12.5 °C.

The offset can be adjusted with OFFSET bit field of **DTSCON** register. The range of the result is limited to 1023_D . The bits are in twos complement format, with the MSB as the sign bit.

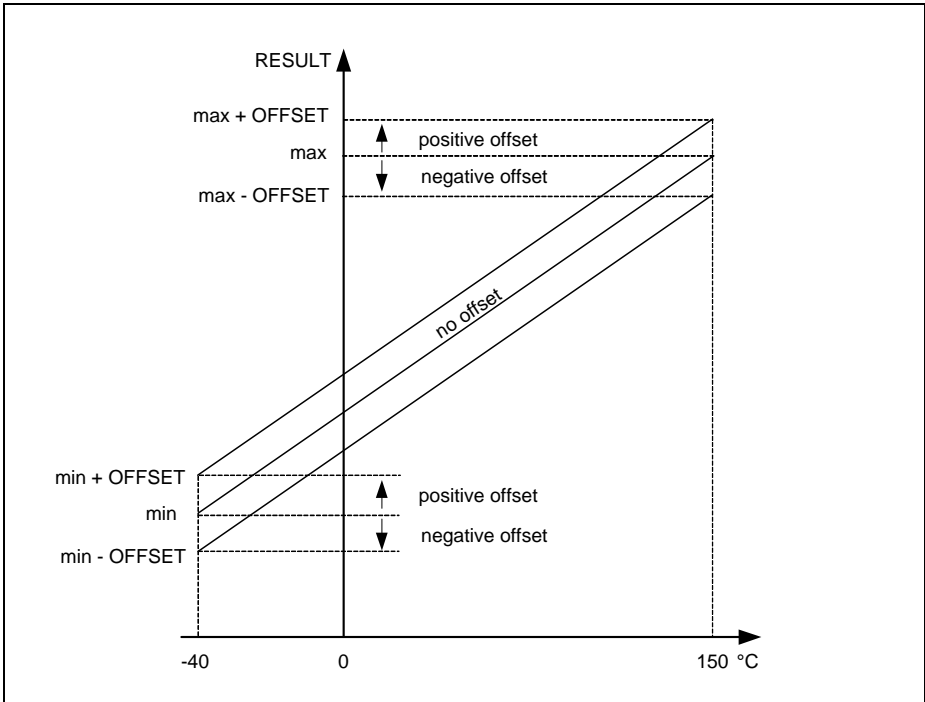


Figure 11-5 Offset adjustment of the RESULT curve

Offset adjustment has direct effect on the RESULT value in **DTSSTAT** register as illustrated in **Figure 11-5**.

11.2.5.3 Gain Adjustment

The most convenient way to express effect of gain on the result is by a curve, fitted to pass from 0 level point to its maximum value at the highest value in the RESULT bit field of **DTSSTAT** register register as illustrated in **Figure 11-6**. The range of the RESULT is limited to 1023_D .

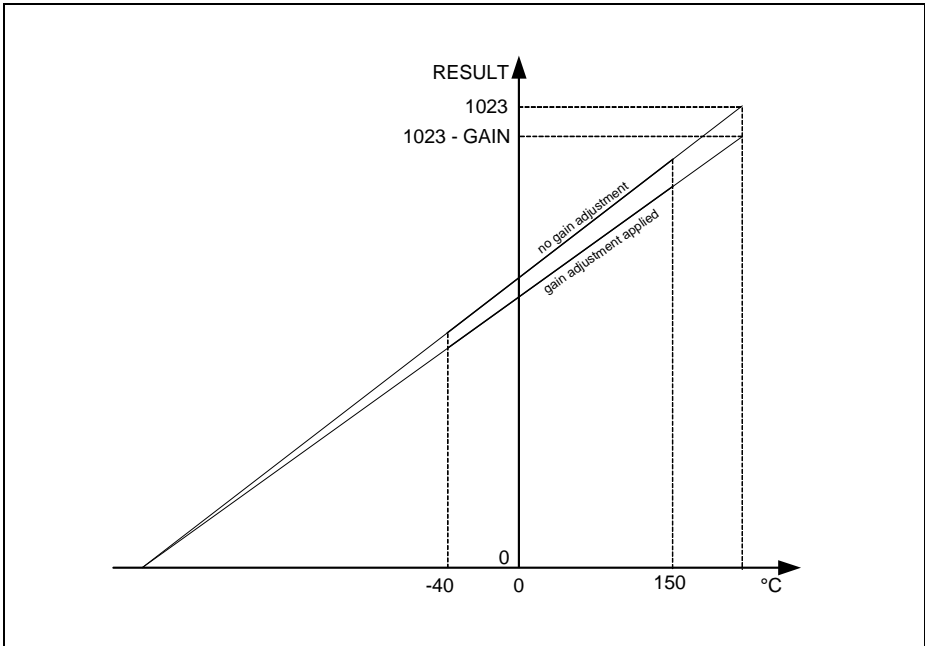


Figure 11-6 Gain adjustment of the RESULT curve

The gain error can be minimized using 6-bit GAIN bit field of **DTSCON** register where 0 corresponds with a maximum gain and value of 63 corresponds with minimum gain i.e. RESULT value reduced by 63 at the upper imaginary end of the curve.

11.2.6 Retention Memory

Retention memory for context store/restore is available for hibernate mode support. The retention memory is located in Hibernate domain and accessible via regular memory address space. Retention memory content is retained in hibernate mode, while the core might be wpowered off. The user software can store some context critical data in the memory before entering hibernate mode. Access to the data after booting up can be performed at any time with user software. Occurrence of a wake-up from hibernate mode is signaled in **RSTSTAT** register.

Access to the 64 Bytes of retention memory is provided via **RMDATA** register, controlled with **RMACR** register. The purpose of **RMACR** register is addressing of the memory cells and issuing read/write command. The **RMDATA** is read or written by software according to the access direction after data transfer has been completed as indicated with RMX bit field in **MIRRSTS**.

11.2.7 Out of Range Comparator Control

The out of range comparator serves the purpose of overvoltage monitoring for analog input pins of the chip. A number of analog channels are associated with dedicated pads connected to inputs of analog modules. They get supervised by dedicated circuits constituted of analog comparator controlled by digital signals as depicted in [Figure 11-7](#).

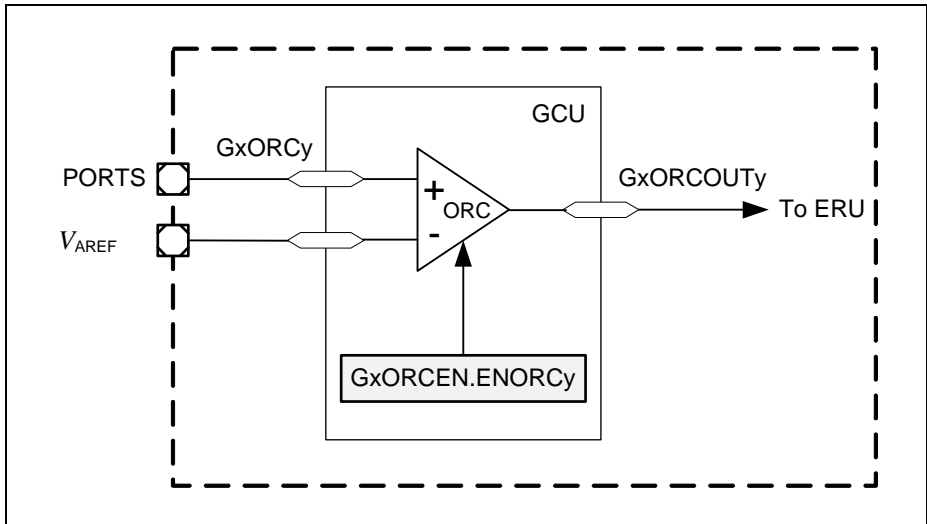


Figure 11-7 Out of Range Comparator Control

Detection of input voltage exceeding V_{AREF} triggers a service request to the ERU. Digital control signals are generated in GCU submodule of SCU. Dedicated registers [G0ORCEN](#) and [G1ORCEN](#) provide control means for enabling and disabling monitoring of analog channels.

11.3 Power Management

Power management control is performed with the Power Control Unit (PCU).

11.3.1 Functional Description

The XMC4300 is running from a single external power supply (V_{DDP}). The main supply voltage is supervised by a supply watchdog.

The I/Os and the main part of the flash block are running directly from the external supply voltage. The core voltage (V_{DDC}) is generated by an on-chip Embedded Voltage Regulator (EVR). The safe voltage range of the core voltage is supervised by a power validation circuit, which is part of the EVR.

System Control Unit (SCU)

Logic in the hibernate domain, mainly the real-time clock RTC, hibernate control and retention memory, is supplied by an auxiliary power supply using an additional power pad. The auxiliary V_{BAT} voltage, supplied from e.g. a coin battery, enables the RTC to operate while the main supply is switched off.

11.3.2 System States

The system has the following general system states:

- Active
- Sleep
- Deep Sleep
- Hibernate

Figure 11-8 shows the state diagram and the transitions between these power modes. The additional state power-up is only a transient state which is passed on cold or warm start-up from Off state.

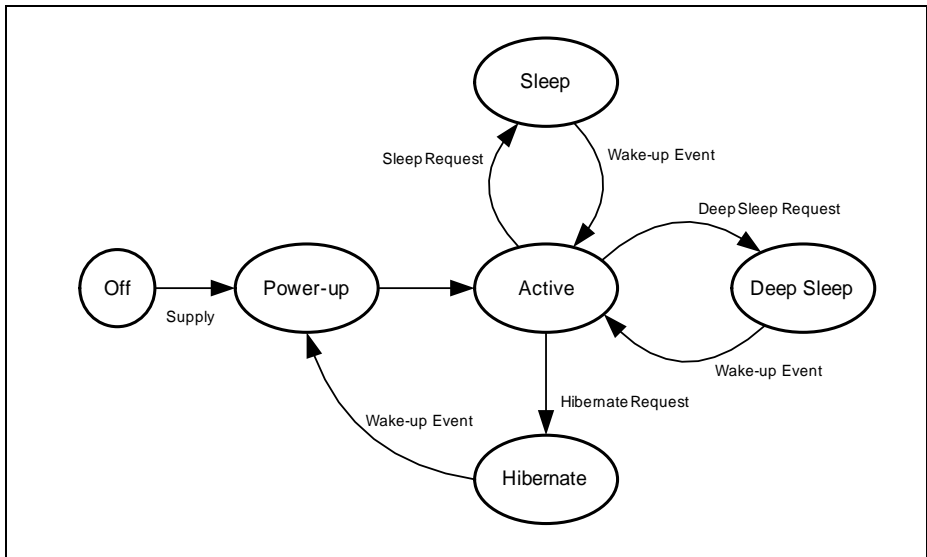


Figure 11-8 System States Diagram

Active State

The Active state is the normal operation state. The system is fully powered. The CPU is usually running from a high-speed clock. Depending on the application the system clock might be slowed down. The PLL output clock or another clock can be selected as clock

System Control Unit (SCU)

source. Unused peripherals might be stopped. Stopping a peripheral means that the peripheral is put into reset and the clock to this peripherals is disabled.

After a cold start the hibernate domain stays disabled until activated by user code.

Sleep State

The Sleep state of the system corresponds to the Sleep state of the CPU. The state is entered via WFI or WFE instruction of the CPU. In this state the clock to the CPU is stopped. The source of the system clock may be altered. Peripherals clocks are gated according to the **SLEEP** register.

Peripherals can continue to operate unaffected and eventually generate an event to wake-up the CPU. Any interrupt to the NVIC will bring the CPU back to operation. The clock tree upon exit from SLEEP state is restored to what it was before entry into SLEEP state.

Deep Sleep State

The Deep Sleep state is entered on the same mechanism as the Sleep state. User code configures the Deep Sleep state in **DSLEEP** control register. In Deep Sleep state the OSC_HP and the PLL may be switched off. The wake-up logic in the NVIC is still clocked by a free-running clock. Peripherals are only clocked when configured to stay enabled in the **DSLEEP** register. Configuration of peripherals and any SRAM content is preserved.

The Flash module can be put into low-power mode to achieve a further power reduction. On wake-up Flash module will be restarted again before instructions or data access is possible.

Any interrupt will bring the system back to operation via the NVIC. The clock setup before entering Deep Sleep state is restored upon wake-up.

Hibernate State

In Hibernate mode the power supply to the core is switched off. Additionally the power to the analog domain and the main supply V_{DDP} can be switched off. Only the Hibernate power domain will stay powered. The power supply of the Hibernate domain is switched automatically to the auxiliary supply when the main supply is no longer present.

The Hibernate State is entered using control register **HDCR** of HCU in the Hibernate domain that will drive the external Voltage Regulator with HIBOUT signal to switch off power to the chip (see **System Level Power Control example - externally controlled with two pins**).

Depending on configuration the following wake-up sources will wake-up the system to normal operation:

- Edge detection on external WKUP signal

- RTC Alarm Event
- RTC Periodic Event
- OSC_ULP Watchdog Event

The system can only wake-up from Hibernate if V_{DDP} is present. An external power supply can be switched on by the HIBOUT signal of the Hibernate Control Unit.

All blocks outside of the Hibernate domain will see a complete power-up sequence upon wake-up.

11.3.3 Hibernate Domain Operating Modes

The standard use case of the Hibernate domain is the time keeping function with V_{BAT} available, keeping the Real Time Counter active and data in Retention Memory while the System Supply is off.

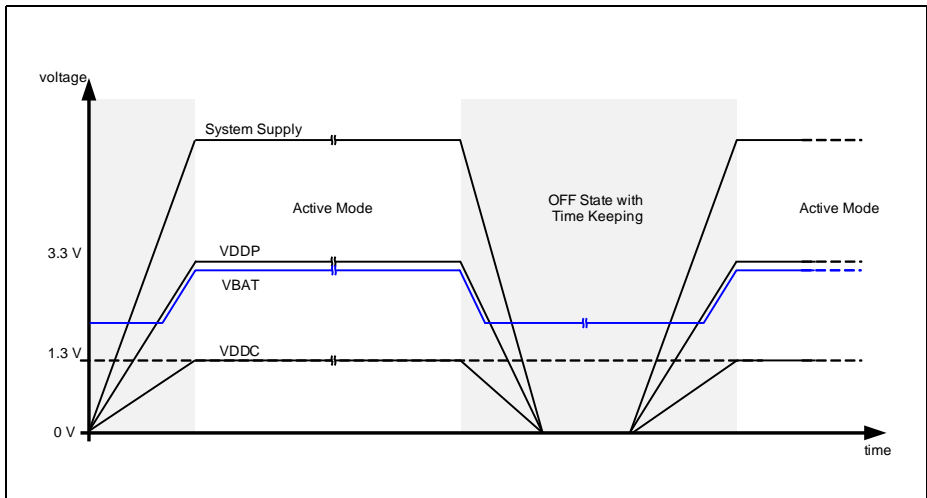


Figure 11-9 Hibernate state in time keeping mode

A special case of the Hibernate domain assumes that the V_{BAT} supply voltage is available only after the core domain power-up. This may occur if, for example, the battery gets plugged in or replaced after the core domain startup, or, if only a capacitor is available to hold V_{BAT} voltage for some limited time in case of absence of the main supply voltage in order to keep the Real Time Counter unaffected. The Hibernate domain requires to be switched on once after core domain power up with the dedicated register **PWRSET.HIB**. In this application case even switching off the main supply of the board does not affect availability of the V_{HIB} voltage required to keep RTC running.

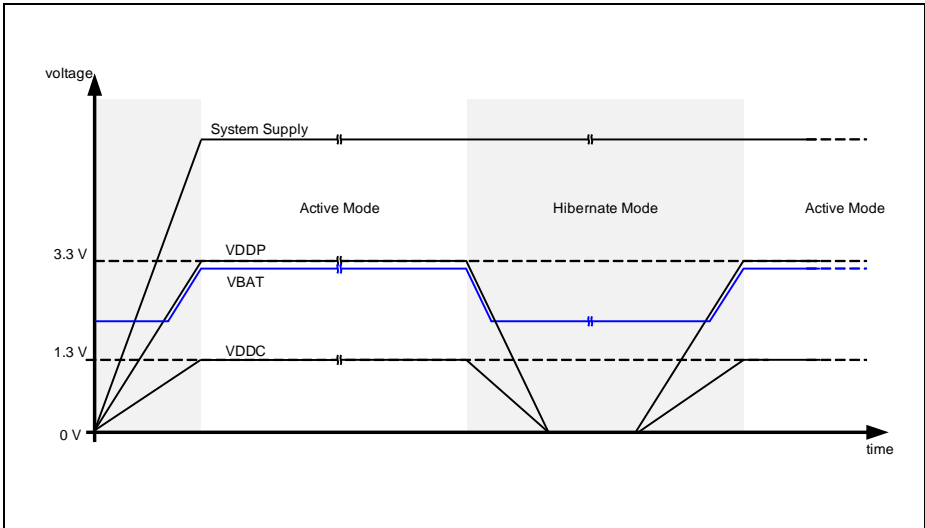


Figure 11-10 Hibernate controlled with external voltage regulator

The externally controlled Hibernate mode is realized with HIB pin and external power supply device. As illustrated in [Figure 11-10](#) the main supply voltage of the board stays active allowing selective disabling of the XMC4300 in order to save power while other devices of on the board remain active. At the time HIB pin indicates Hibernate mode the dedicated voltage supply connected to the chip will stop generating V_{DDP} and voltage and in effect complete chip except the Hibernate domain will be powered off. On a wake-up event the Hibernate domain will deactivate the HIB control signal and enable generation of V_{DDP} voltage, hence complete power-up sequence of the core domain.

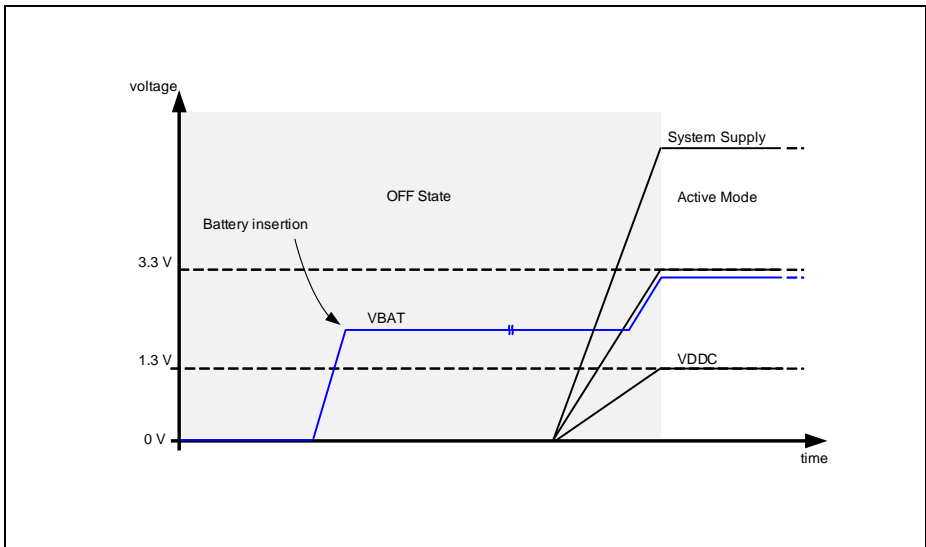


Figure 11-11 Initial power-up sequence

One of the valid power-up scenarios assumes that battery will be installed, possibly soldered, before core supply is available (see [Figure 11-11](#)). That would be a common situation after PCB assembly, before the complete device is installed in the target system. The battery voltage V_{BAT} gets connected before main supply of the board is available and only the Hibernate domain is supplied. No current (only leakage is allowed) is drawn from the battery before explicit switching on Hibernate domain with software after Core domain power up using dedicated register [PWRSET.HIBEN](#). This feature allows to keep the device in a storage for a long time before shipment to the end user, without significant loss of charge in the battery.

11.3.4 Embedded Voltage Regulator (EVR)

The EVR generates the core voltage V_{DDC} out of the external supplied voltage V_{DDP} . The EVR provides a supply watchdog (SWD) for the input voltage V_{DDP} . The generated core voltage V_{DDC} is monitored by a power validation circuit (PV).

11.3.5 Power-on Reset

The EVR starts operation as soon as V_{DDP} is above defined minimum level. It releases the reset, when the external voltage V_{DDP} and the generated voltage V_{DDC} are above the reset thresholds and reaching the nominal values.

11.3.6 Supply Watchdog (SWD)

Figure 11-12 shows the operation of the supply monitor. The supply watchdog compares the supply voltage against the reset threshold V_{POR} . The Data Sheet defines the nominal value and applied hysteresis.

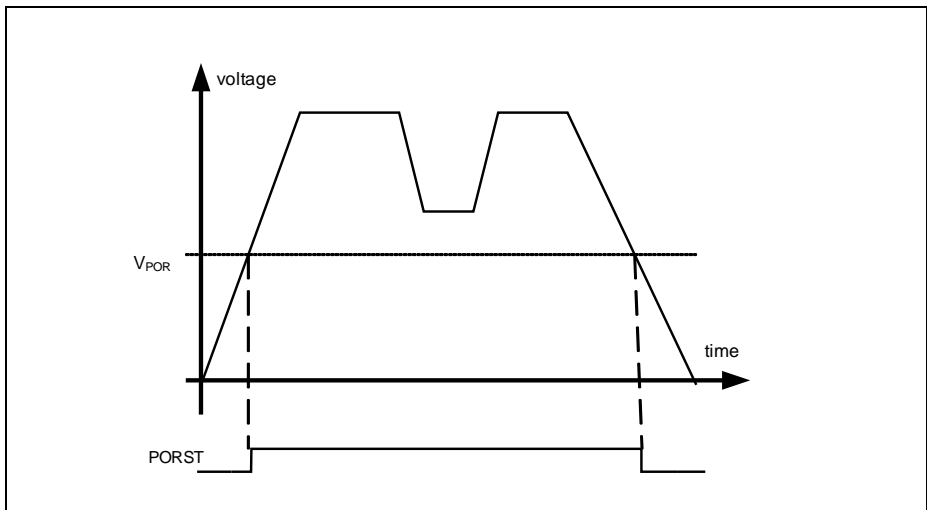


Figure 11-12 Supply Voltage Monitoring

While the supply voltage is below V_{POR} the device is held in reset. As soon as the voltage falls below V_{POR} a power on reset is triggered.

11.3.7 Power Validation

A power validation circuit monitors the internal core supply voltage of the core domain. It monitors that the core voltage is above the voltage threshold V_{PV} which guarantees save operation. Whenever the voltage falls below the threshold level a power-on reset is generated.

11.3.8 Supply Voltage Brown-out Detection

Brown-out detection is an additional voltage monitoring feature that enables the user software to perform some corrective action that bring the chip into safe operation in case a critical supply voltage drop and avoids System Reset generated by the Supply Voltage Monitoring.

System Control Unit (SCU)

A drop of supply voltage to a critical threshold level programmed by the user can be signaled to the CPU with an NMI. An emergency corrective action may involve e.g. reduction of current consumption by switching of some modules or some interaction with external devices that should result in recovery of the supply voltage level.

Automatic monitoring of the voltage against programmed voltage allows efficient operation without a need for software interaction if the supply voltage remains at a safe level. The supply voltage can be monitored also directly by software in register **EVRVADCSTAT**. The threshold value and the inspection interval is configured at **PWRMON**.

11.3.9 Hibernate Domain Power Management

It is strongly recommended to supply Hibernate domain with V_{DDP} when available in order to extend the battery life time. An example of an external supply voltage switching solution based on Schottky diodes is shown in **Figure 11-14** and **Figure 11-15**.

11.3.10 Flash Power Control

The Flash module can be configured to operate in low power mode while the system is in Deep Sleep mode. Control of the Flash power mode it is performed using the **DSLEEP** register prior to entering the Deep Sleep mode. The Flash cannot be accessed while in the lower mode. Upon a wake-up event the Flash gets automatically reactivated. The user needs to trade the reduced leakage current against wake-up time of the Flash.

11.4 Hibernate Control

Hibernate is the operation mode of lowest power consumption. Activity of the microcontroller is limited to real time keeping and wake up functionality.

11.4.1 Hibernate Mode

Hibernate control is performed with Hibernate Control Unit (HCU).

Externally Controlled Hibernate Mode

In this operation mode of the with lowest power consumption only the hibernate domain remains powered up. In this state the hibernate domain enable switching off externally the main power supply (see **Figure 11-14** and **Figure 11-15**).

External Voltage Regulator is controlled from Hibernate domain and V_{DDP} gets switched off when hibernate mode is entered. No reset of hibernate control occurs in hardware upon power-up but the chip will boot up as normal since in this mode no internal state of the chip is affected except for the hibernate control pin from hibernate domain to the External Voltage Regulator. Re-initialization of is possible with software upon boot-up.

Hibernation Support

The entry of the hibernate state is configured via the register **HDCR** by setting of the HIB bit. The HIBOUT bit in conjunction with selected HIBIONPOL bit of **HDCR** register drives HIB_IO_n pad.

The hibernation control signal HIBOUT is connected to an open-drain pad to enable control of an external power regulator for V_{DDP} .

Wake-up from Hibernate Mode

The hibernation control supports multiple wake-up sources. Occurrence of any of the Wake-up from hibernate triggers resets of **HDCR.HIB** bit which results resetting of the HIBOUT signal that controls External Voltage Regulator.

11.4.2 Hibernate Domain Pin Functions

Hibernate domain control register **HDCR** implements fields for alternate pin function selection.

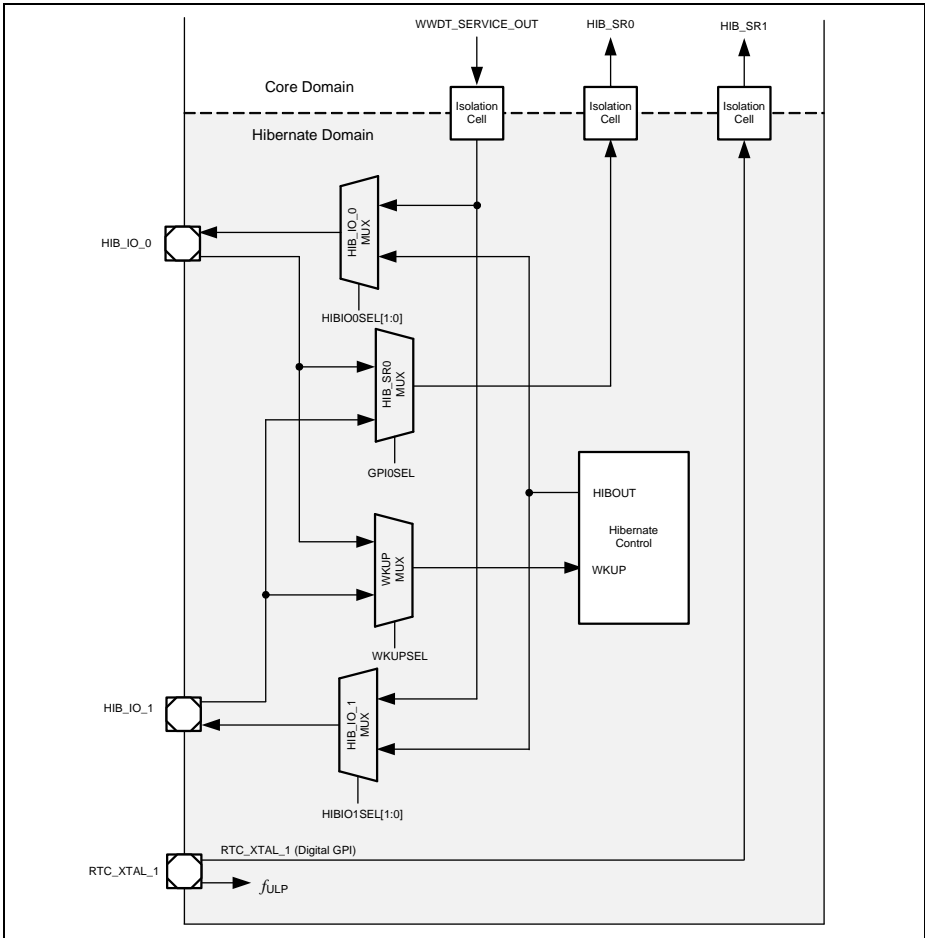


Figure 11-13 Alternate function selection of HIB_IO_0 and HIB_IO_1 pins of Hibernate Domain

11.4.3 System Level Integration

The XMC4300 enables various system level configuration options for Hibernate mode support, determined by application specific requirements. The examples shown below illustrate functional principles of the hibernate control mechanism in the system level integration aspect.

Externally Controlled Hibernate mode

The Externally Controlled Power Supply scheme require external devices on in order to fully support power management related functions.

The external power supply needs to support on/off control of the VDDP voltage generation. This scheme enables the XMC4300 e.g. to control supply voltage for multiple on-board devices.

A simple power control scheme assumes availability of two pins - HIB_IO_0 and HIB_IO_1 as shown in **Figure 11-14**. If the External Voltage regulator implements low-active (enabled with low) power control then it can be reliably controlled with HIB_IO_0 pin, which by default is set to open-drain and ensures proper driving value from the moment a valid VBAT voltage gets applied. A wake-up signal can be supplied externally via HIB_IO_1 pin or internally generated with from RTC module.

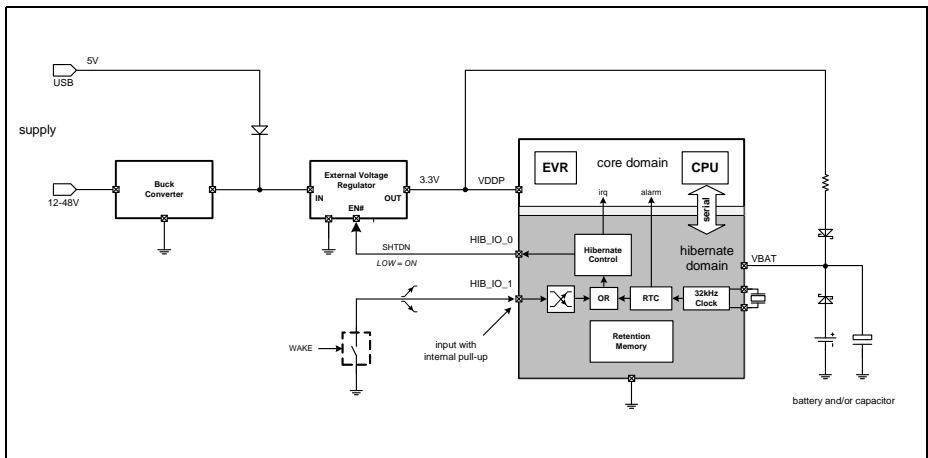


Figure 11-14 System Level Power Control example - externally controlled with two pins

The HIB_IO_0 pad is configured as open drain low and HIB_IO_1 as input after reset. This configuration enables use of low-enabled external voltage regulator as depicted in the **Figure 11-14**. In case of high-active external voltage regulator and battery presence he pins may be swapped to ensure reliable startup of the system start up of the system from the moment a valid VBAT voltage is applied (additional external pull-up required on HIB_IO_1).

In case of application cases with only one hibernate control pin available for external power control, it is possible to use the same pin for control of the External Voltage Regulator and for an external wake-up trigger signal. The example setup is shown in **Figure 11-15**, where HIB_IO_0 pin is by default configured to open-drain mode.

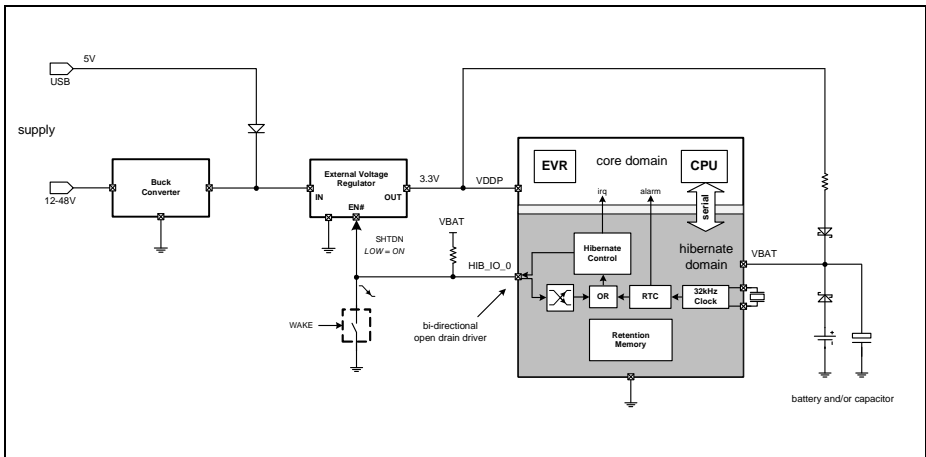


Figure 11-15 System Level Power Control example - externally controlled with single pin

After power up and before entering Hibernate mode the HIB_IO_0 needs to be reconfigured to bidirectional (but still open-drain driver) mode. Hibernate mode activation automatically changes state of the HIB_IO_0 output to high impedance which allow the external pull up resistor to drive high value of the External Voltage Regulator in order to switch off VDDP generation. A wake-up trigger signal from external source to the Hibernation Control logic will propagate via the bidirectional HIB_IO_0 pin. A wake-up trigger needs to be driven by an external open-drain device capable to overcome driving strength of the pull-up resistor.

Internally Controlled Hibernate Mode

The Internally Controlled Power Supply scheme assumes that VDDP generation isn't controlled by the XMC4300 and VDDC voltage is controlled with the on-chip Embedded Voltage Regulator (EVR) instead, as shown in [Figure 11-16](#). An External Voltage Regulator without on/off control may be used in this scenario.

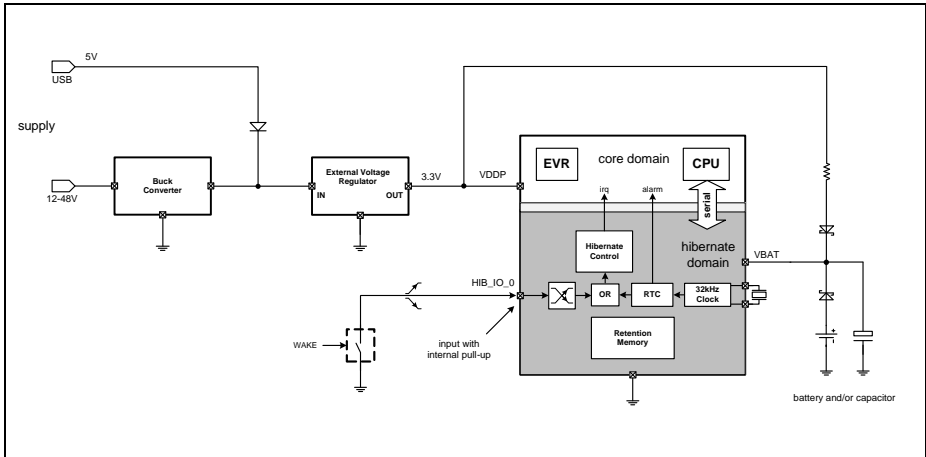


Figure 11-16 System Level Power Control example - internally controlled

The external switch based on a Schottky diode prevents discharging battery when voltage is supplied to the chip from the external power supply.

11.5 Reset Control

Reset Control Unit (RCU) performs control of all reset related functionality including:

- Reset assertion on various reset request sources
- Inspection of reset sources after reset
- Selective reset of peripherals

11.5.1 Supported Reset types

The XMC4300 implements the following reset signals:

- Power-on Reset, $\overline{\text{PORESET}}$
- System Reset, $\overline{\text{SYSRESET}}$
- Standby Reset, $\overline{\text{STDBYRESET}}$
- Debug Reset, $\overline{\text{DBGRESET}}$

Power-on Reset ($\overline{\text{PORESET}}$)

A complete reset of the core domain of the device is executed upon power-up. Whenever the supply V_{DDP} is ramped-up and crossing the PORST voltage threshold the power-on reset is released. Additional a power-on reset is triggered by asserting the external PORST pin.

System Control Unit (SCU)

A Power-on reset is asserted again whenever the V_{DDP} voltage or the V_{DDC} voltage falls below defined reset thresholds.

System Reset (SYSRESET)

System Reset is triggered by sources:

- Power-on reset
- Software reset via Cortex-M4 Application Interrupt and Reset Control Register (AIRCR)
- Lockup signal from Cortex-M4 when enabled at RCU
- Watchdog reset
- Memory Parity Error
- EtherCAT (ECAT) module triggered by an external master

The System Reset resets almost all logic in the core domain. The only exceptions in the core domain are RCU Registers and Debug Logic.

Standby Reset, (STDBYRESET)

The Hibernate domain including the RTC is only reset by a standby reset. A standby reset is triggered by a power-on reset specific to the Hibernate domain. Additionally a standby reset can be activated by software:

- Power-on reset specific to the Hibernate domain
- Software reset via **RSTSET** register

Debug Reset (DBGRESET)

Debug reset is triggered by the following sources:

- Debug Reset request from DAP while in mission mode and with debug probe present
- System Reset while in normal mode and debug probe not present

The Debug Reset is triggered by System Reset while in normal operation mode while debug probe is not present.

11.5.2 Peripheral Reset Control

Software can activate the reset of all peripherals individually via the registers **PRSET0**, **PRSET1**, and **PRSET2**. The default state is that all peripherals are in reset after power-up. A return to the default state of a peripheral can be performed by forcing it to reset state by a separate reset.

The user needs to properly configure the port values before resetting a module to assure that the default output values of a peripheral do not harm external circuitry. Similarly the user has to take care of top-level interconnects, which will not be affected by a module specific reset.

11.5.3 EtherCAT Reset

The ECAT module can generate reset request to the RCU. The reset request is also flagged in **TRAPRAW.ECAT0RST** register (see **Figure 11-17**).

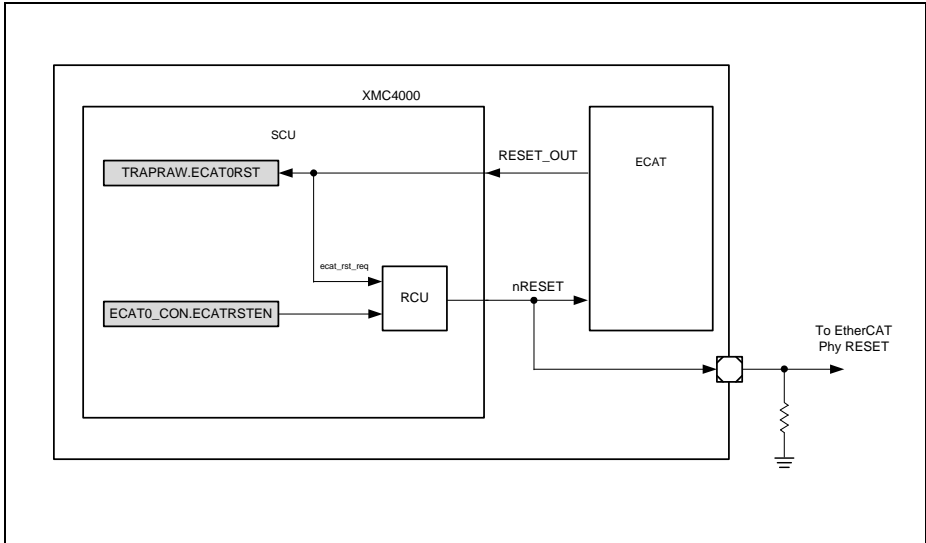


Figure 11-17 EtherCAT Reset Control

The reset request may trigger two alternative types of action, depending on user configuration:

- Hardware mechanism: complete system reset is generated automatically by RCU if reset is enabled with **ECAT0_CON.ECATRSTEN**.
- Software routine: reset request triggers system trap resulting in an NMI, flagged in **TRAPRAW.ECAT0RST** register. Software reset of the complete system, or, a part of the system can be performed in NMI service routine according to application specific requirements.

11.5.4 Reset Status

The EVR provides the cause of a power reset to the RCU. The reset cause can be inspected after resuming operation by reading register **RSTSTAT**.

All register of the RCU undergo reset only by a power-on reset **PORESET**.

Table 11-3 shows an overview of the reset signals their source and effects on the various parts of the system.

Table 11-3 Reset Overview

Name	Source	Core Domain	PAD Domain	Hibernate Domain	Debug & Trace System
PORESET	EVR	yes	yes	no	yes
SYSRESET	PORESET Software WDT	yes	no	no	no
DBGRESET	DAP Software	no	no	no	yes

11.6 Clock Control

Clock generation and control is performed in the Clock Control Unit (CCU).

11.6.1 Block Diagram

The Clock Control Unit (CCU) consists of two major sub blocks:

- Clock Generation Unit (CGU)
- Clock Selection Unit (CSU)

The CGU provides in parallel three clocks to the CSU:

- USB PLL clock f_{PLLUSB} ,
- System PLL output clock f_{PLL}
- internally generated clock f_{OFI} from the Backup Clock Source.

The f_{STDBY} clock of 32.768 kHz is generated in the Standby Clock Generation Unit (SCGU) of the hibernate domain by either the external crystal oscillator OSC_ULP or by Internal Slow Clock Source.

The module clocks available in the Core Domain get derived from the CGU, passed via CSU module which implements a number of multiplexers and clock dividers enabling clock selection for all system modules and scaling of the frequencies.

The CSU receives in addition a slow standby clock f_{STDBY} from the hibernate domain that can be used in the WDT module for safety sensitive application purpose.

Major clock sources can be selected for driving the external clock pin EXTCLK.

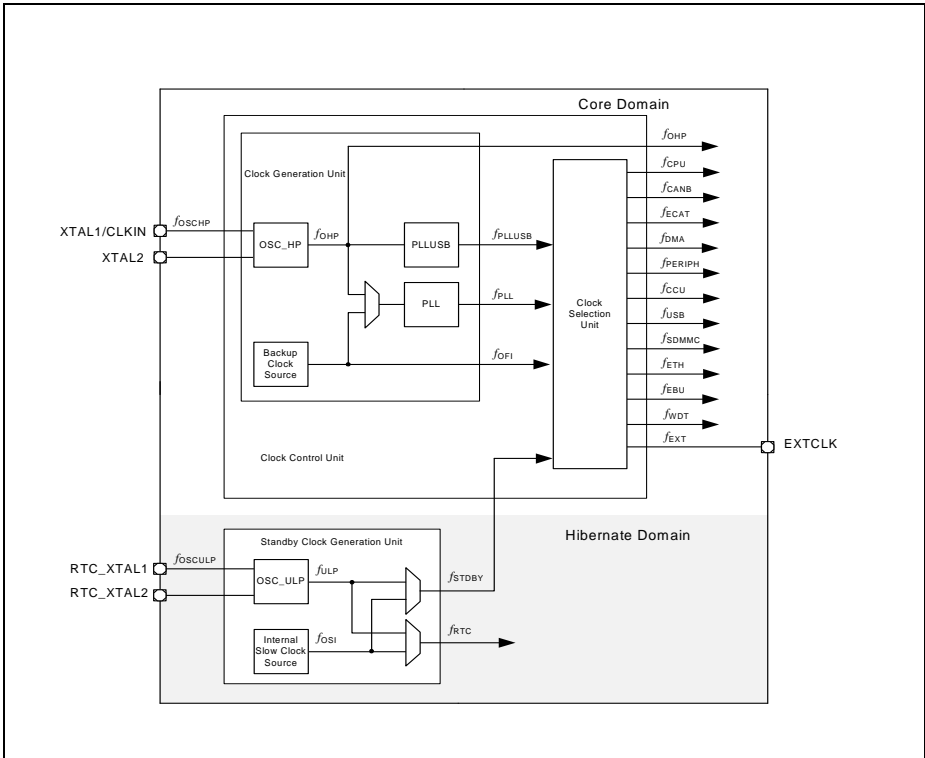


Figure 11-18 Clock Control Unit

11.6.2 Clock Sources

The system has multiple clock sources distributed over the core power domain and the hibernate power domain.

The source clock for CGU can be supplied from:

- internally generate in the Backup Clock Source
- external clock source directly via CLKIN pin
- external crystal High Precision Oscillator (OSC_HP) utilizing XTAL1 and XTAL2 pins

During system start-up the backup clock f_{OFI} of is supplied to the system. The f_{OFI} clock may be used during normal or low power operation after the startup.

The high precision oscillator OSC_HP can be used with an external crystal to generate high precision clock via XTAL1 and XTAL2 pins. Either, the OSC_HP or Backup Clock Source can be selected as input clock source for the main PLL. Alternatively, an external

System Control Unit (SCU)

clock can also be supplied directly to CGU from CLKIN pin, via the OSC_HP module, bypassing the oscillator circuit.

The clock sources in the hibernate power domain are:

- OSC_ULP, an ultra low power external crystal oscillator 32.768 kHz clock f_{ULP}
- Internal Slow Clock source 32.768 kHz clock f_{OSI}

The clocks drive the logic in the hibernate domain. In addition one of the two can be used as standby clock for low power operation in the core power domain.

To generate the required high precision f_{PLLUSB} clock for operation of the USB and the MMC/SD modules additional PLL can be optionally used if the desired clock frequency cannot be derived from the system PLL output f_{PLL} . The dedicated PLL referred to as PLLUSB, shown the block diagram in **Figure 11-19**.

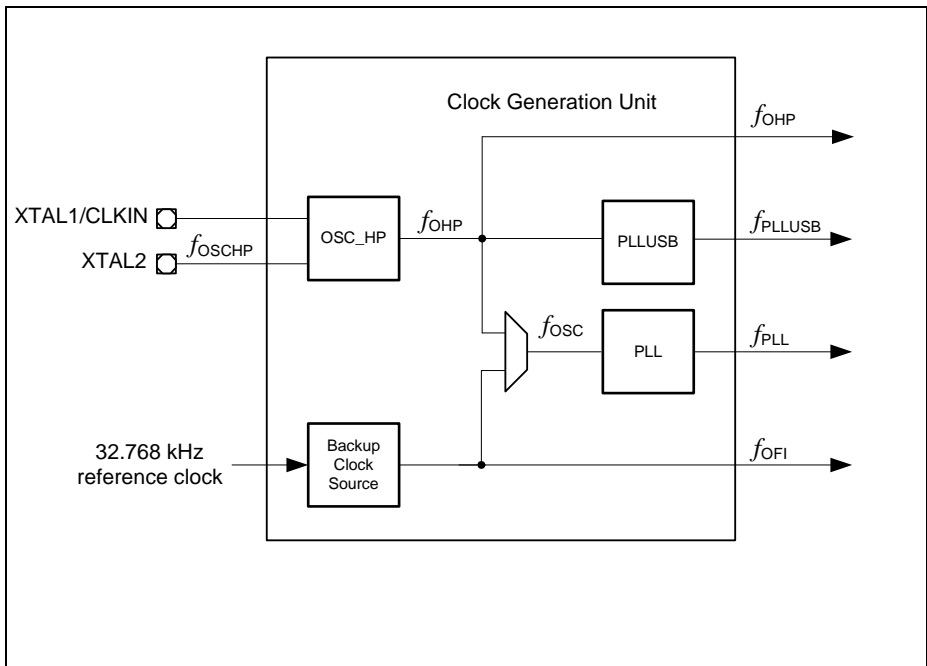


Figure 11-19 Clock Generation Block Diagram

11.6.3 Clock System Overview

The clock selection unit CSU provides the following clocks to the system:

Table 11-4 Clock Signals

Clock name	From/to module or pin	Description
System Clock Signals		
f_{SYS}	SCU.CCU	System master clock
f_{CPU}	CPU	CPU and NVIC clock
f_{DMA}	DMA0	DMA clock
f_{PERIPH}	PBA0 PBA1	Peripheral clock for modules connected to peripheral bridges PBA0 and PBA1
f_{CCU}	CCU4 CCU8 POSIF	Clock for CCU4, CCU8 and POSIF modules
f_{USB}	USB	UTMI clock of USB
f_{WDT}	WDT	Clock for independent watchdog timer
Internal Clock Signals		
f_{PLL}	System PLL	System PLL output clock
f_{PLLUSB}	USB PLL	USB PLL output clock
f_{OHP}	OSC_HP	External crystal oscillator output clock
f_{OFI}	Internal Backup Clock Source	System Backup Block
f_{ULP}	OSC_ULP	External crystal slow oscillator output clock
f_{OSI}	Slow Internal Backup Clock Source	32.768 kHz backup clock for Hibernate domain
f_{OSC}	PLL input	PLL input clock
f_{STDBY}	Hibernate Domain	32.768 kHz clock, optional WDT independent clock
f_{FLASH}	FLASH	Internal Flash clock
External Clock Signals		
f_{OSCHP}	XTAL1/CLKIN	External crystal input, optionally also direct input clock to system PLL

Table 11-4 Clock Signals (cont'd)

Clock name	From/to module or pin	Description
f_{OSCULP}	RTC_XTAL1	External crystal input, optionally also direct input clock to Hibernate domain and RTC module
f_{EXT}	Clock output to pin EXTCLK	Chip output clock

The clock sources of the clock selection unit are the four clocks from the clock generation unit, i.e. the USB clock f_{USB} , the PLL output clocks f_{PLL} and a fast internal generated clock f_{OFI} . The clock selection unit receives as additional clocks source a slow standby clock from the hibernate domain.

11.6.3.1 Clock System Architecture

The system clock f_{SYS} drives the CPU clock and all peripheral modules. It can be selected from the following clock sources:

- f_{PLL} , main PLL output clock
- f_{OFI} , fast internal clock, bypassing PLL
- f_{OSCHP} , external clock, bypassing PLL
- f_{OHP} , external crystal oscillator clock, bypassing PLL

Please note that in dependence of the PLL mode setting the PLL output clock f_{PLL} is either a scaled version of the VCO clock in normal mode or a scaled version of one of the input clocks.

In Deep Sleep mode the system clocks can be switched to a clock which does not require PLL and thereby allowing the power down of the system PLL. This is either the fast internal clock or the slow standby clock. The **DSLEEP** register controls the clock settings for Deep Sleep mode.

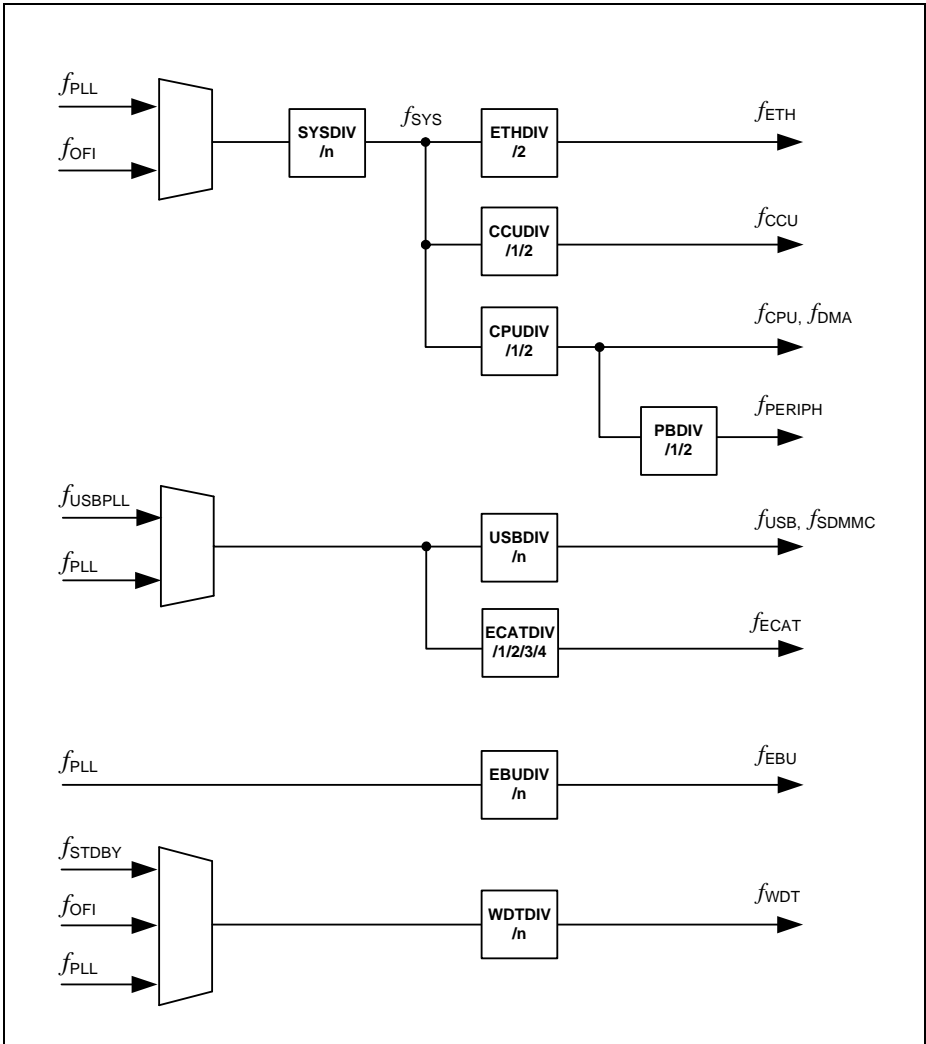


Figure 11-20 Clock Selection Unit

Some limitations apply on clock ratio combinations between f_{CCU} , f_{CPU} and f_{PERIPH} . Only divider setting listed in the table [Table 11-5](#) are allowed for the f_{CCU} , f_{CPU} and f_{PERIPH} clocks. All other clock dividers settings must be prohibited by application software or by hardware when programmed with **MLINKCLKCR** register in order to avoid invalid clock

System Control Unit (SCU)

ratios leading to system malfunctions. The **Table 11-5** table shows also example clock rate values assuming f_{SYS} rate of 144 MHz.

Table 11-5 Valid values of clock divide registers for f_{CCU} , f_{CPU} and f_{PERIPH} clocks

CCUCLKCR.CCUDIV	CPUCLKCR.CPUDIV	PBCLKCR.PBDIV
0 (144 MHz)	0 (144MHz)	0 (144MHz)
0 (144MHz)	0 (144MHz)	1 (72 MHz)
0 (144MHz)	1 (72MHz)	0 (72MHz)
1 (72MHz)	0 (MHz)	1 (72MHz)
1 (72MHz)	1 (72MHz)	0 (72 MHz)

USB and MMC/SD Clock Selection

The clock for the USB module and the clock for the MMC/SD module are both derived from the clock f_{USB} or from the clock f_{PLL} of the main PLL. The later is only feasible, when the main PLL is providing a frequency which allows to generate the 48MHz with a 3-bit divider.

The USB clock is automatically gated and the USB PLL put in power-down by the USB suspend signal.

The clock divider and the MUX must only be configured while the USB and MMC/SD are not enabled.

Ethernet Clock Selection

The Ethernet module receives SRAM clock f_{SYS} which has to be twice the frequency of the ETH MAC internal clock f_{ETH} . The SRAM clock rate must always be above 100 MHz which guarantee that the clock of ETH MAC is of rate above its acceptable minimum of 50 MHz.

Both clocks are disabled with software in Wake-On-Lan mode. The clocks have to be enabled via software when the related interrupt the end of Wake-On-Lan mode.

The clock divider and the clock MUX must only be configured while ETH module is not enabled.

CPU Clock Selection

The CPU clock f_{CPU} may be equal to or a half of the system clock f_{SYS} .

Peripheral Bus Clock Selection

The Peripheral Bus clock is derived from the f_{CPU} clock and may be equal to or a half of the f_{CPU} . There are some limitations imposed on peripheral bus configuration in terms of clock ratio with respect to the f_{CCU} clock. The Peripheral Bus clock rate must never be less than half and more than the f_{CCU} clock.

CAPCOM and POSIF Clock Selection

The capture compare blocks CCU4, CCU8 and POSIF use as timer clock the clock f_{CCU} derived from system f_{SYS} via CCUDIV clock divider. The clock divider allows to adjust clock frequency of the timers with respect to the rest of the system. Relation between f_{CCU} clock and other clocks in the system is constrained as described in the [Table 11-5](#). The f_{CCU} clock frequency must only be configured while all CAPCOM and POSIF modules are not enabled.

Watchdog Clock Selection

The watchdog module uses as independent clock either the internal fast clock f_{OFI} or the slow clock f_{STDBY} . The system clock f_{PLL} is available as additional clock source.

Both the clock divider and the clock MUX must only be configured while the WDT is not enabled.

External Clock Output

An external clock is provided via clock pin EXTCLK. All clock sources can be selected as clock signal for the external clock output. Optionally a divider can be used before bringing the system clock to the outside to stay within the limit of the supported frequencies for the pads.

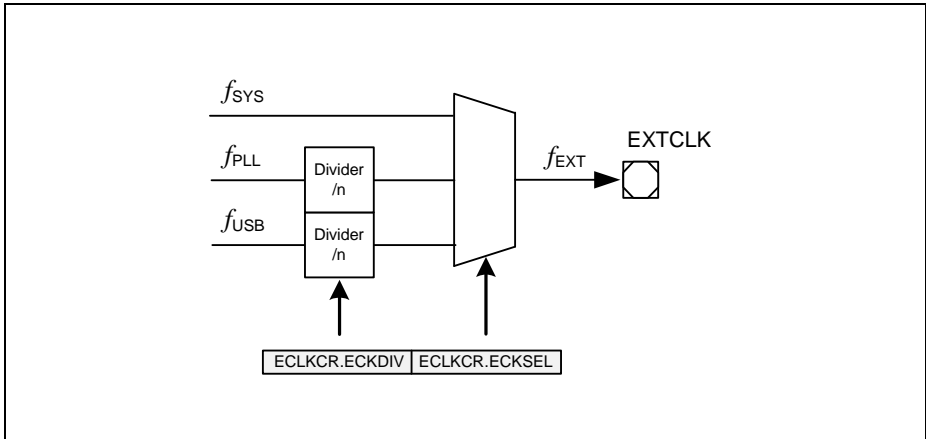


Figure 11-21 External Clock Selection

11.6.4 High Precision Oscillator Circuit (OSC_HP)

The high precision oscillator circuit can drive an external crystal or accepts an external clock source. It consists of an inverting amplifier with XTAL1 as input, and XTAL2 as output.

Figure 11-23 and **Figure 11-22** show the recommended external circuitries for both operating modes, External Crystal Mode and External Input Clock Mode.

External Input Clock Mode

In this usage an external clock signal is supplied directly not using an external crystal and bypassing the amplifier of the oscillator. The input frequency must be in the range from 4 to 40 MHz. When using an external clock signal it must be connected to XTAL1. XTAL2 is left open i.e. unconnected.

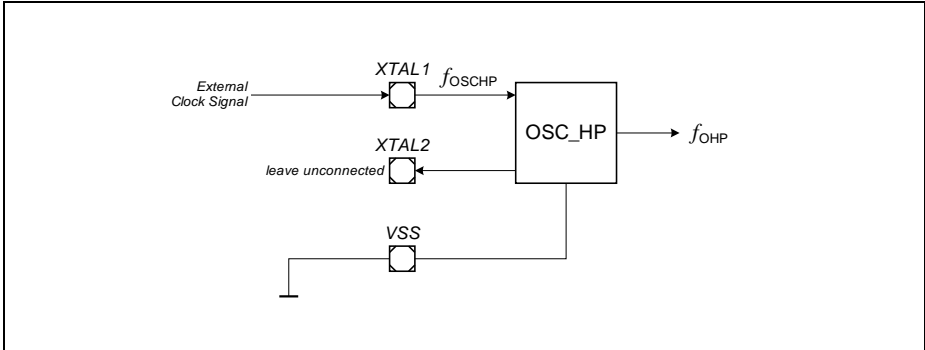


Figure 11-22 External Clock Input Mode for the High-Precision Oscillator

External Crystal Mode

For the external crystal mode an external oscillator load circuitry is required. The circuitry must be connected to both pins, XTAL1 and XTAL2. It consists normally of the two load capacitances C1 and C2. For some crystals a series damping resistor might be necessary. The exact values and related operating range depend on the crystal and have to be determined and optimized together with the crystal vendor using the negative resistance method.

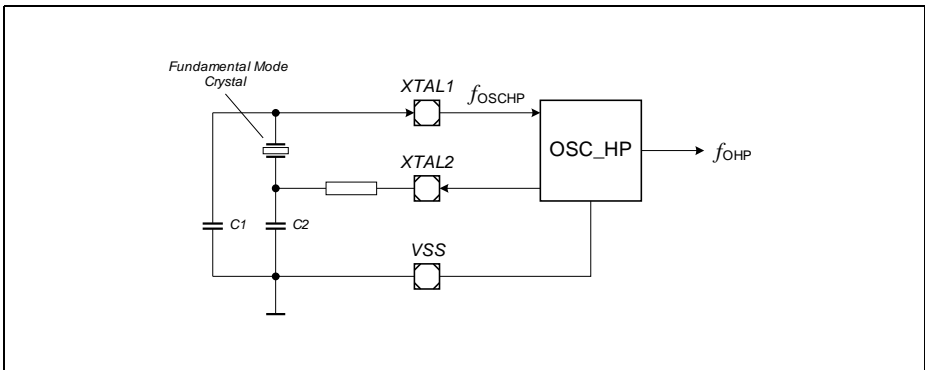


Figure 11-23 External Crystal Mode Circuitry for the High-Precision Oscillator

11.6.5 Backup Clock Source

The backup clock f_{OFI} generated internally is the default clock after start-up. It is used for bypassing the PLL for startup of the system without external clock. Furthermore it can be used as independent clock source for the watchdog module or even as system clock

System Control Unit (SCU)

source during normal operation. While in prescaler mode this clock is automatically used as emergency clock if the external clock failure has been detected.

Clock adjustment is required to reach desired level of f_{OFI} precision. The backup clock source provides two adjustment procedures:

- loading of adjustment value during start-up
- continuous adjustment using the high-precision f_{STDBY} clock as reference

11.6.6 Main PLL

The main PLL converts a low-frequency external clock signal to a high-speed internal clock. The PLL also has fail-safe logic that detects degenerative external clock behavior such as abnormal frequency deviations or a total loss of the external clock. The PLL triggers autonomously emergency action if it loses its lock on the external clock and switches to the Fast Internal Backup Clock.

This module is a phase locked loop for integer frequency synthesis. It allows the use of input and output frequencies of a wide range by varying the different divider factors.

11.6.6.1 Features

- VCO lock detection
- 4-bit input divider **P**: (divide by PDIV+1)
- 7-bit feedback divider **N**: (multiply by NDIV+1)
- 7-bit output dividers **K1 or K2**: (divide by KxDIV+1)
- Oscillator Watchdog
 - Detecting too low input frequencies
 - Detecting too high input frequencies
 - Spike detection for the OSC input frequency
- Different operating modes
 - Bypass Mode
 - Prescaler Mode
 - Normal Mode
- VCO Power Down
- PLL Power Down
- Glitch less switching between K-Dividers
- Switching between Normal Mode and Prescaler Mode

11.6.6.2 System PLL Functional Description

The PLL consists of a Voltage Controlled Oscillator (VCO) with a feedback path. A divider in the feedback path (N-Divider) divides the VCO frequency down. The resulting frequency is then compared with the externally provided and divided frequency (P-Divider). The phase detection logic determines the difference between the two clocks

and accordingly controls the frequency of the VCO (f_{VCO}). A PLL lock detection unit monitors and signals this condition. The phase detection logic continues to monitor the two clocks and adjusts the VCO clock if required.

The following figure shows the PLL block structure.

Clock Source Control

The input clock for the PLL f_{OSC} can be one of the following two clock sources:

- The internal generated fast clock f_{OFI}
- The clock f_{OHP} sourced by the crystal oscillator OSC_HP

The PLL clock f_{PLL} is generated from the input clock in one of two software selectable operation modes:

- Normal Mode, using VCO output clock
- Prescaler Mode, using input clock directly

The PLL output clock f_{PLL} is derived from either

- VCO clock divided by the K2-Divider, Normal Mode
- external oscillator clock divided by the K1-Divider, Prescaler Mode
- backup clock divided by the K1-Divider, Prescaler Mode

The PLL clock f_{PLL} is generated in emergency from one of two sources:

- Free running VCO if Emergency entered from Normal Mode of PLL
- Backup Clock if emergency entered from Prescaler Mode of PLL

Configuration and Operation of the Normal Mode

In Normal Mode, the PLL is running at the frequency f_{OSC} and f_{PLL} is divided down by a factor P, multiplied by a factor N and then divided down by a factor K2.

The output frequency is given by:

(11.1)

$$f_{PLL} = \frac{N}{P \cdot K_2} \cdot f_{OSC}$$

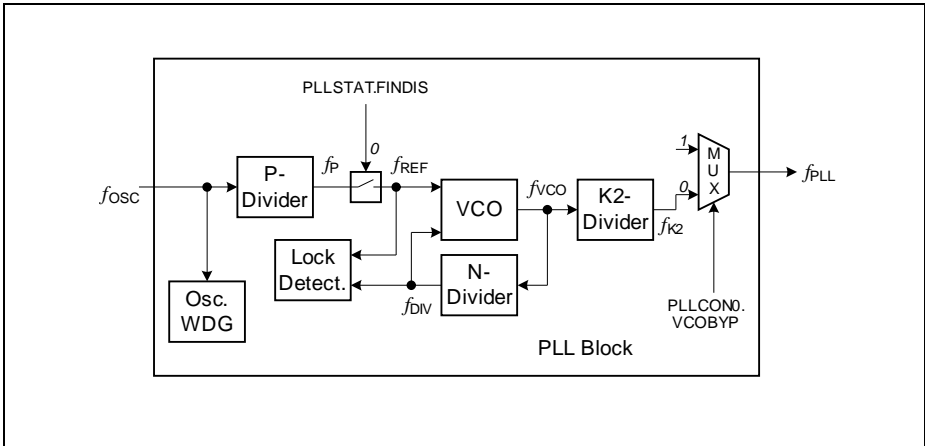


Figure 11-24 PLL Normal Mode

It is strongly recommended to apply even values of P and K2 Divider parameters in order to minimize PLL output clock jitter effect. Please find PLL configuration examples values in [Table 11-6](#).

Table 11-6 PLL example configuration values

Target Frequency of f_{PLL} [MHz]	External Crystal Frequency [MHz]	P Parameter	N Parameter	K2 Parameter
80	4	1	80	4
	8	1	40	4
	12	2	80	6
	16	1	20	4

Note: The values of P, N and K2 configuration parameters specified in [Table 11-6](#) should must decremented by one before programmed into corresponding registers.

The Normal Mode is selected by the following settings

- Register Setting
 - PLLCON0.VCOBYP = 0
 - PLLCON0.FINDIS = 0

The Normal Mode is entered when the following requirements are all together valid:

- Register Values
 - PLLCON0.FINDIS = 0

- PLLSTAT.VCOBYST = 0
- PLLSTAT.VCOLOCK = 1
- PLLSTAT.PLLLV = 1
- PLLSTAT.PLLLH = 1

Operation on the Normal Mode does require an input clock frequency of f_{OSC} . Therefore it is recommended to check and monitor if an input frequency f_{OSC} is available at all by checking PLLSTAT.PLLLV. For a better monitoring also the upper frequency can be monitored via PLLSTAT.PLLHV.

For the Normal Mode there is the following requirement regarding the frequency of f_{OSC} . A modification of the two dividers P and N has a direct influence to the VCO frequency and lead to a loss of the VCO Lock status. A modification of the K2-divider has no impact on the VCO Lock status.

When the frequency of the Normal Mode should be modified or entered the following sequence needs to be followed:

- Configure and enter Prescaler Mode
- Disable NMI trap generation for the VCO Lock
- Configure Normal Mode
- Wait for a positive VCO Lock status (PLLSTAT.VCOLOCK = 1).
- Switch to Normal Mode by clearing PLLCON.VCOBYP

The Normal Mode is entered when the status bit PLLSTAT.VCOBYST is cleared.

When the Normal Mode is entered, the NMI status flag for the VCO Lock trap should be cleared and then enabled again. The intended PLL output target frequency can now be configured by changing only the K2-Divider. This can result in multiple changes of the K2-Divider to avoid to big frequency changes. Between the update of two K2-Divider values 6 cycles of f_{PLL} should be waited. For ramping up PLL output frequency in Normal Mode the following steps are required:

- The first target frequency of the Normal Mode should be selected in a way that it matches or is only slightly higher as the one used in the Prescaler Mode. This avoids big changes in the system operation frequency and therefore power consumption when switching later from Prescaler Mode to Normal Mode.
- Selecting P and N in a way that f_{VCO} is in the upper area of its allowed values leads to a slightly increased power consumption but to a slightly reduced jitter
- Selecting P and N in a way that f_{VCO} is in the lower area of its allowed values leads to a slightly reduced power consumption but to a slightly increased jitter
- It is recommended to reset the f_{VCO} Lock detection (PLLCON0.RESLD = 1) after the new values of the dividers are configured to get a defined VCO lock check time.

Depending on the selected divider value of the K2-Divider the duty cycle of the clock is selected. This can have an impact for the operation with an external communication interface. The duty cycles values for the different K2-divider values are defined in the Data Sheet.

PLL VCO Lock Detection

The PLL has a lock detection that supervises the VCO part of the PLL in order to differentiate between stable and unstable VCO circuit behavior. The lock detector marks the VCO circuit and therefore the output f_{VCO} of the VCO as instable if the two inputs f_{REF} and f_{DIV} differ too much. Changes in one or both input frequencies below a level are not marked by a loss of lock because the VCO can handle such small changes without any problem for the system.

PLL VCO Loss-of-Lock Event

The PLL may become unlocked, caused by a break of the crystal or the external clock line. In such a case, an NMI trap is generated if it were enabled. Additionally, the OSC clock input f_{OSC} is disconnected from the PLL VCO to avoid unstable operation due to noise or sporadic clock pulses coming from the oscillator circuit. Without a clock input f_{OSC} , the PLL gradually slows down to its VCO base frequency and remains there. This default feature can be disabled by setting bit `PLLCON0.OSCDISCDIS`. If this bit is set the OSC clock remains connected to the VCO.

11.6.6.3 Configuration and Operation of the Prescaler Mode

In Prescaler Mode, the PLL is running at the external frequency f_{OSC} and f_{PLL1} is derived from f_{OSC} only by the K1-Divider.

The output frequency is given by

$$f_{PLL} = \frac{f_{OSC}}{K_1} \quad (11.2)$$

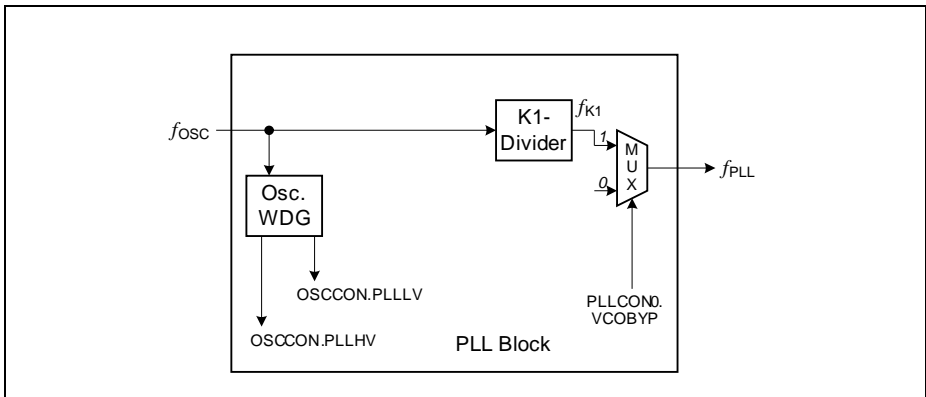


Figure 11-25 PLL Prescaler Mode Diagram

The Prescaler Mode is selected by the following settings

- PLLCON0.VCOBYP = 1

The Prescaler Mode is entered when the following requirements are all together valid:

- PLLSTAT.VCOBYST = 1
- PLLSAT.PLLLV = 1

Operation on the Prescaler Mode does require an input clock frequency of f_{osc} . Therefore it is recommended to check and monitor if an input frequency f_{osc} is available at all by checking PLLSAT.PLLLV. For a better monitoring also the upper frequency can be monitored via PLLSAT.PLLHV.

For the Prescaler Mode there are no requirements regarding the frequency of f_{osc} .

The system operation frequency is controlled in the Prescaler Mode by the value of the K1-Divider. When the value of PLLCON1.DIV was changed the next update of this value should not be done before bit PLLSTAT.K1RDY is set.

Depending on the selected divider value of the K1-Divider the duty cycle of the clock is selected. This can have an impact for the operation with an external communication interface. The duty cycles values for the different K1-divider values are defined in the Data Sheet.

The Prescaler Mode is requested from the Normal Mode by setting bit PLLCON.VCOBYP. The Prescaler Mode is entered when the status bit PLLSTAT.VCOBYST is set. Before the Prescaler Mode is requested the K1-Divider should be configured with a value generating a PLL output frequency f_{PLL} that matches the one generated by the Normal Mode as much as possible. In this way the frequency change resulting out of the mode change is reduced to a minimum.

The Prescaler Mode is requested to be left by clearing bit PLLCON.VCOBYP. The Prescaler Mode is left when the status bit PLLSTAT.VCOBYST is cleared.

11.6.6.4 Bypass Mode

The bypass mode is used only for testing purposes. In Bypass Mode the input clock f_{OSC} is directly connected to the PLL output f_{PLL} .

The output frequency is given by:

(11.3)

$$f_{PLL} = f_{OSC}$$

11.6.6.5 System Oscillator Watchdog (OSC_WDG)

The oscillator watchdog monitors the incoming clock frequency f_{OSC} from the OSC_HP oscillator. A stable and defined input frequency is a mandatory requirement for operation in both Prescaler Mode and Normal Mode. In addition for the Normal Mode it is required that the input frequency f_{OSC} is in a certain frequency range to obtain a stable master clock from the VCO part.

The expected input frequency is selected via the bit field **OSCHPCTRL.OSCVAL**. The OSC_WDG checks for spikes, too low frequencies, and for too high frequencies.

The frequency that is monitored is f_{OSCREF} which is derived from f_{OSC} .

(11.4)

$$f_{OSCREF} = \frac{f_{OSC}}{OSCVAL + 1}$$

The divider value **OSCHPCTRL.OSCVAL** has to be selected in a way that f_{OSCREF} is 2.5 MHz.

Note: f_{OSCREF} has to be within the range of 2 MHz to 3 MHz and should be as close as possible to 2.5 MHz.

The monitored frequency is too low if it is below 1.25 MHz and too high if it is above 7.5 MHz. This leads to the following two conditions:

- Too low: $f_{OSCREF} < 1.25 \text{ MHz} \times (\text{OSCHPCTRL.OSCVAL} + 1)$
- Too high: $f_{OSCREF} > 7.5 \text{ MHz} \times (\text{OSCHPCTRL.OSCVAL} + 1)$

Before configuring the OSC_WDG function all the trap options should be disabled in order to avoid unintended traps. Thereafter the value of **OSCHPCTRL.OSCVAL** can be changed. Then the OSC_WDG should be reset by setting **PLLCON0.OSCVAL**. This requests the start of OSC_WDG monitoring with the new configuration. When the expected positive monitoring results of **PLLSTAT.PLLLV** and / or **PLLSTAT.PLLHV** are set the input frequency is within the expected range. As setting **PLLCON0.OSCVAL**

clears all three bits **PLLSTAT.PLLSP**, **PLLSTAT.PLLLV**, and **PLLSTAT.PLLHV** all three trap status flags will be set. Therefore all three flags should be cleared before the trap generation is enabled again. The trap disabling-clearing-enabling sequence should also be used if only bit **PLLCON0.OSCVAL** is set without any modification of **OSCHPCTRL.OSCVAL**.

11.6.6.6 VCO Power Down Mode

The PLL offers a VCO Power Down Mode. This mode can be entered to save power within the PLL. The VCO Power Down Mode is entered by setting bit **PLLCON0.VCOPWD**. While the PLL is in VCO Power Down Mode only the Prescaler Mode is operable. Please note that selecting the VCO Power Down Mode does not automatically switch to the Prescaler Mode. So before the VCO Power Down Mode is entered the Prescaler Mode must be active.

11.6.6.7 PLL Power Down Mode

The PLL offers a Power Down Mode. This mode can be entered to save power if the PLL is not required. The Power Down Mode is entered by setting bit **PLLCON0.PLLPWD**. While the PLL is in Power Down Mode no PLL output frequency is generated.

11.6.6.8 Application Hints

The PLL configuration is a complex module and require a proper configuration sequence. The following application hints are intended to avoid unexpected effects and keep the system in a stable operation state.

Switching from Normal Mode to Prescaler Mode

Switching from Normal Mode to Prescaler Mode can result in generating spikes on the PLL clock output. For setting the Prescaler Mode the following sequence should be applied:

1. Switch from PLL clock (normal mode) to OFI clock using **SYCLKCR** register
2. Set the bit **PLLCON0.VCOBYP** to change from normal to prescaler mode
3. Switch back from OFI clock to PLL clock (prescaler mode) using **SYCLKCR** register

Switching from Prescaler Mode to Normal Mode

Switching from Prescaler Mode to Normal Mode can result in generating spikes on the PLL clock output. For setting the Normal Mode the following sequence should be applied:

1. Switch from OFI clock (prescaler mode) to PLL clock using **SYCLKCR** register
2. Set the bit **PLLCON0.VCOBYP** to change from prescaler to normal mode

3. Switch back from PLL clock to OFI clock (normal mode) using **SYSCLKCR** register

11.6.7 Internally Generated System Clock Calibration

The internal Backup Clock Source by default generates output clock signal of a frequency parameters as specified in the Data Sheet, referred to as uncalibrated OSC_FI clock. In order to improve accuracy of the Backup Clock Source clock calibration mechanisms are available. For details please refer to the following sub-chapters.

11.6.7.1 Factory Calibration

The Factory Calibration mechanism allows the Backup Clock Source clock output adjustment to parameters specified in the Data Sheet, referred to as OSC_FI factory calibrated clock. The factory calibration can be enabled with user software via bit field FOTR of the **PLLCON0** register. Enabling of this calibration has an immediate effect on the clock output.

11.6.7.2 Automatic Calibration

The Automatic Calibration mechanism enables Backup Clock Source clock output continuous adjustment based on the relation to a high precision reference clock f_{STDBY} . This calibration brings the clock output parameters to the level specified in the Data Sheet as the OSC_FI automatically calibrated clock. The automatic calibration can be enabled with user software via bit field AOTREN of the **PLLCON0** register. It is required that the reference clock f_{STDBY} generated in the Hibernate domain is activated prior to enabling this method of clock calibration.

11.6.7.3 Alternative Internal Clock Calibration

An alternative system clock calibration can be performed by programming of the system PLL with register values reflecting individual chip calibration characteristics determined by the **CLKCALCONST.CALIBCONST** register value. This way of calibration allows to achieve clock of frequency variation comparable with the factory calibration enabled Internal Backup Clock Source, but with significantly lower output clock jitter as defined for f_{OFI} untrimmed oscillator. For more details please refer to Data Sheet.

The formula describing dependencies between calibration constant and PLL configuration settings that allow to program desired output frequency is provided in **Equation (11.5)**

(11.5)

$$f_{\text{PLL}} = 24 \times \left(\frac{\text{CALIBCONST} \times 488 + 11651}{11146} \right) \times \left(\frac{N}{P \times K2} \right) [\text{MHz}]$$

System Control Unit (SCU)

It is strongly recommended to apply even value of P and K2 parameters in order to minimize PLL output clock jitter effect. Please find PLL configuration examples for different calibration constant CALIBCONST values in [Table 11-7](#).

Table 11-7 PLL example configuration values

Target Frequency of f_{PLL} [MHz]	TRIM Constant	P Parameter	N Parameter	K2 Parameter
80	0	6	77	4
	1	6	74	4
	2	6	71	4
	3	6	69	4
	4	6	66	4
	5	6	64	4
	6	6	62	4
	7	6	60	4
	8	6	58	4
	9	6	56	4
	10	6	54	4
	11	6	53	4
	12	6	51	4
	13	6	50	4
	14	8	97	6
15	8	63	4	

Note: The values of P, N and K2 configuration parameters specified in [Table 11-7](#) should must decremented by one before programmed into corresponding registers.

11.6.8 USB PLL

The USB PLL serves the special purpose to provide an accurate clock for USB 2.0 full-speed operation. The basic functionality of the USB PLL is similar to the main PLL (see [Figure 11-26](#)).

Configuration and Operation

The PLLUSB is running at the frequency f_{OHP} and f_{PLLUSB} is divided down by a factor P, multiplied by a factor N and then divided down by the factor of 2.

The output frequency is given by:

(11.6)

$$f_{\text{PLLUSB}} = \frac{N}{P \cdot 2} \cdot f_{\text{OHP}}$$

Operation of the PLLUSB require an input clock frequency of f_{OHP} .

The following requirement must be fulfilled regarding the frequency of f_{OHP} (see).

A modification of the two dividers P and N has a direct influence to the VCO frequency and lead to a loss of the VCO Lock status.

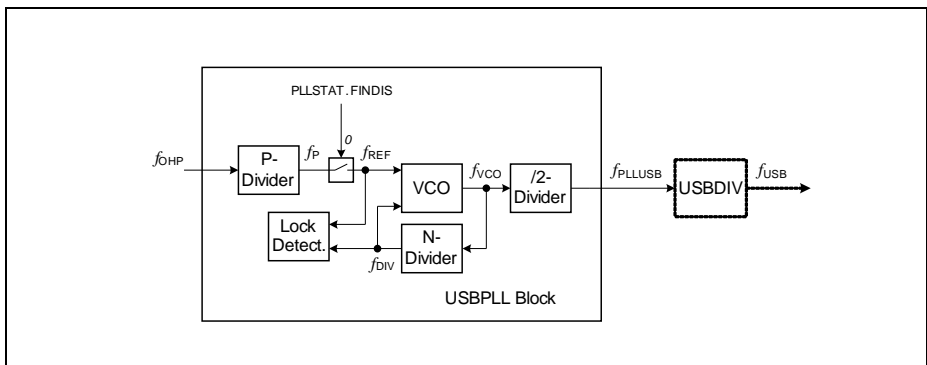


Figure 11-26 PLLUSB Block Diagram

It is strongly recommended to apply even value of P and USB Clock Divider (referred to as USBDIV divider) parameters in order to minimize the PLL output clock jitter effect. Please find PLLUSB configuration examples values in [Table 11-8](#).

Table 11-8 PLLUSB example configuration values

Crystal frequency [MHz]	P Parameter	N Parameter	USB Divider Parameter
8	2	96	4
12	2	64	4
16	2	48	4

Note: The values of P, N and USB Divider configuration parameters specified in [Table 11-8](#) should must decremented by one before programmed into corresponding registers.

System Control Unit (SCU)

The USB PLL is put automatically in power-down by the USB suspend signal, when the clock is not used for SD/MMC operation.

Note: Re-configuration of the P-Divider before the USBPLL has locked must be avoided.

11.6.9 Ultra Low Power Oscillator

The ultra low power oscillator is providing a real time clock source of 32.768 kHz when paired with an external crystal. It operates in the supply voltage range of the Hibernate power domain. The crystal pads are always powered by V_{BAT} or V_{DDP} .

The precise and stable clock can be propagated to Core domain as f_{STDBY} and used for continuous adjustments the fast internal backup clock source f_{OFI} .

11.6.9.1 OSC_ULP Oscillator Watchdog (ULPWDG)

The slow oscillator watchdog monitors the incoming clock frequency f_{ULP} from OSC_ULP. A reliable clock is required in the Hibernate domain in Hibernate state in order to perform system wake-up upon occurrence of configured events. In order to ensure that a clock watchdog is required to continuously monitor the enabled clock sources.

In case of external crystal failure the clock source switches automatically to the Internal Slow Clock Source generating f_{OSI} .

11.6.10 Internal Slow Clock Source

The slow internal clock source provides a clock f_{OSI} of 32.768 kHz. This clock can be used as independent clock source for WDT module and as the clock for periodic wake-up events in power saving modes and for continuous adjustments the fast internal backup clock source f_{OFI} .

11.6.11 Clock Gating Control

The clock to peripherals can be individually gated and parts of the system stopped by these means.

Clock gating in Sleep and Deep Sleep modes

Global power management related to clock generation and selection is supported for the sleep modes of the system. The user has full control on the clock configuration for these modes. The registers **SLEEP** and **DSLEEP** control which clocks should remain active and which are to be gated by entering the corresponding Sleep or Deep Sleep mode. Furthermore the system clock can be switched to a slow standby clock and the PLLs put into power-down mode in Deep Sleep mode.

11.7 Debug Behavior

The SCU module does not get affected with the HALTED signal from CPU upon debug activities performed using external debug probe.

11.8 Power, Reset and Clock

The SCU module consists of sub-modules that interact with different power, clock and reset domains. Some of the sub-modules are controlled via dedicated interfaces across the power, clock and reset boundaries. The sub-modules are considered parts of the SCU in the functional sense therefore the complete SCU module is considered a multi domain circuit.

Power domains

Power domains get separated with appropriate power separation cells.

- Pad domain supplied with V_{DDP} voltage
- Analog domain supplied with V_{DDA} voltage
- Core domain supplied with V_{DDC} voltage
- Hibernate domain supplied with V_{BAT} (optionally with a battery or capacitor) or V_{DDP} voltage

Clock domains

All cross-domain interfaces of SCU implement signal synchronization.

- SCU clock in the core domain is f_{SYS}
- Hibernate Control Unit (HCU) clock is f_{OSI} or f_{ULP} (32.768 kHz)
- The Register Mirror Interface of HCU and RTC clock is clocked with f_{STDBY} (32.768 kHz) and f_{SYS} clocks

Reset domains

All reset signals get synchronized to the respective clocks (please refer to **“Reset Control” on Page 11-26** chapter for more details)

- System Reset ($\overline{SYSRESET}$) resets the core logic and can be triggered from various sources.
- Power-on Reset ($\overline{PORESET}$) resets analog modules and contributes in generation of the System Reset. The reset is controlled from internal power validation circuit or driven externally via the bi-directional PORST pin.
- Standby Reset ($\overline{STDBYRESET}$) resets HCU part of SCU. It is triggered by power-up sequence of Hibernate domain and is not affected by power-up sequence of the Core domain. It can also be controlled with software access to **RSTCLR** and **RSTSET** registers.
- Debug Reset ($\overline{DBGRESET}$) is used in debug with an external debug probe.

11.9 Initialization and System Dependencies

The initialization sequence of the XMC4300 is a process taking place before user application software takes control of the system and is comprising of two major phases (see [Figure 11-27](#)), split in several distinctive steps:

Hardware Controlled Initialization Phase

The hardware controlled initialization phase gets performed automatically after power up of the microcontroller. This part is generic and it ensures basic configuration common to most applications. The hardware setup needs to ensure fulfillment of requirements specified in Data Sheet in order to enable reliable start up of the microcontroller before control is handed over to the user software. The sequence where boot code gets executed is considered a part of the hardware controlled phase of the initialization sequence.

For details of the setup requirements please refer to Data Sheet.

Software Controlled Initialization Phase

The software controlled initialization phase is the part of the complete start-up sequence where the application specific configuration gets applied with user software. It involves several steps that are critical for proper operation of the microcontroller in the application context and may also involve some optional configuration actions in order to improve system performance and stability in the application context.

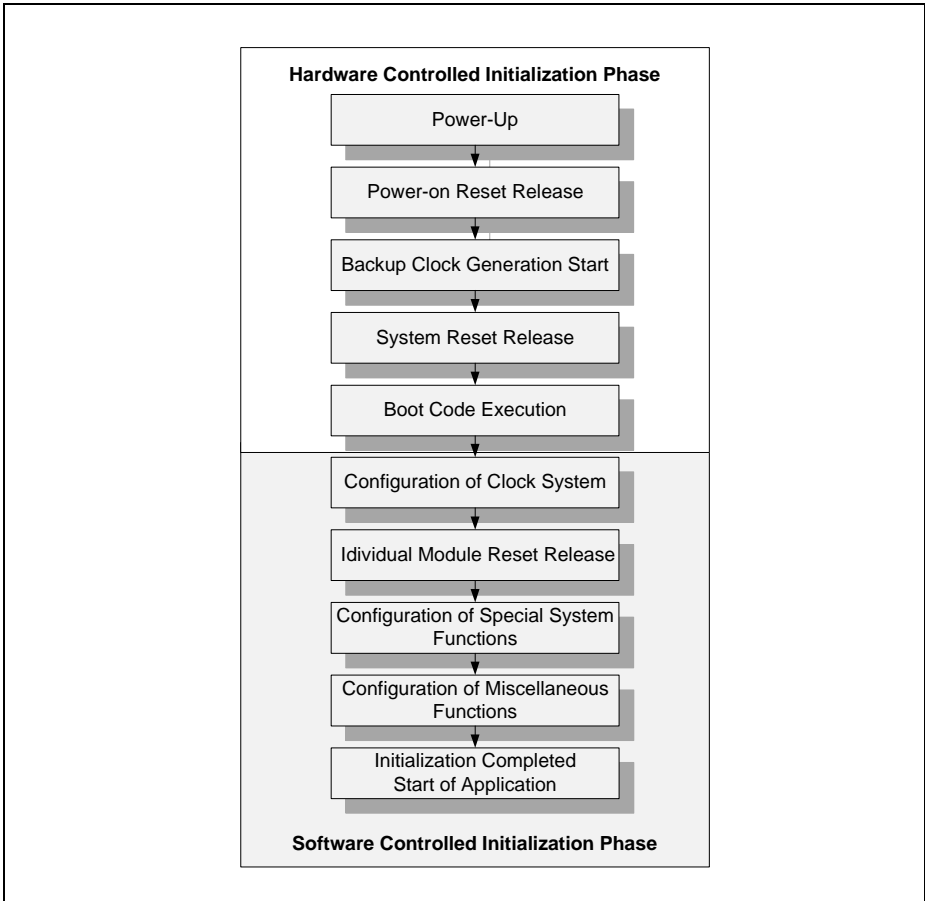


Figure 11-27 Initialization sequence

A more detailed description of the configuration steps is available in the following sub-chapters.

11.9.1 Power-Up

Power up of the microcontroller gets performed by applying VDDP and VDDA supply. Internal voltage generation in the EVR module gets activated automatically after VDDP has been applied (for details of the supply requirements please refer to Data Sheet).

11.9.2 Power-on Reset Release

The XMC4300 implements bi-directional pin PORST for the Power-on Reset PORESET control. The internal Power-on Reset generation is based on the supply and core voltage validation.

The PORST pin may be used either to control the PORESET from an external source in the system or to control the reset of external components from the XMC4300. The PORST function, implemented as open drain driver, allows to share the pin between multiple devices in the system.

An example of the system where XMC4300 may act as the reset control master is shown in [Figure 11-28](#). Any of the devices capable to drive low level of the reset signal is potentially able to assert reset of the system. Release of the reset is effectively performed when devices with output driving capability are not driving low level and the signal is driven high via the pull-up resistance. The driving strength of the pull-up resistance is required to ensure fast release of PORST pin and effectively also fast release of PORESET reset (for details on Power Sequencing please refer to Data Sheet).

Release of the PORESET of XMC4300 results in start of Backup Clock generation.

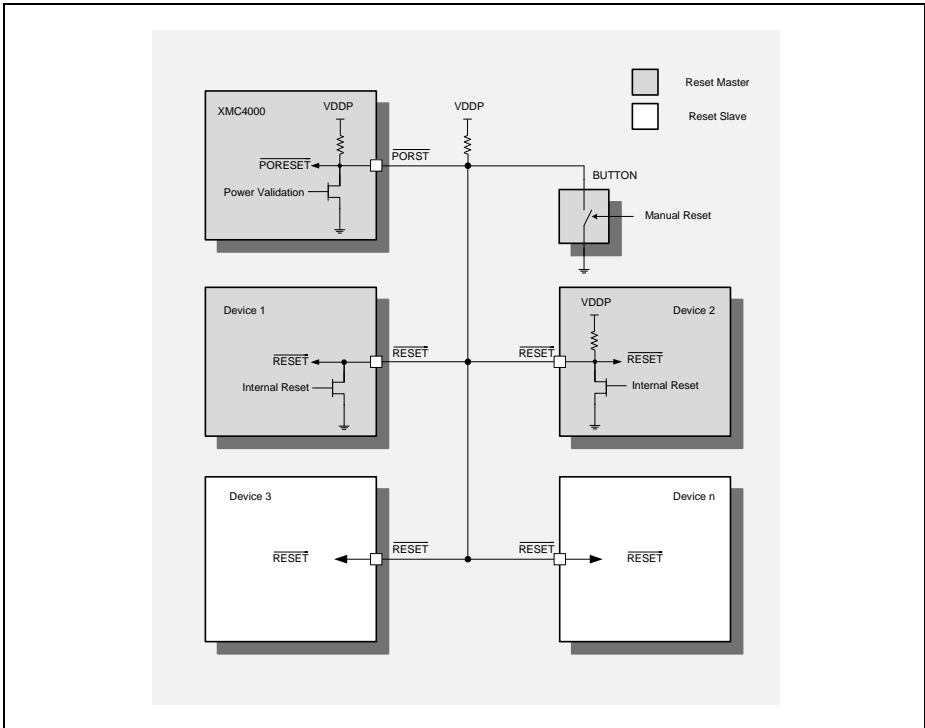


Figure 11-28 System Level Power On Reset Control

11.9.3 System Reset Release

Release of the PORESET and start of the Backup Clock generation results in automatic System Reset SYSRESET release and start of boot code execution.

The System Reset SYSRESET can be triggered from various sources like software controlled CPU reset register, watchdog time-out-triggered reset or a system trap triggered reset. For more details on Reset Control details please refer to **“Reset Control” on Page 11-26**

The cause of the last reset gets automatically stored in the **RSTSTAT** register and can be checked by user software to determine the state of the system and for debug purpose. The reset status in the **RSTSTAT** register shall be reset with **RSTCLR** register after each startup in order to ensure consistent source indication after the next reset.

After SYSRESET release a number of modules clocked with f_{PERIPH} still remain in reset state. Reset release of these modules needs to be released with **PRCLR0**, **PRCLR1**, and **PRCLR2** registers which by default keep the peripheral resets active. It is

System Control Unit (SCU)

recommended to keep the unused modules in reset state in order to reduce power consumption. It is also highly recommended to ensure that clock of the corresponding modules are active before individual peripheral reset release.

11.9.4 Clock System Setup

The following system clocking modes are supported:

- PLL Normal
- PLL Prescaler
- Backup Clock

The default clock signal available after power-up is the internally generated Backup Clock f_{OFI} . The user software starts execution of code clocked with the Backup Clock and can change the clock at any point of time applying a system clock configuration sequence. For details of the clock structure please refer to **“Clock System Overview” on Page 11-31**.

The flowchart in **Figure 11-29** illustrates the recommended system clock configuration sequence. It is strongly recommended to follow the steps in order to ensure proper initialization of the PLL and power sequencing within specified limits (for details on Power Sequencing please refer to Data Sheet). Each of the steps depicted in the diagram may require a sequence of register access actions.

The configuration of the clock system may involve various actions like:

- internal clock trimming configurations, election between Factory or Automatic trimming of the Backup Clock f_{OFI} using **PLLCON0** register
- calibration of the PLL, calculation of the optimal PLL settings using the **Equation (11.5)** on **Page 11-47**
- enabling of external crystal oscillator or direct clock input, configuration of the external clock watchdog with **OSCHPCTRL** register, according to **Equation (11.4)** on **Page 11-45**, selection of the external crystal oscillator or direct clock f_{OSC} using **OSCHPCTRL**
- powering up of the System PLL in **PLLCON0** register
- locking-up of the System PLL, configuration of the System PLL using **PLLCON0**, **PLLCON1** and **PLLCON2** registers
- configuration of clock dividers, switching the system clock to the selected source (please refer to)

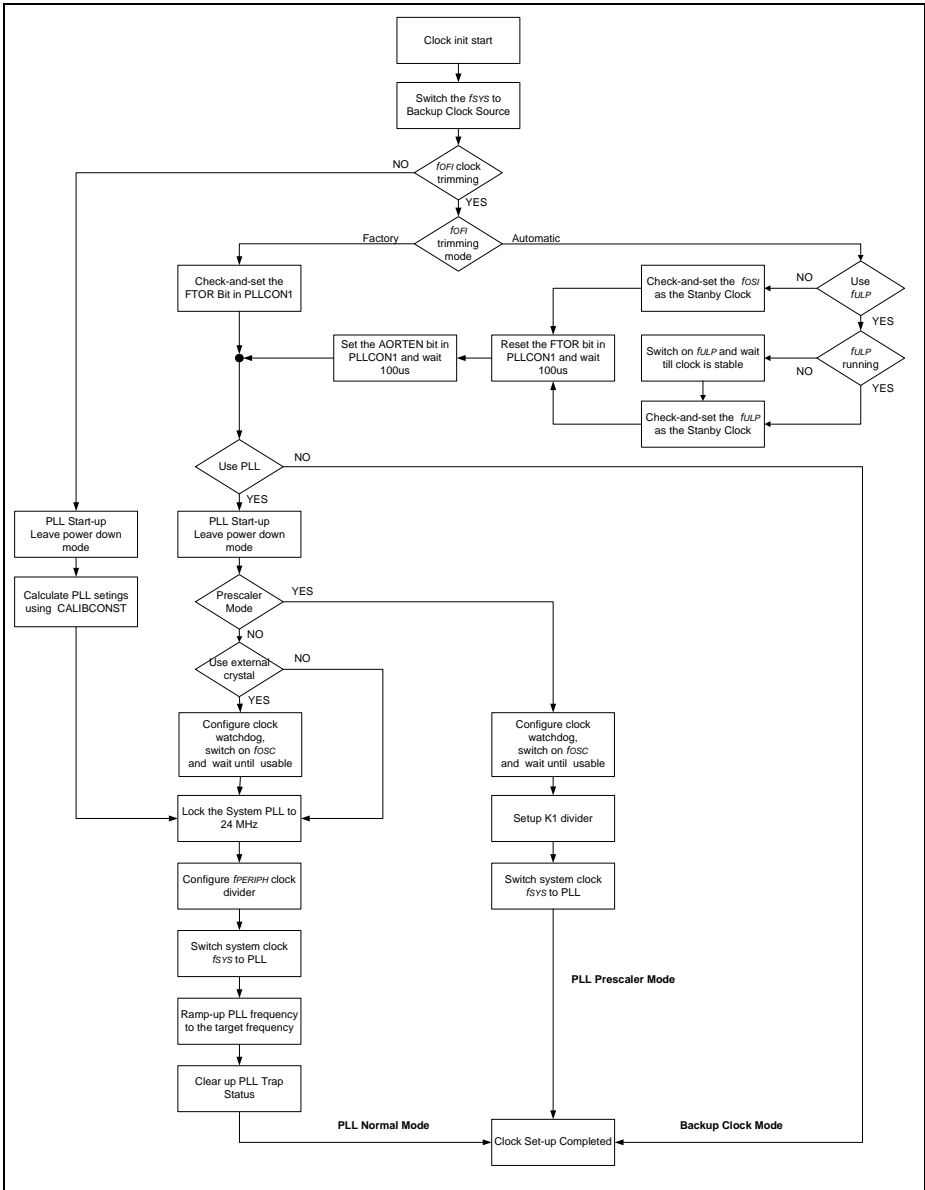


Figure 11-29 Clock initialization sequence

After reset release the system is clocked with a clock derived from the Backup Clock source. If a PLL output clock is required as the system clock source then it is necessary to initialize the respective PLL with a software routine. For details please refer to PLL section in **“Main PLL” on Page 11-39**.

The system relevant and module clocks are configured with the dedicated registers **SYSCCLKCR**, **CPUCLKCR** and **PBCLKCR**.

Some peripheral clocks require explicit enable with **CLKSET** register.

11.9.5 Configuration of Special System Functions

Special system functions are may be required to perform actions that improve system stability and robustness. The following special system functions or modules require initialization before the actual user application starts:

- Memory Parity Protection
- Supply Voltage Brown-out Detection
- Initialization of the Hibernate Domain
- Watchdog Timer (WDT)
- Real Time Clock (RTC)
- System Trap Initialization

Memory Parity Protection

Memory Parity Protection is performed using memory parity check. The parity check can be enabled individually for each instance of memory with **PEEN** register. A trap generation can be individually enabled with **PETE** register in order to get the trap flag reflected in **TRAPRAW** register or to generate System Reset if enabled in **PERSTEN** register.

For details of the Memory Parity Protection please refer to the **“Memory Parity Protection” on Page 11-7**.

Brown Out Detection

Brown out detection mechanism allow active monitoring of the supply voltage and a corrective reaction in case the voltage level is below a programmed threshold. A trap request will be flagged if a programmed condition is detected. For details please refer to **“Supply Voltage Brown-out Detection” on Page 11-20**

Initialization of the Hibernate Domain

Hibernate Control logic and the RTC module needs to be activated with **PWRSET** register before it can be used. This initialization needs to be performed only once after the V_{BAT} has been applied and it stays enabled if V_{BAT} remains applied. After power off

System Control Unit (SCU)

of the main supply of the chip i.e. VDDP, the hibernate domain will remain intact if V_{BAT} is still supplied. For details of hibernate control please refer to **“Hibernate Control” on Page 11-21**. For details of RTC module control please refer to RTC chapter.

Watchdog Timer

The Watchdog Timer requires a clock source selection and activation. It is highly recommended to use reliable clock source, preferably independent from the system clock source in order to ensure corrective action in case of a system failure which will bring the microcontroller into a safe operation state. For more details please refer to WDT chapter. For details of the WDT module configuration please refer to the “Initialization and Control Sequence” section of the WDT chapter.

Real Time Clock

If the real time clock is required then the initialization must take place after Hibernate domain has been activated. For details of the RTC module configuration please refer to the “Initialization and Control Sequence” section of the RTC chapter.

System Trap Initialization

System Traps are by default disabled and need to be enabled with user software before used. Any active trap flag will be reflected in the **TRAPRAW** register. In order to enable an NMI interrupt generation it needs to be unmasked in **TRAPDIS** register.

For details of the System Trap configuration please refer to the **“Trap Generation” on Page 11-9**.

11.9.6 Configuration of Miscellaneous Functions

A number of miscellaneous functions may be used as a part of the user application, or, e.g. by an Operating System. The following functions are available and require configuration before used:

- Power Saving modes
- Die Temperature Sensor (DTS)
- Out of Range Comparators for analog I/Os

Power Saving modes

Different power modes like Sleep, Deep Sleep and Hibernate mode require configuration of their functions and configuration of wake-up triggers prior to entering the modes. For details of the available modes and configuration please refer to **“Power Management” on Page 11-14**

Die Temperature Sensor

The Die Temperature Sensor allows to perform temperature measurements of the die. The module needs to be enabled with **DTSCON** register before used. For details please refer to **“Die Temperature Measurement” on Page 11-10**

Out of Range Comparators

The out of range comparator serves the purpose of overvoltage monitoring for analog input pins of the chip. This functionality can be enabled with registers **G0ORCEN** and **G1ORCEN**.

11.10 Registers

This section describes the registers of SCU. Most of the registers are reset SYSRESET reset signal but some of the registers can be reset only with PORST reset.

Table 11-9 Base Addresses of sub-sections of SCU registers

Short Name	Description	Offset Addr. ¹⁾
GCU Registers	Offset address of General Control Unit	0000 _H
PCU Registers	Offset address of Power Control Unit	0200 _H
HCU Registers	Offset address of Hibernate Control Unit	0300 _H
RCU Registers	Offset address of Reset Control Unit	0400 _H
CCU Registers	Offset address of Clock Control Unit	0600 _H

1) The absolute register address is calculated as follows:

Module Base Address + Sub-Module Offset Address (shown in this column) + Register Offset Address

Following access result an AHB error response:

- Read or write access to undefined address
- Write access in user mode to registers which allow only privileged mode write
- Write access to read-only registers
- Write access to startup protected registers

Table 11-10 Registers Address Space

Module	Base Address	End Address	Note
SCU	5000 4000 _H	5000 7FFF _H	System Control Unit Registers

Table 11-11 Registers Overview

Short Name	Register Long Name	Offset Addr. ¹⁾	Access Mode		Description
			Read	Write	

General SCU Registers

GCU Registers

ID	Module Identification Register	0000 _H	U, PV	BE	Page 11-67
IDCHIP	Chip ID	0004 _H	U, PV	BE	Page 11-68

System Control Unit (SCU)
Table 11-11 Registers Overview (cont'd)

Short Name	Register Long Name	Offset Addr. 1)	Access Mode		Description
			Read	Write	
IDMANUF	Manufactory ID	0008 _H	U, PV	BE	Page 11-68
STCON	Start-up Control	0010 _H	U, PV	PV	Page 11-69
GPR0	General Purpose Register 0	002C _H	U, PV	PV	Page 11-70
GPR1	General Purpose Register 1	0030 _H	U, PV	PV	Page 11-70
CCUCON	CCUx Global Start Control Register	004C _H	U, PV	U	Page 11-73
SRSTAT	Service Request Status	0074 _H	U, PV	U, PV	Page 11-75
SRRAW	RAW Service Request Status	0078 _H	U, PV	BE	Page 11-77
SRMSK	Service Request Mask	007C _H	U, PV	PV	Page 11-79
SRCLR	Service Request Clear	0080 _H	nBE	PV	Page 11-81
SRSET	Service Request Set	0084 _H	nBE	PV	Page 11-83
NMIREQEN	Enable Promoting Events to NMI Request	0088 _H	U, PV	PV	Page 11-85
DTSCON	DTS Control	008C _H	U, PV	PV	Page 11-86
DTSSTAT	DTS Status	0090 _H	U, PV	BE	Page 11-87
G0ORCEN	Out-Of-Range Comparator Enable Register	00A0 _H	U, PV	U, PV	Page 11-88
G1ORCEN	Out-Of-Range Comparator Enable Register	00A4 _H	U, PV	U, PV	Page 11-89
MIRRSTS	Mirror Update Status Register	00C4 _H	U, PV	BE	Page 11-92
RMACR	Retention Memory Access Control Register	00C8 _H	U, PV	U, PV	Page 11-93
RMADATA	Retention Memory Access Data Register	00CC _H	U, PV	U, PV	Page 11-94
PEEN	Parity Error Enable Register	013C _H	U, PV	PV	Page 11-95

Table 11-11 Registers Overview (cont'd)

Short Name	Register Long Name	Offset Addr. 1)	Access Mode		Description
			Read	Write	
MCHKCON	Memory Checking Control Register	0140 _H	U, PV	PV	Page 11-97
PETE	Parity Error Trap Enable Register	0144 _H	U, PV	PV	Page 11-99
PERSTEN	Reset upon Parity Error Enable Register	0148 _H	U, PV	PV	Page 11-101
PEFLAG	Parity Error Control Register	0150 _H	U, PV	PV	Page 11-101
PMPTR	Parity Memory Test Pattern Register	0154 _H	U, PV	PV	Page 11-103
PMTSR	Parity Memory Test Select Register	0158 _H	U, PV	PV	Page 11-105
TRAPSTAT	Trap Status Register	0160 _H	U, PV	BE	Page 11-106
TRAPRAW	Trap Raw Status Register	0164 _H	U, PV	BE	Page 11-108
TRAPDIS	Trap Mask Register	0168 _H	U, PV	PV	Page 11-109
TRAPCLR	Trap Clear Register	016C _H	nBE	PV	Page 11-111
TRAPSET	Trap Set Register	0170 _H	nBE	PV	Page 11-112
PCU Registers					
PWRSTAT	Power Status Register	0000 _H	U, PV		Page 11-121
PWRSET	Power Set Control Register	0004 _H	nBE	PV	Page 11-121
PWRCLR	Power Clear Control Register	0008 _H	nBE	PV	Page 11-122
EVRSTAT	EVR Status Register	0010 _H	U, PV	BE	Page 11-123
EVRVADCSTAT	EVR VADC Status Register	0014 _H	U, PV	BE	Page 11-124
PWRMON	Power Monitor Value	002C _H	U, PV	PV	Page 11-125
HCU Registers					
HDSTAT	Hibernate Domain Status Register	0000 _H	U, PV	PV	Page 11-126

System Control Unit (SCU)

Table 11-11 Registers Overview (cont'd)

Short Name	Register Long Name	Offset Addr. 1)	Access Mode		Description
			Read	Write	
HDCLR	Hibernate Domain Status Clear Register	0004 _H	U, PV	PV	Page 11-127
HDSET	Hibernate Domain Status Set Register	0008 _H	U, PV	PV	Page 11-128
HDCR	Hibernate Domain Control Register	000C _H	U, PV	PV	Page 11-129
OSCSICTRL	Internal 32.768 kHz Clock Source Control Register	0014 _H	U, PV	PV	Page 11-132
OSCULSTAT	OSC_ULP Status Register	0018 _H	U, PV	BE	Page 11-133
OSCULCTRL	OSC_ULP Control Register	001C _H	U, PV	PV	Page 11-133
RCU Registers					
RSTSTAT	System Reset Status	0000 _H	U, PV	BE	Page 11-134
RSTSET	Reset Set Register	0004 _H	nBE	PV	Page 11-136
RSTCLR	Reset Clear Register	0008 _H	nBE	PV	Page 11-137
PRSTAT0	Peripheral Reset Status Register 0	000C _H	U, PV	PV	Page 11-138
PRSET0	Peripheral Reset Set Register 0	0010 _H	nBE	PV	Page 11-139
PRCLR0	Peripheral Reset Clear Register 0	0014 _H	nBE	PV	Page 11-140
PRSTAT1	Peripheral Reset Status Register 1	0018 _H	U, PV	BE	Page 11-142
PRSET1	Peripheral Reset Set Register 1	001C _H	nBE	PV	Page 11-143
PRCLR1	Peripheral Reset Clear Register 1	0020 _H	nBE	PV	Page 11-144
PRSTAT2	Peripheral Reset Status Register 2	0024 _H	U, PV	BE	Page 11-145

Table 11-11 Registers Overview (cont'd)

Short Name	Register Long Name	Offset Addr. 1)	Access Mode		Description
			Read	Write	
PRSET2	Peripheral Reset Set Register 2	0028 _H	nBE	PV	Page 11-147
PRCLR2	Peripheral Reset Clear Register 2	002C _H	nBE	PV	Page 11-148
CCU Registers					
CLKSTAT	Clock Status Register	0000 _H	U,PV	BE	Page 11-149
CLKSET	Clock Set Control Register	0004 _H	nBE	PV	Page 11-150
CLKCLR	Clock clear Control Register	0008 _H	nBE	PV	Page 11-151
SYSCCLKCR	System Clock Control	000C _H	U, PV	PV	Page 11-152
CPUCCLKCR	CPU Clock Control	0010 _H	U, PV	PV	Page 11-153
PBCLKCR	Peripheral Bus Clock Control	0014 _H	U, PV	PV	Page 11-154
USBCLKCR	USB Clock Control	0018 _H	U, PV	PV	Page 11-154
CCUCLKCR	CCU Clock Control	0020 _H	U, PV	PV	Page 11-155
WDTCLKCR	WDT Clock Control	0024 _H	U, PV	PV	Page 11-156
EXTCLKCR	External clock Control Register	0028 _H	U, PV	PV	Page 11-157
MLINKCLKCR	Multi-Link Clock Control Register	002C _H	U, PV	PV	Page 11-157
SLEEPKR	Sleep Control Register	0030 _H	U, PV	PV	Page 11-159
DSLEEPKR	Deep Sleep Control Register	0034 _H	U, PV	PV	Page 11-161
OSCHPSTAT	OSC_HP Status Register	0100 _H	U, PV	BE	Page 11-173
OSCHPCTRL	OSC_HP Control Register	0104 _H	U, PV	PV	Page 11-174
CLKCALCONST	Clock Calibration Constant Register	010C _H	U, PV	SP	Page 11-175
PLLSTAT	System PLL Status Register	0110 _H	U, PV	BE	Page 11-176

Table 11-11 Registers Overview (cont'd)

Short Name	Register Long Name	Offset Addr. 1)	Access Mode		Description
			Read	Write	
PLLCON0	System PLL Configuration 0 Register	0114 _H	U, PV	PV	Page 11-178
PLLCON1	System PLL Configuration 1 Register	0118 _H	U, PV	PV	Page 11-180
PLLCON2	System PLL Configuration 2 Register	011C _H	U, PV	PV	Page 11-181
USBPLLSTAT	USB PLL Status Register	0120 _H	U, PV	PV	Page 11-182
USBPLLCON	USB PLL Control Register	0124 _H	U, PV	PV	Page 11-183
CLKMXSTAT	Clock Multiplexing Status Register	0138 _H	U, PV	BE	Page 11-185

1) The absolute register address is calculated as follows:
Module Base Address + Sub-Module Offset Address + Offset Address (shown in this column)

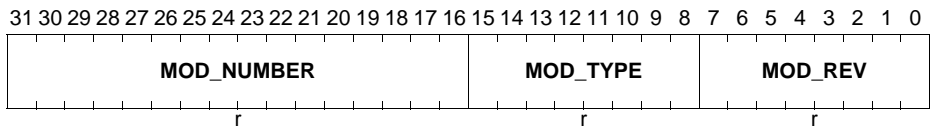
11.10.1 GCU Registers

ID

Register containing unique ID of the module.

ID

SCU Module ID Register (0000_H) Reset Value: 0090 C0XX_H



Field	Bits	Type	Description
MOD_REV	[7:0]	r	Module Revision Indicates the revision number of the implementation. This information depends on the design step.
MOD_TYPE	[15:8]	r	Module Type This internal marker is fixed to C0 _H .

System Control Unit (SCU)

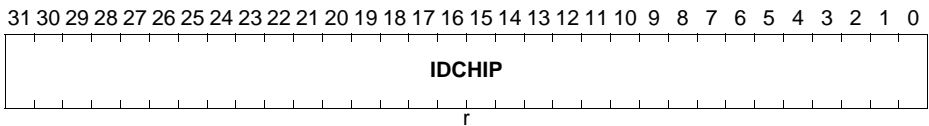
Field	Bits	Type	Description
MOD_NUMBER	[31:16]	r	Module Number Indicates the module identification number

IDCHIP

Register containing unique ID of the chip.

IDCHIP

Chip ID Register (0004_H) **Reset Value: XXXX XXXX_H**



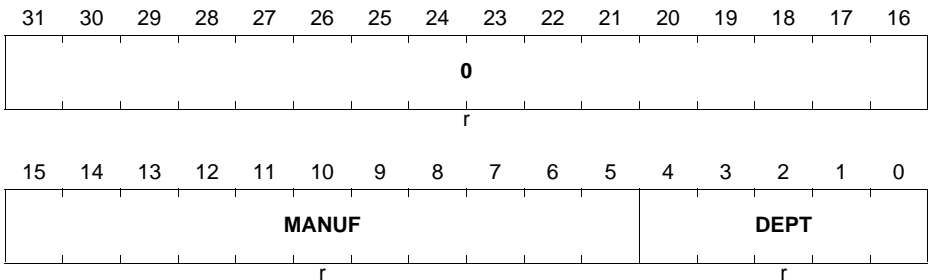
Field	Bits	Type	Description
IDCHIP	[31:0]	r	Chip ID

IDMANUF

Register containing unique factory ID of the chip.

IDMANUF

Manufactory ID Register (0008_H) **Reset Value: 0000 1820_H**



Field	Bits	Type	Description
DEPT	[4:0]	r	Department Identification Number DEPT indicates department within Infineon Technologies.

System Control Unit (SCU)

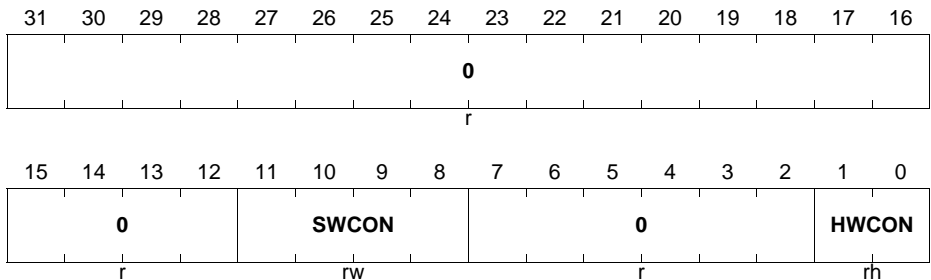
Field	Bits	Type	Description
MANUF	[15:5]	r	Manufacturer Identification Number JEDEC normalized Manufacturer code. MANUF = C1 _H stands for Infineon Technologies.
0	[31:16]	r	Reserved

STCON

Startup configuration register determining boot process of the chip.

STCON

Startup Configuration Register (0010_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
HWCON	[1:0]	rh	HW Configuration At PORESET the following values are latched HWCON.0 = not (TMS) HWCON.1 = TCK 00 _B Normal mode, JTAG 01 _B ASC BSL enabled 10 _B BMI customized boot enabled 11 _B CAN BSL enabled

System Control Unit (SCU)

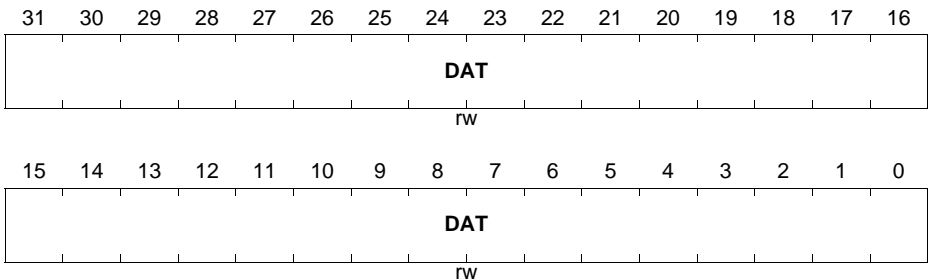
Field	Bits	Type	Description
SWCON	[11:8]	rw	SW Configuration Bit[9:8] is copy of Bit[1:0] after PORESET 0000 _B Normal mode, boot from Boot ROM 0001 _B ASC BSL enabled 0010 _B BMI customized boot enabled 0011 _B CAN BSL enabled 0100 _B Boot from Code SRAM 1000 _B Boot from alternate Flash Address 0 1100 _B Boot from alternate Flash Address 1 1110 _B Enable fallback Alternate Boot Mode (ABM) Note: Only reset with Power-on Reset
0	[7:2], [31:12]	r	Reserved Read as 0; should be written with 0.

GPRx

Software support registers. Can be reset only with PORST reset.

GPRx (x=0-1)

General Purpose Register x **(002C_H+ x*4)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DAT	[31:0]	rw	User Data 32-bit data <i>Note: GPRx registers can be reset with PORST reset only</i>

ETH0_CON

ETH0 module configuration register.

ETH0_CON

Ethernet 0 Port Control Register (50004040_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				INFS EL	0	MDIO		0	CLK_TX		COL				
r				rw	r	rw		r	rw		rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXER		CRS		CRS_DV		CLK_RMII		RXD3		RXD2		RXD1		RXD0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
RXD0	[1:0]	rw	<p>MAC Receive Input 0</p> <p>This bit field indicates the receive input position of the RXD0 signal.</p> <p>00_B Data input RXD0A is selected</p> <p>01_B Data input RXD0B is selected</p> <p>10_B Data input RXD0C is selected</p> <p>11_B Data input RXD0D is selected</p>
RXD1	[3:2]	rw	<p>MAC Receive Input 1</p> <p>This bit field indicates the receive input position of the RXD1 signal.</p> <p>00_B Data input RXD1A is selected</p> <p>01_B Data input RXD1B is selected</p> <p>10_B Data input RXD1C is selected</p> <p>11_B Data input RXD1D is selected</p>
RXD2	[5:4]	rw	<p>MAC Receive Input 2</p> <p>This bit field indicates the receive input position of the RXD2 signal.</p> <p>00_B Data input RXD2A is selected</p> <p>01_B Data input RXD2B is selected</p> <p>10_B Data input RXD2C is selected</p> <p>11_B Data input RXD2D is selected</p>

System Control Unit (SCU)

Field	Bits	Type	Description
RXD3	[7:6]	rw	MAC Receive Input 3 This bit field indicates the receive input position of the RXD3 signal. 00 _B Data input RXD3A is selected 01 _B Data input RXD3B is selected 10 _B Data input RXD3C is selected 11 _B Data input RXD3D is selected
CLK_RMII	[9:8]	rw	RMII clock input This bit field indicates the receive input position of the RMII clock input signal. 00 _B Data input RMIIA is selected 01 _B Data input RMIIB is selected 10 _B Data input RMIIC is selected 11 _B Data input RMIID is selected
CRS_DV	[11:10]	rw	CRS_DV input This bit field indicates the receive input position of the CRS_DV input signal. 00 _B Data input CRS_DVA is selected 01 _B Data input CRS_DVB is selected 10 _B Data input CRS_DVC is selected 11 _B Data input CRS_DVD is selected
CRS	[13:12]	rw	CRS input This bit field indicates the receive input position of the CRS input signal. 00 _B Data input CRSA 01 _B Data input CRSB 10 _B Data input CRSC 11 _B Data input CRSD
RXER	[15:14]	rw	RXER Input This bit field indicates the receive input position of the RXER input signal. 00 _B Data input RXERA is selected 01 _B Data input RXERB is selected 10 _B Data input RXERC is selected 11 _B Data input RXERD is selected

System Control Unit (SCU)

Field	Bits	Type	Description
COL	[17:16]	rw	COL input This bit field indicates the receive input position of the COL clock input signal. 00 _B Data input COLA is selected 01 _B Data input COLB is selected 10 _B Data input COLC is selected 11 _B Data input COLD is selected
CLK_TX	[19:18]	rw	CLK_TX input This bit field indicates the receive input position of the CLK_TX input signal. 00 _B Data input CLK_TXA is selected 01 _B Data input CLK_TXB is selected 10 _B Data input CLK_TXC is selected 11 _B Data input CLK_TXD is selected
MDIO	[23:22]	rw	MDIO Input Select This bit field selects the input position of the MDI signal. 00 _B Data input MDIA is selected 01 _B Data input MDIB is selected 10 _B Data input MDIC is selected 11 _B Data input MDID is selected
INFSEL	26	rw	Ethernet MAC Interface Selection This bit selects Ethernet MAC interface to PHY. 0 _B MII 1 _B RMII
0	[21:20], [25:24], [31:27]	r	Reserved Read as 0; should be written with 0.

CCUCON

CAPCOM module control register. Individual signals signal is generated with f_{CCU} clock.

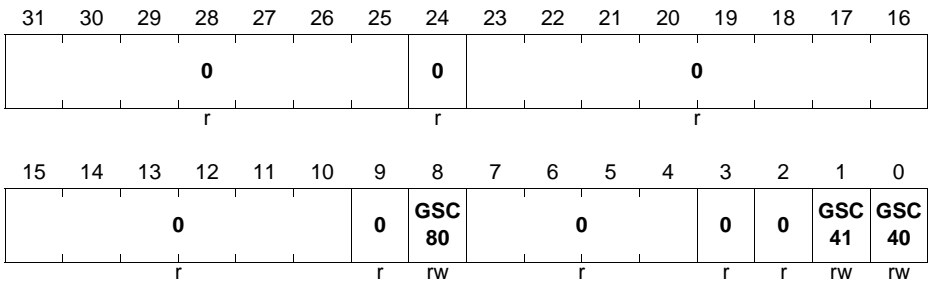
System Control Unit (SCU)

CCUCON

CCU Control Register

(004C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
GSC40	0	rw	<p>Global Start Control CCU40</p> <p>This register can be used to control a synchronous start of multiple timers of CCU40.</p> <p>It also can be used to control additional functions that are available in the timers, e.g. Count or Capture. For a complete description of the available set of functions, please address the specific section of the CCU4 chapters.</p> <p>Writing 1 or 0 into this field will not by itself trigger any action and one must before configure the specific peripheral function accordingly.</p>
GSC41	1	rw	<p>Global Start Control CCU41</p> <p>This register can be used to control a synchronous start of multiple timers of CCU41.</p> <p>It also can be used to control additional functions that are available in the timers, e.g. Count or Capture. For a complete description of the available set of functions, please address the specific section of the CCU4 chapters.</p> <p>Writing 1 or 0 into this field will not by itself trigger any action and one must before configure the specific peripheral function accordingly.</p>

System Control Unit (SCU)

Field	Bits	Type	Description
GSC80	8	rw	Global Start Control CCU80 This register can be used to control a synchronous start of multiple timers of CCU80. It also can be used to control additional functions that are available in the timers, e.g. Count or Capture. For a complete description of the available set of functions, please address the specific section of the CCU8 chapters. Writing 1 or 0 into this field will not by itself trigger any action and one must before configure the specific peripheral function accordingly.
0	2,3,9, [7:4], [23:10], 24, [31:25]	r	Reserved Read as 0; should be written with 0.

SRSTAT

Service request status reflecting masking with SRMSK mask register. Write one to a bit in SRCLR register to clear a bit or SRSET to set a bit. Writing zero has no effect. Outputs of this register are used to trigger interrupts or service requests.

SRSTAT

SCU Service Request Status (0074_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		RMX	RTC _TIM 1	RTC _TIM 0	RTC _ATI M1	RTC _ATI M0	RTC _CT R	OSC ULC TRL	0	OSC SIC TRL	0	HDC R	HDS ET	HDC LR	0
r		rh	rh	rh	rh	rh	rh	rh	r	rh	r	rh	rh	rh	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					0					0		DLR OVR	AI	PI	PRW ARN
r					r					r		rh	rh	rh	rh

Field	Bits	Type	Description
PRWARN	0	rh	WDT pre-warning Interrupt Status 0 _B Inactive 1 _B Active
PI	1	rh	RTC Periodic Interrupt Status Set whenever periodic counter increments
AI	2	rh	Alarm Interrupt Status Set whenever count value matches compare value
DLROVR	3	rh	DLR Request Overrun Interrupt Status Set whenever DLR overrun condition occurs.
HDCLR	17	rh	HDCLR Mirror Register Update Status 0 _B Not updated 1 _B Update completed
HDSET	18	rh	HDSET Mirror Register Update Status 0 _B Not updated 1 _B Update completed
HDCR	19	rh	HDCR Mirror Register Update Status 0 _B Not updated 1 _B Update completed
OSCSICTRL	21	rh	OSCSICTRL Mirror Register Update Status 0 _B Not updated 1 _B Update completed
OSCULCTRL	23	rh	OSCULCTRL Mirror Register Update Status 0 _B Not updated 1 _B Update completed
RTC_CTR	24	rh	RTC CTR Mirror Register Update Status 0 _B Not updated 1 _B Update completed
RTC_ATIM0	25	rh	RTC ATIM0 Mirror Register Update Status 0 _B Not updated 1 _B Update completed
RTC_ATIM1	26	rh	RTC ATIM1 Mirror Register Update Status 0 _B Not updated 1 _B Update completed
RTC_TIM0	27	rh	RTC TIM0 Mirror Register Update Status 0 _B Not updated 1 _B Update completed

System Control Unit (SCU)

Field	Bits	Type	Description
PI	1	rh	RTC Raw Periodic Interrupt Status Before Masking Set whenever periodic counter increments
AI	2	rh	RTC Raw Alarm Interrupt Status Before Masking Set whenever count value matches compare value
DLROVR	3	rh	DLR Request Overrun Interrupt Status Before Masking Set whenever DLR overrun condition occurs.
HDCLR	17	rh	HDCLR Mirror Register Update Status Before Masking 0 _B Not updated 1 _B Update completed
HDSET	18	rh	HDSET Mirror Register Update Status Before Masking 0 _B Not updated 1 _B Update completed
HDCR	19	rh	HDCR Mirror Register Update Status Before Masking 0 _B Not updated 1 _B Update completed
OSCSICTRL	21	rh	OSCSICTRL Mirror Register Update Status Before Masking 0 _B Not updated 1 _B Update completed
OSCUCTRL	23	rh	OSCUCTRL Mirror Register Update Status Before Masking 0 _B Not updated 1 _B Update completed
RTC_CTR	24	rh	RTC CTR Mirror Register Update Status Before Masking 0 _B Not updated 1 _B Update completed
RTC_ATIM0	25	rh	RTC ATIM0 Mirror Register Update Status Before Masking 0 _B Not updated 1 _B Update completed

Field	Bits	Type	Description
PRWARN	0	rw	WDT pre-warning Interrupt Mask 0 _B Disabled 1 _B Enabled
PI	1	rw	RTC Periodic Interrupt Mask 0 _B Disabled 1 _B Enabled
AI	2	rw	RTC Alarm Interrupt Mask 0 _B Disabled 1 _B Enabled
DLROVR	3	rw	DLR Request Overrun Interrupt Mask 0 _B Disabled 1 _B Enabled
HDCLR	17	rw	HDCLR Mirror Register Update Mask 0 _B Disabled 1 _B Enabled
HDSET	18	rw	HDSET Mirror Register Update Mask 0 _B Disabled 1 _B Enabled
HDCR	19	rw	HDCR Mirror Register Update Mask 0 _B Disabled 1 _B Enabled
OSCSICTRL	21	rw	OSCSICTRL Mirror Register Update Mask 0 _B Disabled 1 _B Enabled
OSCUCTRL	23	rw	OSCUCTRL Mirror Register Update Mask 0 _B Disabled 1 _B Enabled
RTC_CTR	24	rw	RTC CTR Mirror Register Update Mask 0 _B Disabled 1 _B Enabled
RTC_ATIM0	25	rw	RTC ATIM0 Mirror Register Update Mask 0 _B Disabled 1 _B Enabled
RTC_ATIM1	26	rw	RTC ATIM1 Mirror Register Update Mask 0 _B Disabled 1 _B Enabled

System Control Unit (SCU)

Field	Bits	Type	Description
RTC_TIM0	27	rw	RTC TIM0 Mirror Register Update Mask 0 _B Disabled 1 _B Enabled
RTC_TIM1	28	rw	RTC TIM1 Mirror Register Update Mask 0 _B Disabled 1 _B Enabled
RMX	29	rw	Retention Memory Mirror Register Update Mask 0 _B Disabled 1 _B Enabled
0	[5:4], [14:6], 15,16, 20, 22, [31:30]	r	Reserved Read as 0; should be written with 0.

SRCLR

Clear service request bits of registers **SRRAW** and **SRSTAT**. Write one to clear corresponding bits. Writing zeros has no effect.

SRCLR

SCU Service Request Clear (0080_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	RMX	RTC_TIM 1	RTC_TIM 0	RTC_ATI M1	RTC_ATI M0	RTC_CT R	OSC_ULC TRL	0	OSC_SICT RL	0	HDC_R	HDS_ET	HDC_LR	0	
r	w	w	w	w	w	w	w	r	w	r	w	w	w	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					0					0	DLR_OVR	AI	PI	PRW ARN	
r					r					r	w	w	w	w	

Field	Bits	Type	Description
PRWARN	0	w	WDT pre-warning Interrupt Clear 0 _B No effect 1 _B Clear the status bit

System Control Unit (SCU)

Field	Bits	Type	Description
PI	1	w	RTC Periodic Interrupt Clear 0 _B No effect 1 _B Clear the status bit
AI	2	w	RTC Alarm Interrupt Clear 0 _B No effect 1 _B Clear the status bit
DLROVR	3	w	DLR Request Overrun Interrupt clear 0 _B No effect 1 _B Clear the status bit
HDCLR	17	w	HDCLR Mirror Register Update Clear 0 _B No effect 1 _B Clear the status bit
HDSET	18	w	HDSET Mirror Register Update Clear 0 _B No effect 1 _B Clear the status bit
HDCR	19	w	HDCR Mirror Register Update Clear 0 _B No effect 1 _B Clear the status bit
OSCSICTRL	21	w	OSCSICTRL Mirror Register Update Clear 0 _B No effect 1 _B Clear the status bit
OSCULCTRL	23	w	OSCULCTRL Mirror Register Update Clear 0 _B No effect 1 _B Clear the status bit
RTC_CTR	24	w	RTC CTR Mirror Register Update Clear 0 _B No effect 1 _B Clear the status bit
RTC_ATIM0	25	w	RTC ATIM0 Mirror Register Update Clear 0 _B No effect 1 _B Clear the status bit
RTC_ATIM1	26	w	RTC ATIM1 Mirror Register Update Clear 0 _B No effect 1 _B Clear the status bit
RTC_TIM0	27	w	RTC TIM0 Mirror Register Update Clear 0 _B No effect 1 _B Clear the status bit

System Control Unit (SCU)

Field	Bits	Type	Description
RTC_TIM1	28	w	RTC TIM1 Mirror Register Update Clear 0 _B No effect 1 _B Clear the status bit
RMX	29	w	Retention Memory Mirror Register Update Clear 0 _B No effect 1 _B Clear the status bit
0	[5:4], [14:6], 15,16, 20, 22, [31:30]	r	Reserved Read as 0; should be written with 0.

SRSET

Set service request fits of registers **SRRAW** and **SRSTAT**. Write one to clear corresponding bits. Writing zeros has no effect.

SRSET

SCU Service Request Set (0084_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	RMX	RTC_TIM1	RTC_TIM0	RTC_ATI_M1	RTC_ATI_M0	RTC_CT_R	OSC_ULC_TRL	0	OSC_SICT_RL	0	HDC_R	HDC_RST	HDC_RCL_R	0	
r	w	w	w	w	w	w	w	r	w	r	w	w	w	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					0					0	DLR_OVR	AI	PI	PRW_ARN	
r					r					r	w	w	w	w	

Field	Bits	Type	Description
PRWARN	0	w	WDT pre-warning Interrupt Set 0 _B No effect 1 _B set the status bit
PI	1	w	RTC Periodic Interrupt Set 0 _B No effect 1 _B set the status bit

System Control Unit (SCU)

Field	Bits	Type	Description
AI	2	w	RTC Alarm Interrupt Set 0 _B No effect 1 _B set the status bit
DLROVR	3	w	DLR Request Overrun Interrupt Set 0 _B No effect 1 _B set the status bit
HDCRCLR	17	w	HDCRCLR Mirror Register Update Set 0 _B No effect 1 _B set the status bit
HDCRSET	18	w	HDCRSET Mirror Register Update Set 0 _B No effect 1 _B set the status bit
HDCR	19	w	HDCR Mirror Register Update Set 0 _B No effect 1 _B set the status bit
OSCSICTRL	21	w	OSCSICTRL Mirror Register Update Set 0 _B No effect 1 _B set the status bit
OSCUCTRL	23	w	OSCUCTRL Mirror Register Update Set 0 _B No effect 1 _B set the status bit
RTC_CTR	24	w	RTC CTR Mirror Register Update Set 0 _B No effect 1 _B set the status bit
RTC_ATIM0	25	w	RTC ATIM0 Mirror Register Update Set 0 _B No effect 1 _B set the status bit
RTC_ATIM1	26	w	RTC ATIM1 Mirror Register Update Set 0 _B No effect 1 _B set the status bit
RTC_TIM0	27	w	RTC TIM0 Mirror Register Update Set 0 _B No effect 1 _B set the status bit
RTC_TIM1	28	w	RTC TIM1 Mirror Register Update Set 0 _B No effect 1 _B set the status bit

System Control Unit (SCU)

Field	Bits	Type	Description
AI	2	rw	Promote RTC Alarm Interrupt Request to NMI Request 0 _B Disabled 1 _B Enabled
ERU00	16	rw	Promote Channel 0 Interrupt of ERU0 Request to NMI Request 0 _B Disabled 1 _B Enabled
ERU01	17	rw	Promote Channel 1 Interrupt of ERU0 Request to NMI Request 0 _B Disabled 1 _B Enabled
ERU02	18	rw	Promote Channel 2 Interrupt of ERU0 Request to NMI Request 0 _B Disabled 1 _B Enabled
ERU03	19	rw	Promote Channel 3 Interrupt of ERU0 Request to NMI Request 0 _B Disabled 1 _B Enabled
0	[15:3], [31:20]	r	Reserved Read as 0; should be written with 0.

DTSCON

Die temperature sensor control register

DTSCON

Die Temperature Sensor Control Register (008C_H) **Reset Value: 0000 0001_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						BGTRIM				REFTRIM			GA I N		
r						rw				rw			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GAIN					OFFSET						0	STA RT	PWD		
rw					rw						r	w	rw		

Field	Bits	Type	Description
PWD	0	rw	Sensor Power Down This bit defines the DTS power state. 0 _B The DTS is powered 1 _B The DTS is not powered
START	1	w	Sensor Measurement Start This bit starts a measurement of the DTS. 0 _B No DTS measurement is started 1 _B DTS measurement is started If set this bit is automatically cleared. This bit always reads as zero.
OFFSET	[10:4]	rw	Offset Calibration Value This bit field interfaces the offset calibration values to the DTS. The calibration values are forwarded to the DTS by setting bit START.
GAIN	[16:11]	rw	Gain Calibration Value This bit field interfaces the gain calibration values to the DTS. The calibration values are forwarded to the DTS by setting bit START.
REFTRIM	[19:17]	rw	Reference Trim Calibration Value This bit field interfaces the reference trim calibration values to the DTS. The calibration values are forwarded to the DTS by setting bit START.
BGTRIM	[23:20]	rw	Bandgap Trim Calibration Value This bit field interfaces the bandgap trim calibration values to the DTS. The calibration values are forwarded to the DTS by setting bit START.
0	[3:2], [31:24]	r	Reserved Read as 0; should be written with 0.

DTSSTAT

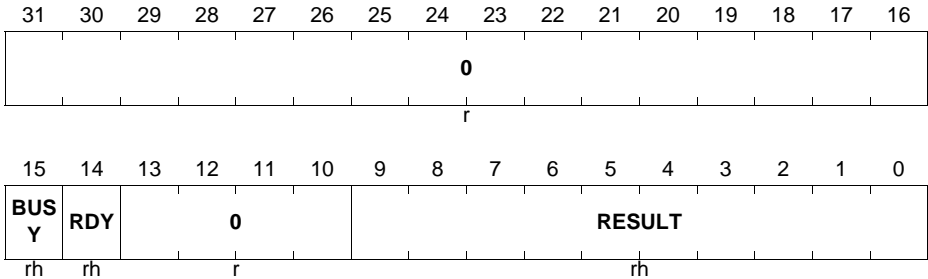
Die temperature status register

System Control Unit (SCU)

DTSSTAT

Die Temperature Sensor Status Register (0090_H)

Reset Value: 0000 0000_H



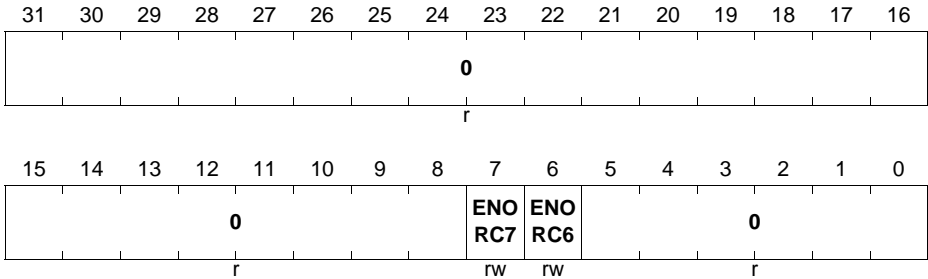
Field	Bits	Type	Description
RESULT	[9:0]	rh	Result of the DTS Measurement This bit field shows the result of the DTS measurement. The value given is directly related to the die temperature.
RDY	14	rh	Sensor Ready Status This bit indicate the DTS is ready or not. 0 _B The DTS is not ready 1 _B The DTS is ready
BUSY	15	rh	Sensor Busy Status This bit indicate if the DTS is currently busy. If the sensor is busy a measurement is still running and the result should not be used. 0 _B not busy 1 _B busy
0	[13:10], [31:16]	r	Reserved

GOORCEN

Enable register for out-of-range comparators of group 0 of analog input channels.

G0ORCEN

Out of Range Comparator Enable Register 0 (00A0_H) **Reset Value: 0000 0000_H**



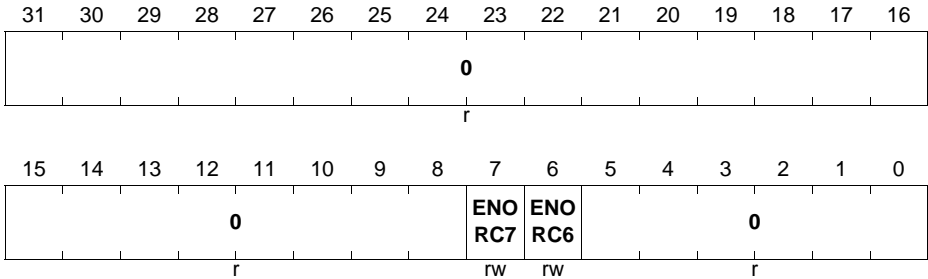
Field	Bits	Type	Description
ENORC6	6	rw	Enable Out of Range Comparator, Channel 6 Each bit (when set) enables the out of range comparator of the associated channel 0 _B Disabled 1 _B Enabled
ENORC7	7	rw	Enable Out of Range Comparator, Channel 7 Each bit (when set) enables the out of range comparator of the associated channel 0 _B Disabled 1 _B Enabled
0	[5:0], [31:8]	r	Reserved returns 0 if read; should be written with 0;

G1ORCEN

Enable register for out-of-range comparators of group 1 of analog input channels.

G1ORCEN

Out of Range Comparator Enable Register 1 (00A4_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ENORC6	6	rw	Enable Out of Range Comparator, Channel 6 Each bit (when set) enables the out of range comparator of the associated channel 0 _B Disabled 1 _B Enabled
ENORC7	7	rw	Enable Out of Range Comparator, Channel 7 Each bit (when set) enables the out of range comparator of the associated channel 0 _B Disabled 1 _B Enabled
0	[5:0], [31:8]	r	Reserved Read as 0; should be written with 0.

SDMMC_CON

SDMMC Write Protection and Card Detection Control

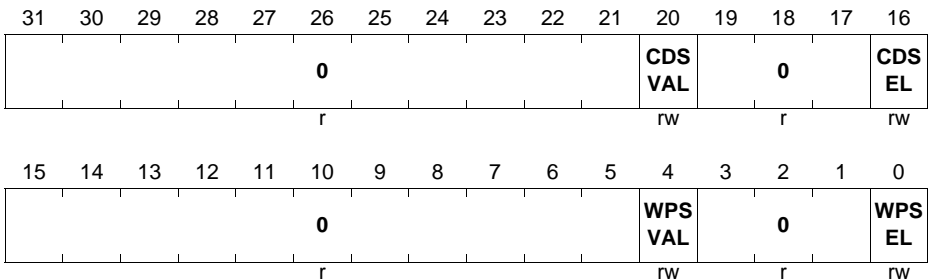
System Control Unit (SCU)

SDMMC_CON

SDMMC Configuration

(00B4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
WPSEL	0	rw	SDMMC Write Protection Input Multiplexer Control This bit field controls SDMMC write protection selection input. 0 _B P1.1 input pin selected 1 _B Software bit WPSVAL is selected
WPSVAL	4	rw	SDMMC Write Protect Software Control This bit field enables software control of the SDMMC write protection. 0 _B No write protection 1 _B Write protection active
CDSSEL	16	rw	SDMMC Card Detection Control This bit field controls SDMMC card detection. 0 _B P1.10 input pin selected 1 _B Software bit CDSVAL is selected
CDSVAL	20	rw	SDMMC Write Protect Software Control This bit field enables software control of the SDMMC card detection. 0 _B No card detected 1 _B Card detected
0	[3:1], [15:5], [19:17], [31:21]	r	Reserved Read as 0; should be written with 0.

System Control Unit (SCU)

MIRRSTS

Mirror status register for control of communication between SCU and other modules in hibernate domain. The bitfields of the register indicate that a corresponding register of the hibernate domain is ready to accept a write, or, that the communication interface is busy with executing the previous to the register.

MIRRSTS

Mirror Write Status Register

(00C4_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			0				0	0	0	0	0	0	0		
			r				r	r	r	r	r	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC _CL RSR	RTC _MS KSR	RMX	RTC _TIM 1	RTC _TIM 0	RTC _ATI M1	RTC _ATI M0	RTC _CT R	OSC ULC TRL	0	OSC SICT RL	0	HDC R	HDS ET	HDC LR	0
rh	rh	rh	rh	rh	rh	rh	rh	rh	r	rh	r	rh	rh	rh	r

Field	Bits	Type	Function
HDCLR	1	rh	HDCLR Mirror Register Write Status 0 _B Ready 1 _B Busy
HDSET	2	rh	HDSET Mirror Register Write Status 0 _B Ready 1 _B Busy
HDCR	3	rh	HDCR Mirror Register Write Status 0 _B Ready 1 _B Busy
OSCSICTRL	5	rh	OSCSICTRL Mirror Register Write Status 0 _B Ready 1 _B Busy
OSCULCTRL	7	rh	OSCULCTRL Mirror Register Write Status 0 _B Ready 1 _B Busy
RTC_CTR	8	rh	RTC CTR Mirror Register Write Status 0 _B Ready 1 _B Busy

System Control Unit (SCU)

Field	Bits	Type	Function
RTC_ATIM0	9	rh	RTC ATIM0 Mirror Register Write Status 0 _B Ready 1 _B Busy
RTC_ATIM1	10	rh	RTC ATIM1 Mirror Register Write Status 0 _B Ready 1 _B Busy
RTC_TIM0	11	rh	RTC TIM0 Mirror Register Write Status 0 _B Ready 1 _B Busy
RTC_TIM1	12	rh	RTC TIM1 Mirror Register Write Status 0 _B Ready 1 _B Busy
RMX	13	rh	Retention Memory Access Register Update Status This fields indicates status of retention memory update from RMDATA register to Hibernate domain retention memory or from Hibernate domain to RMDATA 0 _B Ready 1 _B Busy
RTC_MSKSR	14	rh	RTC MSKSSR Mirror Register Write Status 0 _B Ready 1 _B Busy
RTC_CLRSR	15	rh	RTC CLRSR Mirror Register Write Status 0 _B Ready 1 _B Busy
0	0,4,6, [17:16], 18, 19, [21:20], 22, [24:23], [31:25]	r	Reserved

RMACR

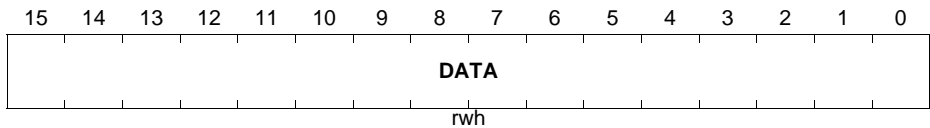
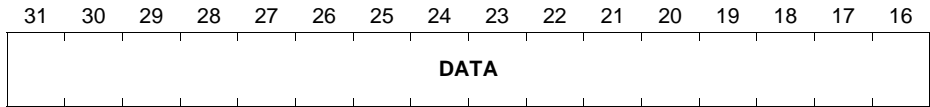
Access control to retention memory in hibernate domain.

System Control Unit (SCU)

RMDATA

Retention Memory Access Data Register (00CC_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Function
DATA	[31:0]	rwh	Hibernate Retention Memory Data This field data of selected of Retention Memory using address. The address of 0-15 is selected with RMACR register.

PEEN

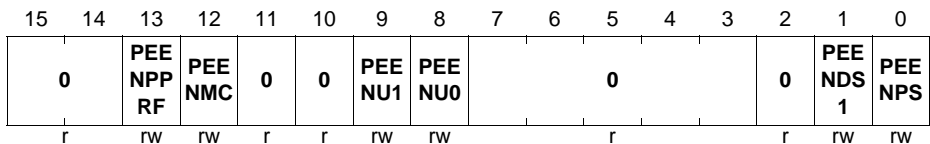
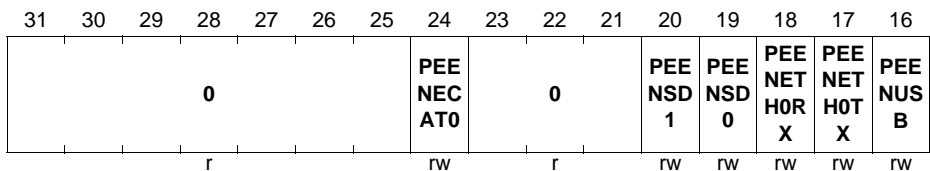
The following register enables parity check mechanism on peripheral modules.

PEEN

Parity Error Enable Register

(013C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PEENPS	0	rw	Parity Error Enable for PSRAM 0 _B Disabled 1 _B Enabled
PEENDS1	1	rw	Parity Error Enable for DSRAM1 0 _B Disabled 1 _B Enabled
PEENU0	8	rw	Parity Error Enable for USIC0 Memory 0 _B Disabled 1 _B Enabled
PEENU1	9	rw	Parity Error Enable for USIC1 Memory 0 _B Disabled 1 _B Enabled
PEENMC	12	rw	Parity Error Enable for MultiCAN Memory 0 _B Disabled 1 _B Enabled
PEENPPRF	13	rw	Parity Error Enable for PMU Prefetch Memory 0 _B Disabled 1 _B Enabled
PEENUSB	16	rw	Parity Error Enable for USB Memory 0 _B Disabled 1 _B Enabled
PEENETH0TX	17	rw	Parity Error Enable for ETH TX Memory 0 _B Disabled 1 _B Enabled
PEENETH0RX	18	rw	Parity Error Enable for ETH RX Memory 0 _B Disabled 1 _B Enabled
PEENSD0	19	rw	Parity Error Enable for SDMMC Memory 0 0 _B Disabled 1 _B Enabled
PEENSD1	20	rw	Parity Error Enable for SDMMC Memory 1 0 _B Disabled 1 _B Enabled

System Control Unit (SCU)

Field	Bits	Type	Description
PENECAT0	24	rw	Parity Error Enable for ECAT0 Memory 0 _B Disabled 1 _B Enabled <i>Note: For ECAT0 parity proper function the clock f_{CPU} must be equal or higher than 50 MHz</i>
0	2, [7:3], 10, 11, [15:14], [23:21], [31:25]	r	Reserved Should be written with 0.

MCHKCON

The following register enables the functional parity check mechanism for testing purpose. MCHKCON register is used to support access to parity bits of SRAM modules for various types of peripherals. The SRAM modules providing direct access natively need to be selected in order to enable direct write to parity bits using **PMTPR** register.

MCHKCON

Memory Checking Control Register (0140_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							SEL ECA T0	0			SEL SD1	SEL SD0	SEL ETH ORX	SEL ETH OTX	SEL USB
r							rw	r			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		PPR FDR A	MCA NDR A	0	0	USIC 1DR A	USIC 0DR A	0				0	SEL DS1	SEL PS	
r		rw	rw	r	r	rw	rw	r				r	rw	rw	

Field	Bits	Type	Description
SELPS	0	rw	Select Memory Check for PSRAM 0 _B Not selected 1 _B Selected

System Control Unit (SCU)

Field	Bits	Type	Description
SELDS1	1	rw	Select Memory Check for DSRAM1 0 _B Not selected 1 _B Selected
USIC0DRA	8	rw	Select Memory Check for USIC0 0 _B Not selected 1 _B Selected
USIC1DRA	9	rw	Select Memory Check for USIC1 0 _B Not selected 1 _B Selected
MCANDRA	12	rw	Select Memory Check for MultiCAN 0 _B Not selected 1 _B Selected
PPRFDRA	13	rw	Select Memory Check for PMU 0 _B Not selected 1 _B Selected
SELUSB	16	rw	Select Memory Check for USB SRAM 0 _B Not selected 1 _B Selected
SELETH0TX	17	rw	Select Memory Check for ETH0 TX SRAM 0 _B Not selected 1 _B Selected
SELETH0RX	18	rw	Select Memory Check for ETH0 RX SRAM 0 _B Not selected 1 _B Selected
SELSD0	19	rw	Select Memory Check for SDMMC SRAM 0 0 _B Not selected 1 _B Selected
SELSD1	20	rw	Select Memory Check for SDMMC SRAM 1 0 _B Not selected 1 _B Selected
SELECAT0	24	rw	Select Memory Check for ECAT0 SRAM 1 0 _B Not selected 1 _B Selected

System Control Unit (SCU)

Field	Bits	Type	Description
PETEU0	8	rw	Parity Error Trap Enable for USIC0 Memory 0 _B Disabled 1 _B Enabled
PETEU1	9	rw	Parity Error Trap Enable for USIC1 Memory 0 _B Disabled 1 _B Enabled
PETEMC	12	rw	Parity Error Trap Enable for MultiCAN Memory 0 _B Disabled 1 _B Enabled
PETEPPRF	13	rw	Parity Error Trap Enable for PMU Prefetch Memory 0 _B Disabled 1 _B Enabled
PETEUSB	16	rw	Parity Error Trap Enable for USB Memory 0 _B Disabled 1 _B Enabled
PETEETH0TX	17	rw	Parity Error Trap Enable for ETH 0TX Memory 0 _B Disabled 1 _B Enabled
PETEETH0RX	18	rw	Parity Error Trap Enable for ETH0 RX Memory 0 _B Disabled 1 _B Enabled
PETESD0	19	rw	Parity Error Trap Enable for SDMMC SRAM 0 Memory 0 _B Disabled 1 _B Enabled
PETESD1	20	rw	Parity Error Trap Enable for SDMMC SRAM 1 Memory 0 _B Disabled 1 _B Enabled
PETEECAT0	24	rw	Parity Error Trap Enable for ECAT0 SRAM Memory 0 _B Disabled 1 _B Enabled

System Control Unit (SCU)

Field	Bits	Type	Description
0	2, [7:3], 10, 11, [15:14], [23:21], [31:25]	r	Reserved Should be written with 0.

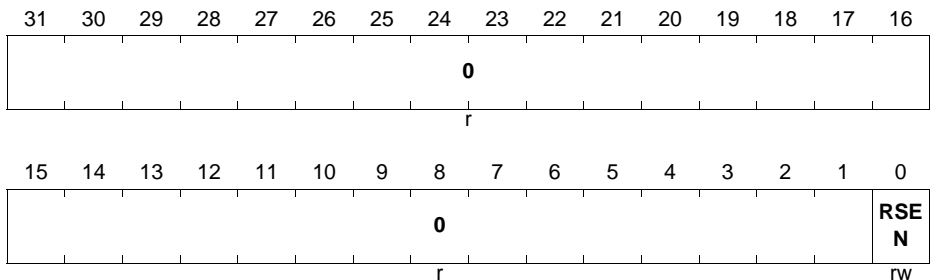
PERSTEN

The following register enables reset upon parity error flag from the functional parity check mechanism indicated in **PEFLAG** register.

PERSTEN

Parity Error Reset Enable Register (0148_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RSEN	0	rw	System Reset Enable upon Parity Error Trap 0 _B Reset request disabled 1 _B Reset request enabled
0	[31:1]	r	Reserved Read as 0; should be written with 0.

PEFLAG

The PEFLAG register controls the functional parity check mechanism.

The register bits can only get set by corresponding parity error assertion if enabled and can only be cleared via software. Writing a zero to this bit does not change the content. Writing a one to this bit does clear the bit.

System Control Unit (SCU)

PEFLAG

Parity Error Flag Register

(0150_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							PEE CAT 0	0			PES D1	PES D0	PEE TH0 RX	PEE TH0 TX	PEU SB
r							rwh	r			rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		PEF PPR F	PEF MC	0	0	PEF U1	PEF U0	0				0		PEF DS1	PEF PS
r		rwh	rwh	r	r	rwh	rwh	r				r		rwh	rwh

Field	Bits	Type	Description
PEFPS	0	rwh	Parity Error Flag for PSRAM 0 _B No parity error detected 1 _B Parity error detected
PEFDS1	1	rwh	Parity Error Flag for DSRAM1 0 _B No parity error detected 1 _B Parity error detected
PEFU0	8	rwh	Parity Error Flag for USIC0 Memory 0 _B No parity error detected 1 _B Parity error detected
PEFU1	9	rwh	Parity Error Flag for USIC1 Memory 0 _B No parity error detected 1 _B Parity error detected
PEFMC	12	rwh	Parity Error Flag for MultiCAN Memory 0 _B No parity error detected 1 _B Parity error detected
PEFPPRF	13	rwh	Parity Error Flag for PMU Prefetch Memory 0 _B No parity error detected 1 _B Parity error detected
PEUSB	16	rwh	Parity Error Flag for USB Memory 0 _B No parity error detected 1 _B Parity error detected

System Control Unit (SCU)

Field	Bits	Type	Description
PEETH0TX	17	rwh	Parity Error Flag for ETH TX Memory 0 _B No parity error detected 1 _B Parity error detected
PEETH0RX	18	rwh	Parity Error Flag for ETH RX Memory 0 _B No parity error detected 1 _B Parity error detected
PESD0	19	rwh	Parity Error Flag for SDMMC Memory 0 0 _B No parity error detected 1 _B Parity error detected
PESD1	20	rwh	Parity Error Flag for SDMMC Memory 1 0 _B No parity error detected 1 _B Parity error detected
PEECAT0	24	rwh	Parity Error Flag for ECAT0 Memory 0 _B No parity error detected 1 _B Parity error detected
0	2, [7:3], 10, 11, [15:14], [23:21], [31:25]	r	Reserved Should be written with 0.

PMPTR

The following register provides direct access to parity bits of a selected module.

The width and therefore the valid bits in register **PMPTR** is listed in [Table 11-12](#).

Table 11-12 Memory Parity Bus Widths

Memory Instance	Number of Parity Bits	Valid Bits in PWR/PRD
Program SRAM (PSRAM)	4	PWR[3:0]/PRD[11:8]
System SRAM (DSRAM1)	4	PWR[3:0]/PRD[11:8]
USIC 0 Buffer (U0)	1	PWR[0]/PRD[8]
USIC 1 Buffer (U1)	1	PWR[0]/PRD[8]
MultiCAN Buffer (MC)	1	PWR[0]/PRD[8]
PMU Prefetch Buffer (PPRF)	4	PWR[3:0]/PRD[11:8]

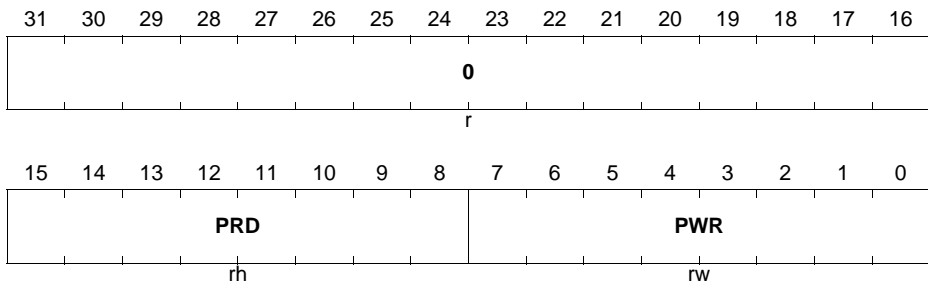
Table 11-12 Memory Parity Bus Widths (cont'd)

Memory Instance	Number of Parity Bits	Valid Bits in PWR/PRD
USB Buffer (USB)	1	PWR[0]/PRD[8]
ETH0 TX Buffer (ETH0TX)	1	PWR[0]/PRD[8]
ETH0 RX Buffer (ETH0RX)	1	PWR[0]/PRD[8]
SDMMC Buffer 0 (SD0)	1	PWR[0]/PRD[8]
SDMMC Buffer 1 (SD1)	1	PWR[0]/PRD[8]
ECAT0 Buffer (ECAT0)	1	PWR[0]/PRD[8]

PMTPR

Parity Memory Test Pattern Register (0154_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PRD	[15:8]	rh	Parity Read Values for Memory Test For each byte of a memory module the parity bits generated during the most recent read access are indicated here.
PWR	[7:0]	rw	Parity Write Values for Memory Test For each byte of a memory module the parity bits corresponding to the next write access are stored here.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

System Control Unit (SCU)

PMTSR

This register selects parity test output from a memory instance that will be reflected in PRD bit field of **PMTPR** register.

*Note: Only one bit shall be set at the same time in register **PMTPR**. Otherwise the result of the parity software test is unpredictable.*

PMTSR

Parity Memory Test Select Register (0158_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							MTE CAT 0	0			MTS D1	MTS D0	MTE TH0 RX	MTE TH0 TX	MTU SB
r							rw	r			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		MTE PPR F	MTE MC	0	0	MTE U1	MTE U0	0				0	MTE NDS 1	MTE NPS	
r		rwh	rwh	r	r	rwh	rwh	r				r	rw	rw	

Field	Bits	Type	Description
MTENPS	0	rw	Test Enable Control for PSRAM 0 _B Standard operation 1 _B Parity bits under test
MTENDS1	1	rw	Test Enable Control for DSRAM1 0 _B Standard operation 1 _B Parity bits under test
MTEU0	8	rwh	Test Enable Control for USIC0 Memory 0 _B Standard operation 1 _B Parity bits under test
MTEU1	9	rwh	Test Enable Control for USIC1 Memory 0 _B Standard operation 1 _B Parity bits under test
MTEMC	12	rwh	Test Enable Control for MultiCAN Memory 0 _B Standard operation 1 _B Parity bits under test
MTEPPRF	13	rwh	Test Enable Control for PMU Prefetch Memory 0 _B Standard operation 1 _B Parity bits under test

System Control Unit (SCU)

Field	Bits	Type	Description
MTUSB	16	rw	Test Enable Control for USB Memory 0 _B Standard operation 1 _B Parity bits under test
MTETH0TX	17	rw	Test Enable Control for ETH TX Memory 0 _B Standard operation 1 _B Parity bits under test
MTETH0RX	18	rw	Test Enable Control for ETH RX Memory 0 _B Standard operation 1 _B Parity bits under test
MTSD0	19	rw	Test Enable Control for SDMMC Memory 0 0 _B Standard operation 1 _B Parity bits under test
MTSD1	20	rw	Test Enable Control for SDMMC Memory 1 0 _B Standard operation 1 _B Parity bits under test
MTECAT0	24	rw	Test Enable Control for ECAT0 Memory 0 _B Standard operation 1 _B Parity bits under test
0	2, [7:3], 10, 11, [15:14], [23:21], [31:25]	r	Reserved Should be written with 0.

TRAPSTAT

This register contains the status flags for all trap request trigger sources of the SCU. A trap flag is set when a corresponding emergency event occurs. Trap mechanism supports testing and debug of these status bits by software using registers **TRAPSET** and **TRAPCLR**. This register reflects masking with **TRAPDIS** register.

System Control Unit (SCU)

TRAPSTAT

Trap Status Register

(0160_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															ECA TOR ST
r															rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0			0		BWE RR1 T	BWE RR0 T	ULP WDG T	BRW NT	PET	UVC OLC KT	SVC OLC KT	0	SOS CWD GT
r	r	r			r		rh	rh	rh	rh	rh	rh	rh	r	rh

Field	Bits	Type	Description
SOSCWDGT	0	rh	OSC_HP Oscillator Watchdog Trap Status 0 _B No pending trap request 1 _B Pending trap request
SVCOLCKT	2	rh	System VCO Lock Trap Status 0 _B No pending trap request 1 _B Pending trap request
UVCOLCKT	3	rh	USB VCO Lock Trap Status 0 _B No pending trap request 1 _B Pending trap request
PET	4	rh	Parity Error Trap Status 0 _B No pending trap request 1 _B Pending trap request
BRWNT	5	rh	Brown Out Trap Status 0 _B No pending trap request 1 _B Pending trap request
ULPWDGT	6	rh	OSC_ULP Oscillator Watchdog Trap Status 0 _B No pending trap request 1 _B Pending trap request
BWERR0T	7	rh	Peripheral Bridge 0 Trap Status This trap flags error responses for buffered write operations on the Peripheral Bridge 0 0 _B No pending trap request 1 _B Pending trap request

System Control Unit (SCU)

Field	Bits	Type	Description
BWERR1T	8	rh	Peripheral Bridge 1 Trap Status This trap flags error responses for buffered write operations on the Peripheral Bridge 1 0 _B No pending trap request 1 _B Pending trap request
ECATORST	16	rh	EtherCat Reset 0 Trap Status 0 _B No effect 1 _B Pending trap request
0	1, [11:9], [13:12], 15,14, [31:17]	r	Reserved Read as 0.

TRAPRAW

This register contains the status flags for all trap request trigger sources of the SCU before masking with **TRAPDIS**.

A trap flag is set when a corresponding emergency event occurs. For setting and clearing of these status bits by software see registers **TRAPSET** and **TRAPCLR**, respectively.

TRAPRAW

Trap Raw Status Register

(0164_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															ECA TOR ST
															rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0			0		BWE RR1 T	BWE RR0 T	ULP WDG T	BRW NT	PET	UVC OLC KT	SVC OLC KT	0	SOS CWD GT
r	r	r			r		rh	rh	rh	rh	rh	rh	rh	r	rh

Field	Bits	Type	Description
SOSCWDGT	0	rh	OSC_HP Oscillator Watchdog Trap Raw Status 0 _B No pending trap request 1 _B Pending trap request
SVCOLCKT	2	rh	System VCO Lock Trap Raw Status 0 _B No pending trap request 1 _B Pending trap request
UVCOLCKT	3	rh	USB VCO Lock Trap Raw Status 0 _B No pending trap request 1 _B Pending trap request
PET	4	rh	Parity Error Trap Raw Status 0 _B No pending trap request 1 _B Pending trap request
BRWNT	5	rh	Brown Out Trap Raw Status 0 _B No pending trap request 1 _B Pending trap request
ULPWDGT	6	rh	OSC_ULP Oscillator Watchdog Trap Raw Status 0 _B No pending trap request 1 _B Pending trap request
BWERR0T	7	rh	Peripheral Bridge 0 Trap Raw Status 0 _B No pending trap request 1 _B Pending trap request
BWERR1T	8	rh	Peripheral Bridge 1 Trap Raw Status 0 _B No pending trap request 1 _B Pending trap request
ECAT0RST	16	rh	EtherCat Reset 0 Trap Raw Status 0 _B No pending trap request 1 _B Pending trap request
0	1, [13:12], [11:9], 15,14, [31:17]	r	Reserved Read as 0.

TRAPDIS

Disable corresponding traps.

System Control Unit (SCU)

TRAPDIS

Trap Disable Register

(0168_H)

Reset Value:0001 01FD_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															ECA TOR ST
r															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	BWE RR1 T	BWE RR0 T	ULP WDG T	BRW NT	PET	UVC OLC KT	SVC OLC KT	0	SOS CWD GT
r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	r	rw

Field	Bits	Type	Description
SOSCWDGT	0	rw	OSC_HP Oscillator Watchdog Trap Disable 0 _B Trap request enabled 1 _B Trap request disabled
SVCOLCKT	2	rw	System VCO Lock Trap Disable 0 _B Trap request enabled 1 _B Trap request disabled
UVCOLCKT	3	rw	USB VCO Lock Trap Disable 0 _B Trap request enabled 1 _B Trap request disabled
PET	4	rw	Parity Error Trap Disable 0 _B Trap request enabled 1 _B Trap request disabled
BRWNT	5	rw	Brown Out Trap Disable 0 _B Trap request enabled 1 _B Trap request disabled
ULPWDGT	6	rw	OSC_ULP Oscillator Watchdog Trap Disable 0 _B Trap request enabled 1 _B Trap request disabled
BWERR0T	7	rw	Peripheral Bridge 0 Trap Disable 0 _B Trap request enabled 1 _B Trap request disabled

System Control Unit (SCU)

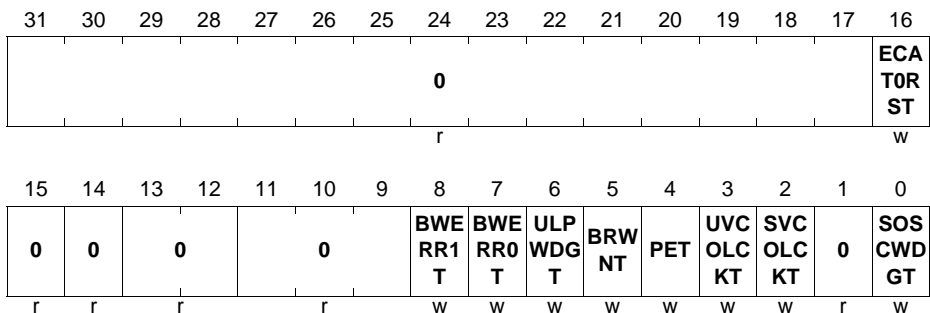
Field	Bits	Type	Description
BWERR1T	8	rw	Peripheral Bridge 1 Trap Disable 0 _B Trap request enabled 1 _B Trap request disabled
ECAT0RST	16	rw	EtherCat Reset 0 Trap Disable 0 _B Trap request enabled 1 _B Trap request disabled
0	1, [13:12], [11:9], 15,14, [31:17]	r	Reserved Read as 0; should be written with 0.

TRAPCLR

This register contains the software clear control for the trap status flags in register **TRAPRAW** and **TRAPSTAT**.

TRAPCLR

Trap Clear Register (016C_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
SOSCWDGT	0	w	OSC_HP Oscillator Watchdog Trap Clear 0 _B No effect 1 _B Clear trap request
SVCOLCKT	2	w	System VCO Lock Trap Clear 0 _B No effect 1 _B Clear trap request

System Control Unit (SCU)

Field	Bits	Type	Description
UVCOLCKT	3	w	USB VCO Lock Trap Clear 0 _B No effect 1 _B Clear trap request
PET	4	w	Parity Error Trap Clear 0 _B No effect 1 _B Clear trap request
BRWNT	5	w	Brown Out Trap Clear 0 _B No effect 1 _B Clear trap request
ULPWDGT	6	w	OSC_ULP Oscillator Watchdog Trap Clear 0 _B No effect 1 _B Clear trap request
BWERR0T	7	w	Peripheral Bridge 0 Trap Clear 0 _B No effect 1 _B Clear trap request
BWERR1T	8	w	Peripheral Bridge 1 Trap Clear 0 _B No effect 1 _B Clear trap request
ECAT0RST	16	w	EtherCat Reset 0 Trap Clear 0 _B No effect 1 _B Set trap request
0	1, [13:12], [11:9], 15,14, [31:17]	r	Reserved Read as 0; should be written with 0.

TRAPSET

This register contains the software set control for the trap status flags in register TRAPRAW.

System Control Unit (SCU)

TRAPSET

Trap Set Register

(0170_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															ECA TOR ST
r															w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0				BWE RR1 T	BWE RR0 T	ULP WDT	BRW NT	PET	UVC OLC KT	SVC OLC KT	0	SOS CWD GT
r	r	r	r				w	w	w	w	w	w	w	r	w

Field	Bits	Type	Description
SOSCWDGT	0	w	OSC_HP Oscillator Watchdog Trap Set 0 _B No effect 1 _B Set trap request
SVCOLCKT	2	w	System VCO Lock Trap Set 0 _B No effect 1 _B Set trap request
UVCOLCKT	3	w	USB VCO Lock Trap Set 0 _B No effect 1 _B Set trap request
PET	4	w	Parity Error Trap Set 0 _B No effect 1 _B Set trap request
BRWNT	5	w	Brown Out Trap Set 0 _B No effect 1 _B Set trap request
ULPWDT	6	w	OSC_ULP Oscillator Watchdog Trap Set 0 _B No effect 1 _B Set trap request
BWERR0T	7	w	Peripheral Bridge 0 Trap Set 0 _B No effect 1 _B Set trap request

System Control Unit (SCU)

Field	Bits	Type	Description
BWERR1T	8	w	Peripheral Bridge 1 Trap Set 0 _B No effect 1 _B Set trap request
ECAT0RST	16	w	EtherCat Reset 0 Trap Set 0 _B No effect 1 _B Set trap request
0	1, [13:12], [11:9], 15,14, [31:17]	r	Reserved Read as 0; should be written with 0.

ECAT0_CON

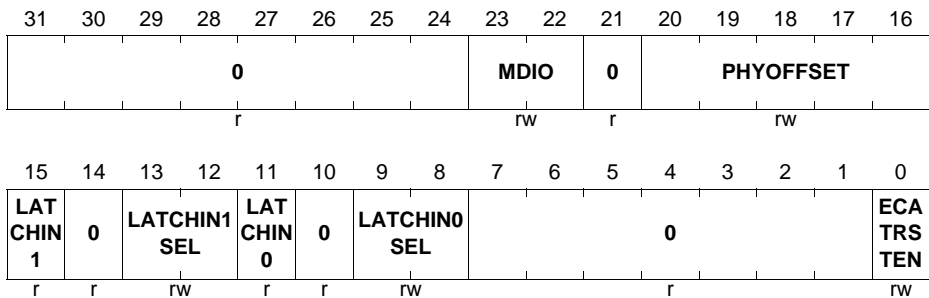
EtherCAT 0 control register.

ECAT0_CON

EtherCAT 0 Control

(01B0_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ECATRSTEN	0	rw	Enable EtherCAT Reset Request By setting the ECATRSTEN the Master can trigger a reset of the XMC4300. 0 _B Reset request by EtherCAT disabled 1 _B Reset request by EtherCAT enabled

System Control Unit (SCU)

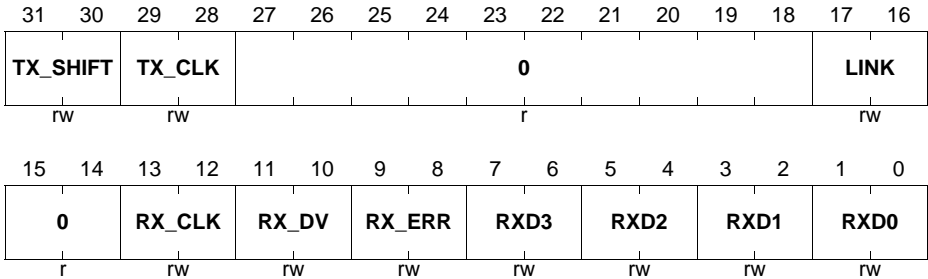
Field	Bits	Type	Description
LATCHIN0SEL	[9:8]	rw	LATCHIN0 Input Select This bit field selects the input position of the LATCHIN0 signal. 00 _B Data input LATCHIN0A is selected 01 _B Data input LATCHIN0B is selected 10 _B Data input LATCHIN0C is selected 11 _B Data input LATCHIN0D is selected
LATCHIN0	11	r	EtherCAT LATCH_IN0 Input Signal
LATCHIN1SEL	[13:12]	rw	LATCHIN1 Input Select This bit field selects the input position of the LATCHIN1 signal. 00 _B Data input LATCHIN1A is selected 01 _B Data input LATCHIN1B is selected 10 _B Data input LATCHIN1C is selected 11 _B Data input LATCHIN1D is selected
LATCHIN1	15	r	EtherCAT LATCH_IN1 Input Signal
PHYOFFSET	[20:16]	rw	Ethernet PHY Address Offset Defines the offset of the PHY address offset (consecutive PHY addresses, address of port 0)
MDIO	[23:22]	rw	MDIO Input Select This bit field selects the input position of the MDI signal. 00 _B Data input MDIA is selected 01 _B Data input MDIB is selected 10 _B Data input MDIC is selected 11 _B Data input MDID is selected
0	[7:1], 10, 14, 21, [31:24]	r	Reserved Read as 0; should be written with 0.

ECAT0_CONP0

EtherCAT 0 port 0 control register.

EtherCAT 0 Port 0 Control Register (01B4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXD0	[1:0]	rw	PORT0 Receive Input 0 Select This bit field indicates the receive input position of the RXD0 signal. 00 _B Data input RXD0A is selected 01 _B Data input RXD0B is selected 10 _B Data input RXD0C is selected 11 _B Data input RXD0D is selected
RXD1	[3:2]	rw	Port0 Receive Input 1 Select This bit field indicates the receive input position of the RXD1 signal. 00 _B Data input RXD1A is selected 01 _B Data input RXD1B is selected 10 _B Data input RXD1C is selected 11 _B Data input RXD1D is selected
RXD2	[5:4]	rw	Port0 Receive Input 2 Select This bit field indicates the receive input position of the RXD2 signal. 00 _B Data input RXD2A is selected 01 _B Data input RXD2B is selected 10 _B Data input RXD2C is selected 11 _B Data input RXD2D is selected

System Control Unit (SCU)

Field	Bits	Type	Description
RXD3	[7:6]	rw	Port0 Receive Input 3 Select This bit field indicates the receive input position of the RXD3 signal. 00 _B Data input RXD3A is selected 01 _B Data input RXD3B is selected 10 _B Data input RXD3C is selected 11 _B Data input RXD3D is selected
RX_ERR	[9:8]	rw	Port0 MII RX ERROR Input Select This bit field indicates the receive input position of the RX EER input signal. 00 _B Data input RX_ERRA is selected 01 _B Data input RX_ERRB is selected 10 _B Data input RX_ERRC is selected 11 _B Data input RX_ERRD is selected
RX_DV	[11:10]	rw	Port0 MII RX DV Input Select This bit field indicates the receive input position of the MII RX DV input signal. 00 _B Data input RX_DVA is selected 01 _B Data input RX_DVB is selected 10 _B Data input RX_DVC is selected 11 _B Data input RX_DVD is selected
RX_CLK	[13:12]	rw	Port0 MII RX Clock Input Select This bit field indicates the receive input position of the MII RX Clock input signal select . 00 _B Clock input RX_CLKA 01 _B Clock input RX_CLKB 10 _B Clock input RX_CLKC 11 _B Clock input RX_CLKD
LINK	[17:16]	rw	Port0 PHY Link Input Select This bit field indicates the receive input position of the PHY Link input signal select . 00 _B PHY LINKA 01 _B PHY LINKB 10 _B PHY LINKC 11 _B PHY LINKD

System Control Unit (SCU)

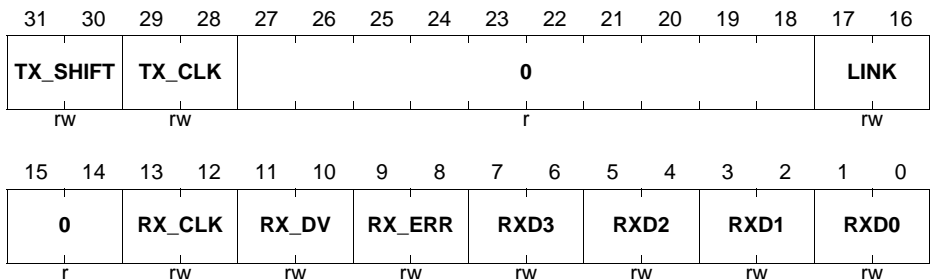
Field	Bits	Type	Description
TX_CLK	[29:28]	rw	Port0 MII TX Clock Input Select This bit field indicates the receive input position of the MII TX Clock input signal select . 00 _B Clock input TX_CLKA 01 _B Clock input TX_CLKB 10 _B Clock input TX_CLKC 11 _B Clock input TX_CLKD
TX_SHIFT	[31:30]	rw	Port0 Manual TX Shift configuration This bit field defines the additional TX signal delay for the Port. 00 _B 0 ns 01 _B 10 ns 10 _B 20 ns 11 _B 30 ns <i>Note: TX Shift availability depends on the ESC configuration.</i>
0	[15:14], [27:18]	r	Reserved Read as 0; should be written with 0.

ECAT0_CONP1

EtherCAT 0 port 1 control register.

EtherCAT 0 Port 1 Control Register (01B8_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXD0	[1:0]	rw	<p>Port1 Receive Input 0 Select This bit field indicates the receive input position of the RXD0 signal.</p> <p>00_B Data input RXD0A is selected 01_B Data input RXD0B is selected 10_B Data input RXD0C is selected 11_B Data input RXD0D is selected</p>
RXD1	[3:2]	rw	<p>Port1 Receive Input 1 Select This bit field indicates the receive input position of the RXD1 signal.</p> <p>00_B Data input RXD1A is selected 01_B Data input RXD1B is selected 10_B Data input RXD1C is selected 11_B Data input RXD1D is selected</p>
RXD2	[5:4]	rw	<p>Port1 Receive Input 2 Select This bit field indicates the receive input position of the RXD2 signal.</p> <p>00_B Data input RXD2A is selected 01_B Data input RXD2B is selected 10_B Data input RXD2C is selected 11_B Data input RXD2D is selected</p>
RXD3	[7:6]	rw	<p>Port1 Receive Input 3 Select This bit field indicates the receive input position of the RXD3 signal.</p> <p>00_B Data input RXD3A is selected 01_B Data input RXD3B is selected 10_B Data input RXD3C is selected 11_B Data input RXD3D is selected</p>
RX_ERR	[9:8]	rw	<p>Port1 MII RX ERROR Input Select This bit field indicates the receive input position of the RX EER input signal.</p> <p>00_B Data input RX_ERRA is selected 01_B Data input RX_ERRB is selected 10_B Data input RX_ERRC is selected 11_B Data input RX_ERRD is selected</p>

System Control Unit (SCU)

Field	Bits	Type	Description
RX_DV	[11:10]	rw	Port1 MII RX DV Input Select This bit field indicates the receive input position of the MII RX DV input signal. 00 _B Data input RX_DVA is selected 01 _B Data input RX_DVB is selected 10 _B Data input RX_DVC is selected 11 _B Data input RX_DVD is selected
RX_CLK	[13:12]	rw	Port1 MII RX Clock Input Select This bit field indicates the receive input position of the MII RX Clock input signal select . 00 _B Clock input RX_CLKA 01 _B Clock input RX_CLKB 10 _B Clock input RX_CLKC 11 _B Clock input RX_CLKD
LINK	[17:16]	rw	Port1 PHY Link Input Select This bit field indicates the receive input position of the PHY Link input signal select . 00 _B PHY LINKA 01 _B PHY LINKB 10 _B PHY LINKC 11 _B PHY LINKD
TX_CLK	[29:28]	rw	Port1 MII TX Clock Input Select This bit field indicates the receive input position of the MII TX Clock input signal select . 00 _B Clock input TX_CLKA 01 _B Clock input TX_CLKB 10 _B Clock input TX_CLKC 11 _B Clock input TX_CLKD
TX_SHIFT	[31:30]	rw	Port1 Manual TX Shift configuration This bit field defines the additional TX signal delay for the Port. 00 _B 0 ns 01 _B 10 ns 10 _B 20 ns 11 _B 30 ns <i>Note: TX Shift availability depends on the ESC configuration.</i>
0	[15:14], [27:18]	r	Reserved Read as 0; should be written with 0.

System Control Unit (SCU)

PWRSET

PCU Set Control Register

(0204_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0													USB PUW Q	USB OTG EN	USB PHY PDQ
r													w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														HIB	
r														w	

Field	Bits	Type	Description
HIB	0	w	Set Hibernate Domain Enable 0 _B No effect 1 _B Enable Hibernate domain
USBPHYPDQ	16	w	Set USB PHY Transceiver Disable 0 _B No effect 1 _B Active
USBOTGEN	17	w	Set USB On-The-Go Comparators Enable 0 _B No effect 1 _B Active
USBPUWQ	18	w	Set USB Weak Pull-Up at PADN Enable 0 _B No effect 1 _B Pull-up not active
0	[15:1], [31:19]	r	Reserved Read as 0; should be written with 0.

PWRCLR

Power control register. Write one to clear, writing zeros have no effect.

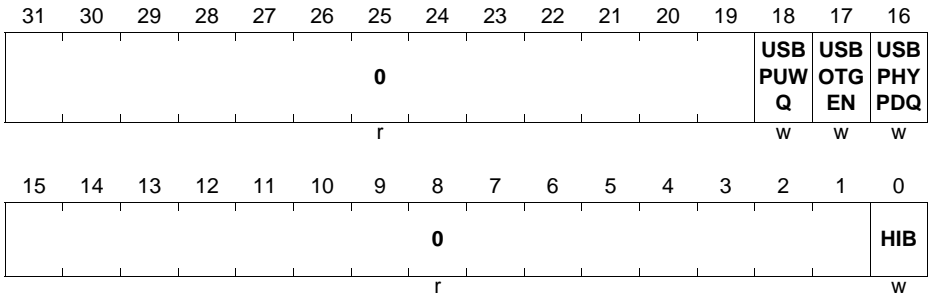
System Control Unit (SCU)

PWRCLR

PCU Clear Control Register

(0208_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
HIB	0	w	Clear Disable Hibernate Domain 0 _B No effect 1 _B Disable Hibernate domain
USBPHYPDQ	16	w	Clear USB PHY Transceiver Disable 0 _B No effect 1 _B Power-down
USBOTGEN	17	w	Clear USB On-The-Go Comparators Enable 0 _B No effect 1 _B Power-down
USBPUWQ	18	w	Clear USB Weak Pull-Up at PADN Enable 0 _B No effect 1 _B Pull-up active
0	[15:1], [31:19]	r	Reserved Read as 0; should be written with 0.

EVRSTAT

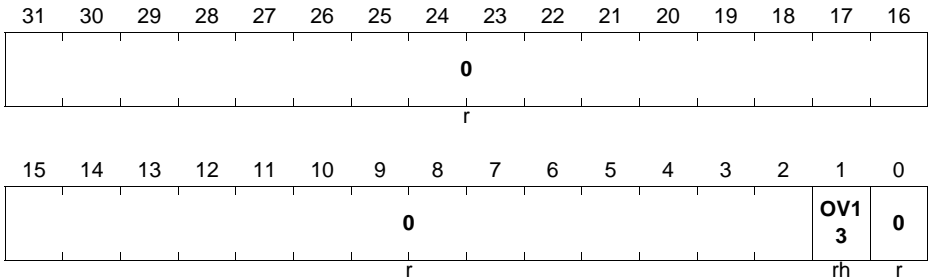
EVR status register.

EVRSTAT

EVR Status Register

(0210_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
OV13	1	rh	Regulator Overvoltage for 1.3 V 0 _B No overvoltage condition 1 _B Regulator is in overvoltage
0	0, [31:2]	r	Reserved

EVRVADCSTAT

Supply voltage monitor register. The actual voltage represented by the VADC13V and VADC33V can be calculated according to the formulas:

$$VADC13V = (V_{DCC} / LSB13V) + 1 \tag{11.7}$$

where LSB13V is 5.8 mV

$$VADC33V = (V_{DDP} / LSB33V) + 1 \tag{11.8}$$

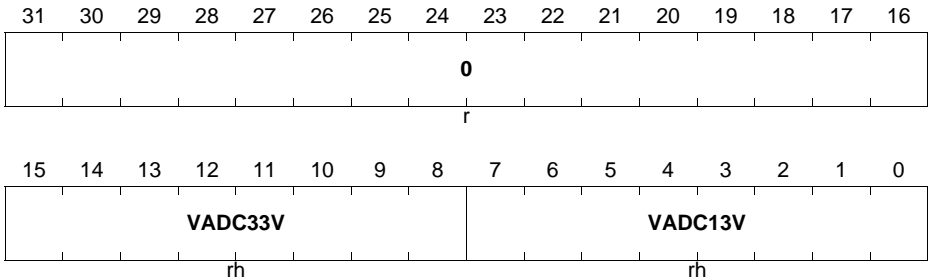
where LSB33V is 22.5 mV

EVRVADCSTAT

EVR VADC Status Register

(0214_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
VADC13V	[7:0]	rh	VADC 1.3 V Conversion Result This bit field contains the last conversion result of the VADC for the EVR13.
VADC33V	[15:8]	rh	VADC 3.3 V Conversion Result This bit field contains the last conversion result of the VADC for the EVR33. The value is used for brown-out detection
0	[31:16]	r	Reserved Read as 0.

PWRMON

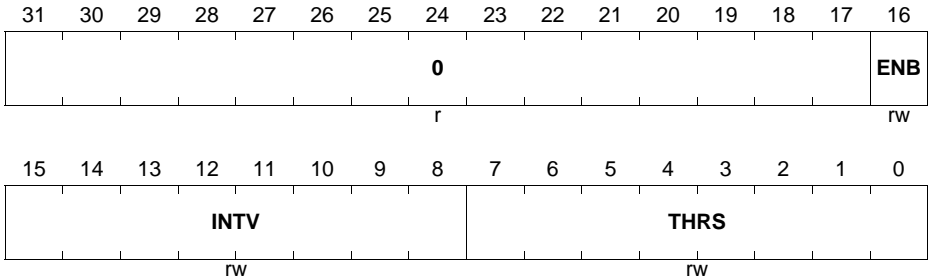
Power monitoring control register for brown-out detection.

PWRMON

Power Monitor Control

(022C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
THRS	[7:0]	rw	Threshold Threshold value for comparison to V_{DDP} for brown-out detection
INTV	[15:8]	rw	Interval Interval value for comparison to V_{DDP} expressed in cycles of system clock
ENB	16	rw	Enable Enable of comparison and interrupt generation
0	[31:17]	r	Reserved Read as 0; should be written with 0.

11.10.3 HCU Registers

HDSTAT

Hibernate domain status register

System Control Unit (SCU)

HDSTAT

Hibernate Domain Status Register (0300_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								0							
r								r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				0				0			HIBN OUT	ULP WDG	RTC EV	ENE V	EPE V
r				r				r			rh	rh	rh	rh	rh

Field	Bits	Type	Description
EPEV	0	rh	Wake-up Pin Event Positive Edge 0 _B Wake-up on positive edge pin event inactive 1 _B Wake-up on positive edge pin event active
ENEV	1	rh	Wake-up Pin Event Negative Edge 0 _B Wake-up on negative edge pin event inactive 1 _B Wake-up on negative edge pin event active
RTCEV	2	rh	RTC Event 0 _B Wake-up on RTC event inactive 1 _B Wake-up on RTC event active
ULPWDG	3	rh	ULP WDG Alarm Status 0 _B Watchdog alarm did not occur 1 _B Watchdog alarm occurred
HIBNOUT	4	rh	Hibernate Control Status 0 _B Hibernate not driven active to pads 1 _B Hibernate driven active to pads
0	[7:5], [13:8], [30:14], 31	r	Reserved

HDCLR

Hibernate domain clear status register. Write one to clear, writing zeros has no effect.

HDCLR

Hibernate Domain Status Clear Register (0304_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								0							
r								r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				0					0			ULP WDG	RTC EV	ENE V	EPE V
r				r					r			w	w	w	w

Field	Bits	Type	Description
EPEV	0	w	Wake-up Pin Event Positive Edge Clear 0 _B No effect 1 _B Clear wake-up event
ENEV	1	w	Wake-up Pin Event Negative Edge Clear 0 _B No effect 1 _B Clear wake-up event
RTCEV	2	w	RTC Event Clear 0 _B No effect 1 _B Clear wake-up event
ULPWDG	3	w	ULP WDG Alarm Clear 0 _B No effect 1 _B Clear watchdog alarm
0	[7:4], [13:8], [30:14], 31	r	Reserved Read as 0; should be written with 0.

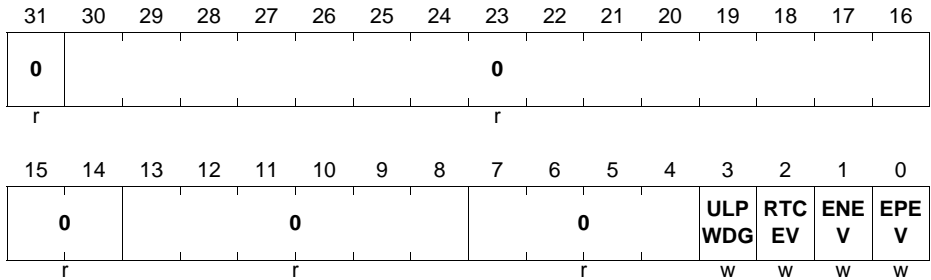
HDSET

Hibernate domain set status register. Write one to set, writing zeros has no effect.

HDSET

Hibernate Domain Status Set Register (0308_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
EPEV	0	w	Wake-up Pin Event Positive Edge Set 0 _B No effect 1 _B Set wake-up event
ENEV	1	w	Wake-up Pin Event Negative Edge Set 0 _B No effect 1 _B Set wake-up event
RTCEV	2	w	RTC Event Set 0 _B No effect 1 _B Set wake-up event
ULPWDG	3	w	ULP WDG Alarm Set 0 _B No effect 1 _B Set watchdog alarm
0	[7:4], [13:8], [30:14], 31	r	Reserved Read as 0; should be written with 0.

HDCR

Hibernate domain configuration register.

System Control Unit (SCU)

HDCR

Hibernate Domain Control Register (030C_H)

Reset Value: 000C 2000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		0						HIBIO1SEL				HIBIO0SEL			
r		r						rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		HIBIO1POL	HIBIO0POL	0	GPIOSEL	0	WKUPEL	STDBYSEL	RCS	0	HIB	ULPWDGEN	RTCE	WKPEN	WKPEP
r		rw	rw	r	rw	r	rw	rw	rw	r	rwh	rw	rw	rw	rw

Field	Bits	Type	Description
WKPEP	0	rw	Wake-Up on Pin Event Positive Edge Enable 0 _B Wake-up event disabled 1 _B Wake-up event enabled
WKPEN	1	rw	Wake-up on Pin Event Negative Edge Enable 0 _B Wake-up event disabled 1 _B Wake-up event enabled
RTCE	2	rw	Wake-up on RTC Event Enable 0 _B Wake-up event disabled 1 _B Wake-up event enabled
ULPWDGEN	3	rw	ULP WDG Alarm Enable 0 _B Wake-up event disabled 1 _B Wake-up event enabled
HIB	4	rwh	Hibernate Request Value Set 0 _B External hibernate request inactive 1 _B External hibernate request active <i>Note: This bit get automatically cleared by hardware upon occurrence of any wake-up event enabled in this register</i>
RCS	6	rw	f_{RTC} Clock Selection 0 _B f _{OSI} selected 1 _B f _{ULP} selected
STDBYSEL	7	rw	f_{STDBY} Clock Selection 0 _B f _{OSI} selected 1 _B f _{ULP} selected

System Control Unit (SCU)

Field	Bits	Type	Description
WKUPSEL	8	rw	Wake-Up from Hibernate Trigger Input Selection 0_B HIB_IO_1 pin selected 1_B HIB_IO_0 pin selected
GPI0SEL	10	rw	General Purpose Input 0 Selection This bit field selects input to ERU0 module that optionally can be used with software as a general purpose input. 0_B 1_B HIB_IO_0 pin selected
HIBIO0POL	12	rw	HIBIO0 Polarity Set Selects the output polarity of the HIBIO0 0_B Direct value 1_B Inverted value
HIBIO1POL	13	rw	HIBIO1 Polarity Set Selects the output polarity of the HIBIO1 0_B Direct value 1_B Inverted value
HIBIO0SEL	[19:16]	rw	HIB_IO_0 Pin I/O Control (default HIBOUT) This bit field determines the Port n line x functionality. 0000_B Direct input, No input pull device connected 0001_B Direct input, Input pull-down device connected 0010_B Direct input, Input pull-up device connected 1000_B Push-pull HIB Control output 1001_B Push-pull WDT service output 1010_B Push-pull GPIO output 1100_B Open-drain HIB Control output 1101_B Open-drain WDT service output 1110_B Open-drain GPIO output 1111_B

System Control Unit (SCU)

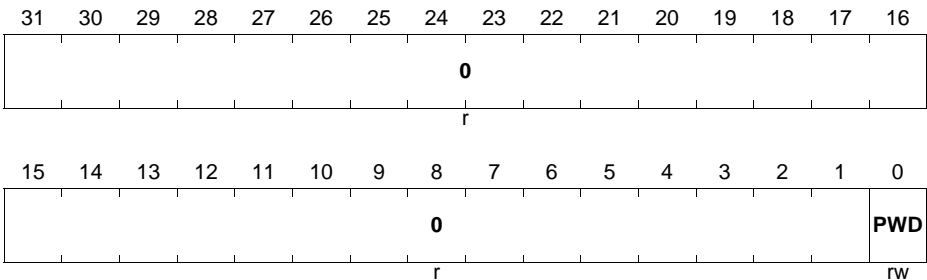
Field	Bits	Type	Description
HIBIO1SEL	[23:20]	rw	HIB_IO_1 Pin I/O Control (Default WKUP) This bit field determines the Port n line x functionality. 0000 _B Direct input, No input pull device connected 0001 _B Direct input, Input pull-down device connected 0010 _B Direct input, Input pull-up device connected 1000 _B Push-pull HIB Control output 1001 _B Push-pull WDT service output 1010 _B Push-pull GPIO output 1100 _B Open-drain HIB Control output 1101 _B Open-drain WDT service output 1110 _B Open-drain GPIO output 1111 _B
0	5,9, 11, [15:14], [29:24], [31:30]	r	Reserved Read as 0; should be written with 0.

OSCSICTRL

Control register for f_{OSI} clock source. A special mechanism keeps the the f_{OSI} clock active if the external crystal oscillator is switched off, regardless of the value of the PWD bit field. The f_{OSI} can be switched off only if the external crystal oscillator is enabled and the f_{ULP} clock toggling.

OSCSICTRL

f_{OSI} Control Register (0314_H) Reset Value: 0000 0001_H



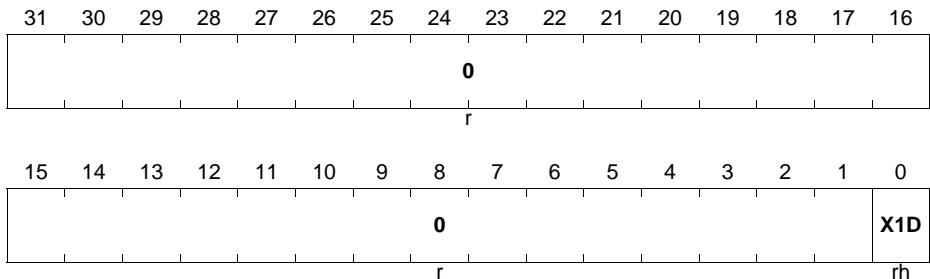
Field	Bits	Type	Description
PWD	0	rw	Turn OFF the f_{OSI} Clock Source 0_B Enabled 1_B Disabled <i>Note: The f_{OSI} needs to be enabled in order to prevent potential deadlock in case of external crystal failure.</i>
0	[31:1]	r	Reserved Read as 0; should be written with 0.

OSCULSTAT

Status register of the OSC_ULP oscillator.

OSCULSTAT

OSC_ULP Status Register (0318_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
X1D	0	rh	XTAL1 Data Value This bit monitors the value (level) of pin XTAL1. If XTAL1 is not used as clock input it can be used as GPI pin. This bit is only updated if X1DEN is set.
0	[31:1]	r	Reserved

OSCULCTRL

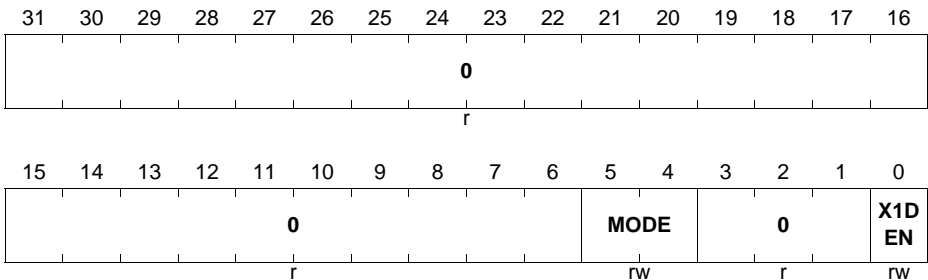
Control register for OSC_ULP oscillator. This register allows selection of clock generation with external crystal, direct clock input, or power down mode. Alternate GPI function of the pin is also controlled with this register.

OSCULCTRL

OSC_ULP Control Register

(031C_H)

Reset Value: 0000 0020_H



Field	Bits	Type	Description
X1DEN	0	rw	<p>XTAL1 Data General Purpose Input Enable The GPI data can be monitored with X1D bit of OSCULSTAT register</p> <p>0_B Data input inactivated, power down 1_B Data input active</p> <p><i>Note: It is strongly recommended to keep this function inactivated if the XTAL1 input is used as clock source</i></p>
MODE	[5:4]	rw	<p>Oscillator Mode</p> <p>00_B Oscillator is enabled, in operation 01_B Oscillator is enabled, in bypass mode 10_B Oscillator in power down 11_B Oscillator in power down, can be used as GPI</p> <p><i>Note: Use of the oscillator input require that X1DEN bit is activated</i></p>
0	[3:1], [31:6]	r	<p>Reserved Read as 0; should be written with 0.</p>

11.10.4 RCU Registers

RSTSTAT

Reset status register. This register needs to be checked after system startup in order to determine last reset reason.

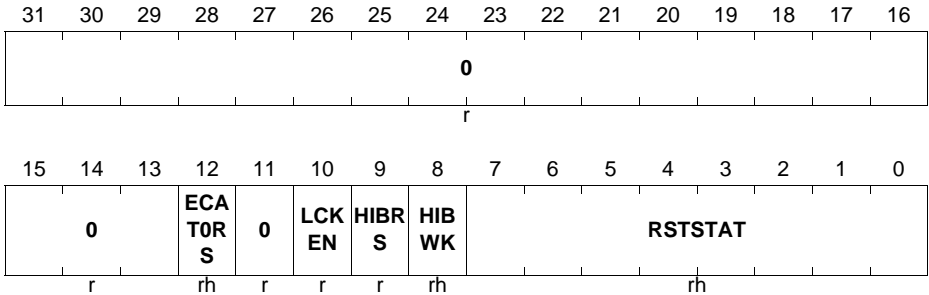
System Control Unit (SCU)

RSTSTAT

RCU Reset Status

(0400_H)

Reset Value:0000 0XXX_H



Field	Bits	Type	Description
RSTSTAT	[7:0]	rh	Reset Status Information Provides reason of last reset 00000001 _B PORST reset 00000010 _B SWD reset 00000100 _B PV reset 00001000 _B CPU system reset 00010000 _B CPU lockup reset 00100000 _B WDT reset 01000000 _B Reserved 10000000 _B Parity Error reset
HIBWK	8	rh	Hibernate Wake-up Status 0 _B No Wake-up 1 _B Wake-up event <i>Note: Field is cleared with enable of Hibernate mode</i>
HIBRS	9	r	Hibernate Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
LCKEN	10	r	Enable Lockup Status 0 _B Reset by Lockup disabled 1 _B Reset by Lockup enabled

System Control Unit (SCU)

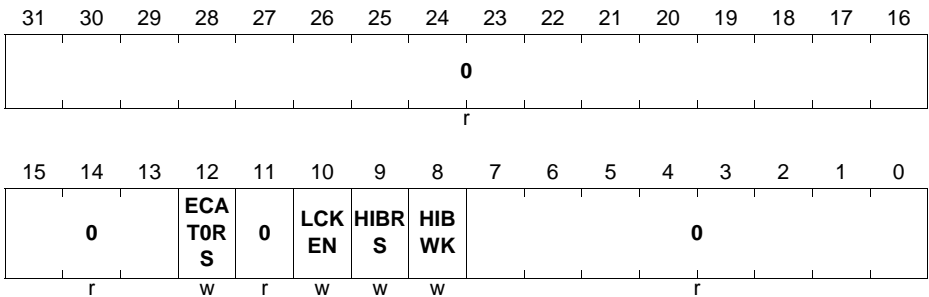
Field	Bits	Type	Description
ECAT0RS	12	rh	ECAT0 Reset Status Information Provides information that the system reset was triggered by the ECAT reset request 0 _B Reset did not occur 1 _B Reset occurred
0	11, [31:13]	r	Reserved

RSTSET

Selective configuration of reset behavior in the system. Write one to set selected bit, writing zeros has no effect.

RSTSET

RCU Reset Set Register (0404_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
HIBWK	8	w	Set Hibernate Wake-up Reset Status 0 _B No effect 1 _B Assert reset status bit
HIBRS	9	w	Set Hibernate Reset 0 _B No effect 1 _B Assert reset
LCKEN	10	w	Enable Lockup Reset 0 _B No effect 1 _B Enable reset when Lockup gets asserted

System Control Unit (SCU)

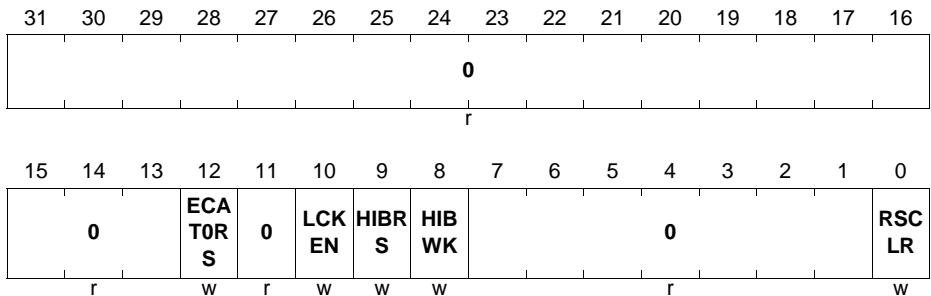
Field	Bits	Type	Description
ECAT0RS	12	w	ECAT0 Reset Status Information 0 _B No effect 1 _B Assert reset status bit
0	[7:0], 11, [31:13]	r	Reserved Read as 0; should be written with 0.

RSTCLR

Selective configuration of reset behavior in the system. Write one to clear selected bit, writing zeros has no effect.

RSTCLR

RCU Reset Clear Register (0408_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
RSCLR	0	w	Clear Reset Status 0 _B No effect 1 _B Clears field RSTSTAT .RSTSTAT
HIBWK	8	w	Clear Hibernate Wake-up Reset Status 0 _B No effect 1 _B De-assert reset status bit
HIBRS	9	w	Clear Hibernate Reset 0 _B No effect 1 _B De-assert reset
LCKEN	10	w	Enable Lockup Reset 0 _B No effect 1 _B Disable reset when Lockup gets asserted

System Control Unit (SCU)

Field	Bits	Type	Description
ECAT0RS	12	w	ECAT0 Reset Status Information 0 _B No effect 1 _B De-assert reset status
0	[7:1], 11, [31:13]	r	Reserved Read as 0; should be written with 0.

PRSTAT0

Selective reset status register for Peripherals 0.

PRSTAT0

RCU Peripheral 0 Reset Status (040C_H) **Reset Value: 0001 0F9F_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			0					0			0				ERU1RS
			r					r			r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		0		USIC0RS	1	1	1	CCU80RS	0	1	CCU41RS	CCU40RS	1	VADCRS	
		r		r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
VADCRS	0	r	VADC Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
CCU40RS	2	r	CCU40 Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
CCU41RS	3	r	CCU41 Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
CCU80RS	7	r	CCU80 Reset Status 0 _B Reset de-asserted 1 _B Reset asserted

System Control Unit (SCU)

Field	Bits	Type	Description
USIC0RS	11	r	USIC0 Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
ERU1RS	16	r	ERU1 Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
0	[6:5], [15:12], [22:17], 23, [31:24]	r	Reserved
1	1,4, 8,9,10	r	Reserved

PRSET0

Selective reset assert register for Peripherals 0. Write one to assert selected reset, writing zeros has no effect.

PRSET0

RCU Peripheral 0 Reset Set (0410_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								0	0						ERU1RS
r								r	r						w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				USIC0RS	0	0	0	CCU80RS	0	0	CCU41RS	CCU40RS	0	VADCRS	
r				w	r	r	r	w	r	r	w	w	r	w	

Field	Bits	Type	Description
VADCRS	0	w	VADC Reset Assert 0 _B No effect 1 _B Assert reset

System Control Unit (SCU)

Field	Bits	Type	Description
CCU40RS	2	w	CCU40 Reset Assert 0 _B No effect 1 _B Assert reset
CCU41RS	3	w	CCU41 Reset Assert 0 _B No effect 1 _B Assert reset
CCU80RS	7	w	CCU80 Reset Assert 0 _B No effect 1 _B Assert reset
USIC0RS	11	w	USIC0 Reset Assert 0 _B No effect 1 _B Assert reset
ERU1RS	16	w	ERU1 Reset Assert 0 _B No effect 1 _B Assert reset
0	1,4, [6:5], 8,9,10, [15:12], [22:17], 23, [31:24]	r	Reserved Read as 0; should be written with 0.

PRCLR0

Selective reset de-assert register for Peripherals 0. Write one to de-assert selected reset, writing zeros has no effect.

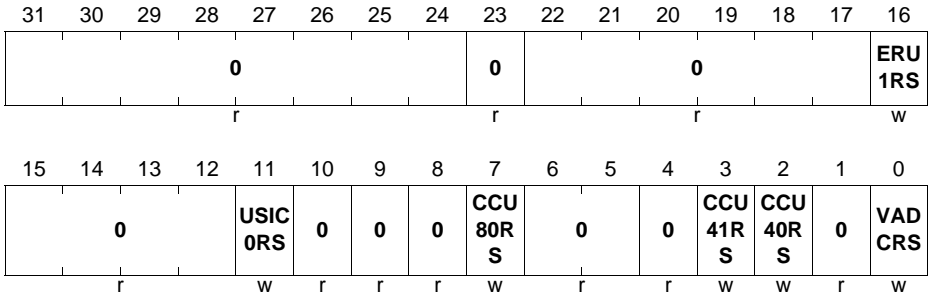
System Control Unit (SCU)

PRCLR0

RCU Peripheral 0 Reset Clear

(0414_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
VADCRS	0	w	VADC Reset Clear 0 _B No effect 1 _B De-assert reset
CCU40RS	2	w	CCU40 Reset Clear 0 _B No effect 1 _B De-assert reset
CCU41RS	3	w	CCU41 Reset Clear 0 _B No effect 1 _B De-assert reset
CCU80RS	7	w	CCU80 Reset Clear 0 _B No effect 1 _B De-assert reset
USIC0RS	11	w	USIC0 Reset Clear 0 _B No effect 1 _B De-assert reset
ERU1RS	16	w	ERU1 Reset Clear 0 _B No effect 1 _B De-assert reset

System Control Unit (SCU)

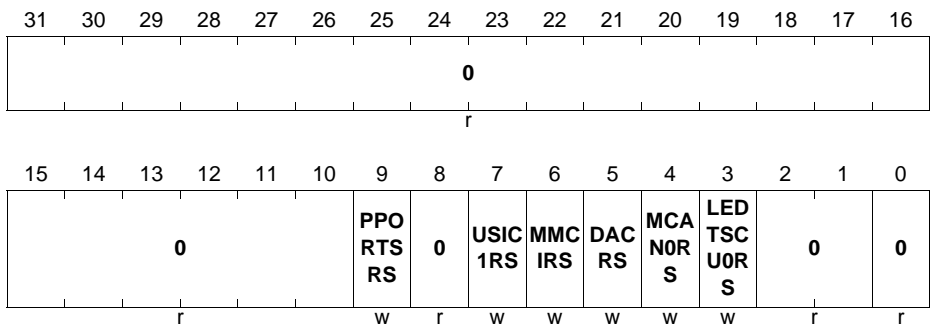
Field	Bits	Type	Description
MMCIRS	6	r	MMC Interface Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
USIC1RS	7	r	USIC1 Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
PSPORTSRS	9	r	PORTS Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
0	[2:1], [31:10]	r	Reserved
1	0, 8	r	Reserved

PRSET1

Selective reset assert register for Peripherals 1. Write one to assert selected reset, writing zeros has no effect.

PRSET1

RCU Peripheral 1 Reset Set (041C_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
LEDTSCU0RS	3	w	LEDTS Reset Assert 0 _B No effect 1 _B Assert reset

System Control Unit (SCU)

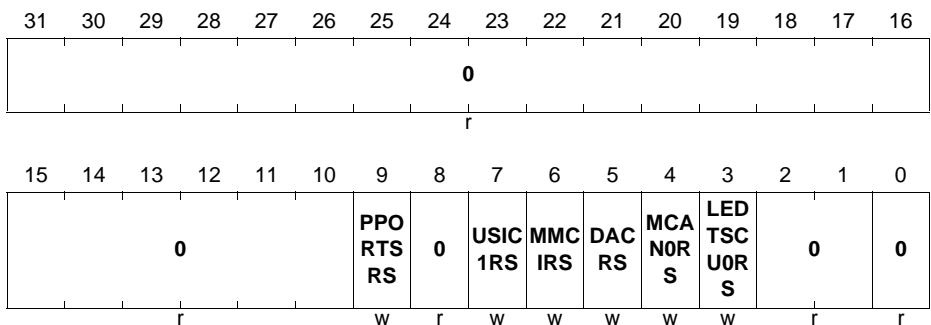
Field	Bits	Type	Description
MCAN0RS	4	w	MultiCAN Reset Assert 0 _B No effect 1 _B Assert reset
DACRS	5	w	DAC Reset Assert 0 _B No effect 1 _B Assert reset
MMCIRS	6	w	MMC Interface Reset Assert 0 _B No effect 1 _B Assert reset
USIC1RS	7	w	USIC1 Reset Assert 0 _B No effect 1 _B Assert reset
PPORTSRS	9	w	PORTS Reset Assert 0 _B No effect 1 _B Assert reset
0	0, [2:1], 8, [31:10]	r	Reserved Read as 0; should be written with 0.

PRCLR1

Selective reset de-assert register for Peripherals 1. Write one to de-assert selected reset, writing zeros has no effect.

PRCLR1

RCU Peripheral 1 Reset Clear (0420_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
LEDTSCU0RS	3	w	LEDTS Reset Clear 0 _B No effect 1 _B De-assert reset
MCAN0RS	4	w	MultiCAN Reset Clear 0 _B No effect 1 _B De-assert reset
DACRS	5	w	DAC Reset Clear 0 _B No effect 1 _B De-assert reset
MMCIRS	6	w	MMC Interface Reset Clear 0 _B No effect 1 _B De-assert reset
USIC1RS	7	w	USIC1 Reset Clear 0 _B No effect 1 _B De-assert reset
PPORTSRS	9	w	PORTS Reset Clear 0 _B No effect 1 _B De-assert reset
0	0, [2:1], 8, [31:10]	r	Reserved Read as 0; should be written with 0.

PRSTAT2

Selective reset status register for Peripherals 2.

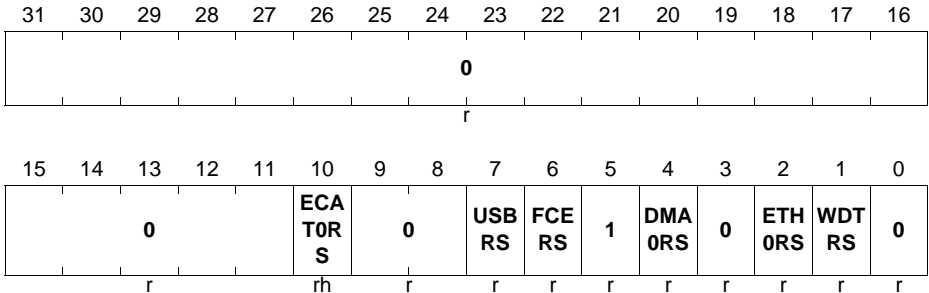
System Control Unit (SCU)

PRSTAT2

RCU Peripheral 2 Reset Status

(0424_H)

Reset Value: 0000 04F6_H



Field	Bits	Type	Description
WDTRS	1	r	WDT Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
ETH0RS	2	r	ETH0 Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
DMA0RS	4	r	DMA0 Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
FCERS	6	r	FCE Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
USBR S	7	r	USB Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
ECAT0RS	10	rh	ECAT0 Reset Status 0 _B Reset de-asserted 1 _B Reset asserted
0	0, 3, [9:8], [31:11]	r	Reserved

System Control Unit (SCU)

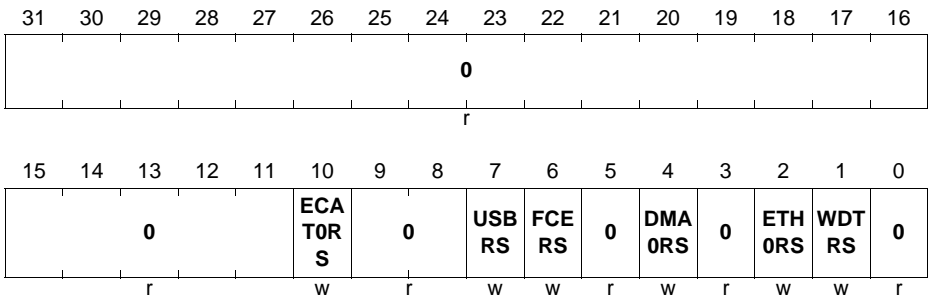
Field	Bits	Type	Description
1	5	r	Reserved

PRSET2

Selective reset assert register for Peripherals 2. Write one to assert selected reset, writing zeros has no effect.

PRSET2

RCU Peripheral 2 Reset Set (0428_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
WDTRS	1	w	WDT Reset Assert 0 _B No effect 1 _B Assert reset
ETH0RS	2	w	ETH0 Reset Assert 0 _B No effect 1 _B Assert reset
DMA0RS	4	w	DMA0 Reset Assert 0 _B No effect 1 _B Assert reset
FCERS	6	w	FCE Reset Assert 0 _B No effect 1 _B Assert reset
USBR5	7	w	USB Reset Assert 0 _B No effect 1 _B Assert reset

System Control Unit (SCU)

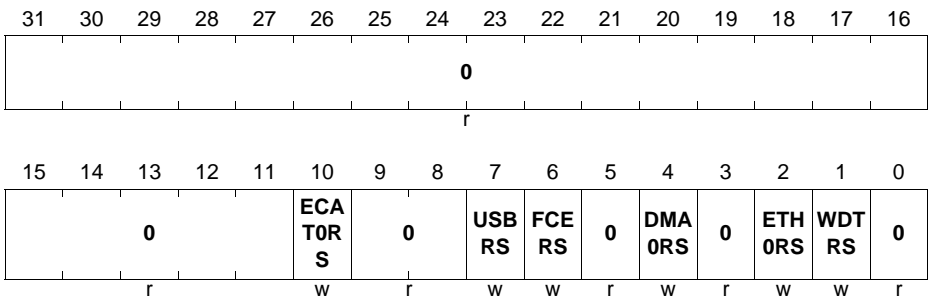
Field	Bits	Type	Description
ECAT0RS	10	w	ECAT0 Reset Assert 0 _B No effect 1 _B Assert reset
0	0, 3, 5, [9:8], [31:11]	r	Reserved Read as 0; should be written with 0.

PRCLR2

Selective reset de-assert register for Peripherals 2. Write one to de-assert selected reset, writing zeros has no effect.

PRCLR2

RCU Peripheral 2 Reset Clear (042C_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
WDTRS	1	w	WDT Reset Clear 0 _B No effect 1 _B De-assert reset
ETH0RS	2	w	ETH0 Reset Clear 0 _B No effect 1 _B De-assert reset
DMA0RS	4	w	DMA0 Reset Clear 0 _B No effect 1 _B De-assert reset

System Control Unit (SCU)

Field	Bits	Type	Description
FCERS	6	w	FCE Reset Clear 0 _B No effect 1 _B De-assert reset
USBRS	7	w	USB Reset Clear 0 _B No effect 1 _B De-assert reset
ECAT0RS	10	w	ECAT0 Reset Clear 0 _B No effect 1 _B De-assert reset
0	0, 3, 5, [9:8], [31:11]	r	Reserved Read as 0; should be written with 0.

11.10.5 CCU Registers

CLKSTAT

Global clock status register.

CLKSTAT

Clock Status Register

(0600_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										WDT CST	CCU CST	0	ETH OCS T	MMC CST	USB CST
r										r	r	r	r	r	r

Field	Bits	Type	Description
USBCST	0	r	USB Clock Status 0 _B Clock disabled 1 _B Clock enabled
MMCCST	1	r	MMC Clock Status 0 _B Clock disabled 1 _B Clock enabled
ETH0CST	2	r	Ethernet Clock Status 0 _B Clock disabled 1 _B Clock enabled
CCUCST	4	r	CCU Clock Status 0 _B Clock disabled 1 _B Clock enabled
WDTCS	5	r	WDT Clock Status 0 _B Clock disabled <i>Note: WDT clock can be put on hold in debug mode when this behavior is enabled at the watchdog</i> 1 _B Clock enabled
0	3, [31:6]	r	Reserved Read as 0.

CLKSET

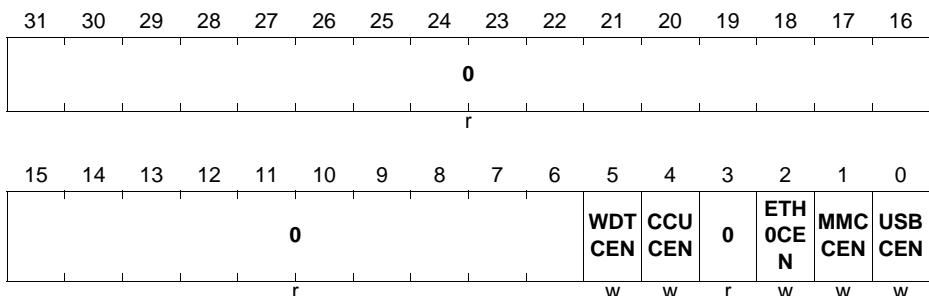
Global clock enable register. Write one to enable selected clock, writing zeros has no effect.

CLKSET

CLK Set Register

(0604_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
USBCEN	0	w	USB Clock Enable 0 _B No effect 1 _B Enable
MMCCEN	1	w	MMC Clock Enable 0 _B No effect 1 _B Enable
ETH0CEN	2	w	Ethernet Clock Enable 0 _B No effect 1 _B Enable
CCUCEN	4	w	CCU Clock Enable 0 _B No effect 1 _B Enable
WDTCEN	5	w	WDT Clock Enable 0 _B No effect 1 _B Enable
0	3, [31:6]	r	Reserved Read as 0; should be written with 0.

CLKCLR

Global clock disable register. Write one to disable selected clock, writing zeros has no effect.

CLKCLR

CLK Clear Register

(0608_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										WDT CDI	CCU CDI	0	ETH 0CDI	MMC CDI	USB CDI
r										w	w	r	w	w	w

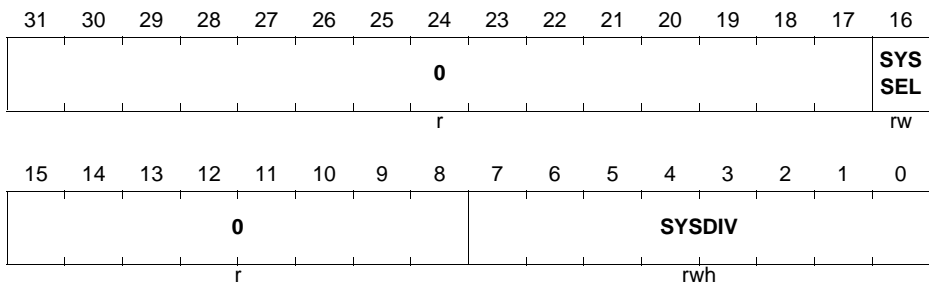
Field	Bits	Type	Description
USBCDI	0	w	USB Clock Disable 0 _B No effect 1 _B Disable clock
MMCCDI	1	w	MMC Clock Disable 0 _B No effect 1 _B Disable clock
ETH0CDI	2	w	Ethernet Clock Disable 0 _B No effect 1 _B Disable clock
CCUCDI	4	w	CCU Clock Disable 0 _B No effect 1 _B Disable clock
WDTCDI	5	w	WDT Clock Disable 0 _B No effect 1 _B Disable clock
0	3, [31:6]	r	Reserved Read as 0; should be written with 0.

SYCLKCR

System clock control register.

SYCLKCR

System Clock Control Register (060C_H) **Reset Value: 0000 0000_H**



System Control Unit (SCU)

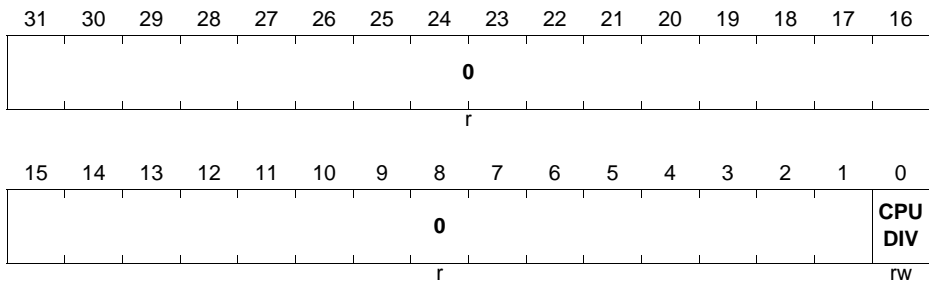
Field	Bits	Type	Function
SYSDIV	[7:0]	rwh	System Clock Division Value The value the divider operates is (SYSDIV+1).
SYSSEL	16	rw	System Clock Selection Value 0 _B f_{OFI} clock 1 _B f_{PLL} clock
0	[15:8], [31:17]	r	Reserved Read as 0; should be written with 0.

CPUCLKCR

CPU clock control register.

CPUCLKCR

CPU Clock Control Register (0610_H) Reset Value: 0000 0000_H



Field	Bits	Type	Function
CPUDIV	0	rw	CPU Clock Divider Enable This bit enables division of f_{SYS} clock to produce f_{CPU} clock. 0 _B $f_{CPU} = f_{SYS}$ 1 _B $f_{CPU} = f_{SYS} / 2$ <i>Note: Some clock division settings are not allowed.</i> <i>See Table 11-5 for more details.</i>
0	[31:1]	r	Reserved Read as 0; should be written with 0.

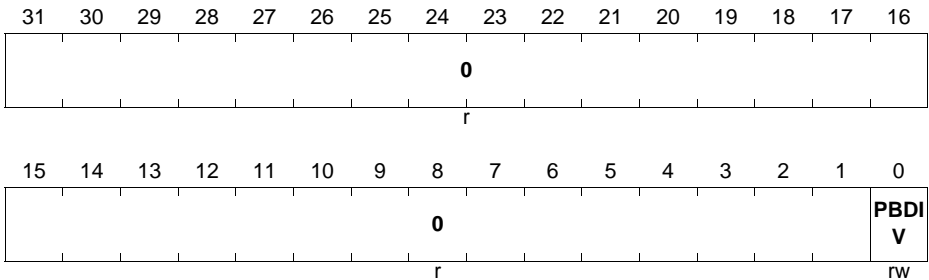
PBCLKCR

Peripheral clock control register.

PBCLKCR

Peripheral Bus Clock Control Register (0614_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Function
PBDIV	0	rw	<p>PB Clock Divider Enable</p> <p>This bit enables division of f_{SYS} clock to produce f_{PERIPH} clock.</p> <p>0_B $f_{PERIPH} = f_{CPU}$</p> <p>1_B $f_{PERIPH} = f_{CPU} / 2$</p> <p><i>Note: Some clock division settings are not allowed. See Table 11-5 for more details.</i></p>
0	[31:1]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

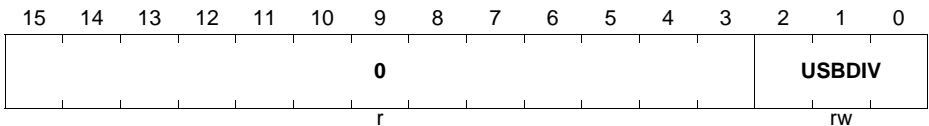
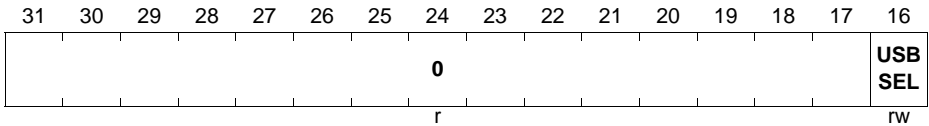
USBCLKCR

USB clock control register.

System Control Unit (SCU)

USBCLKCR

USB Clock Control Register (0618_H) Reset Value: 0000 0000_H



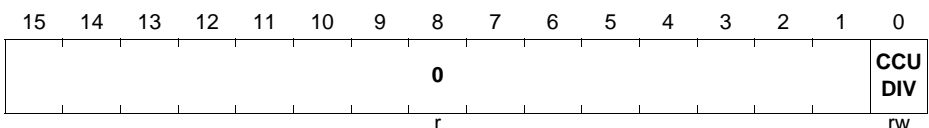
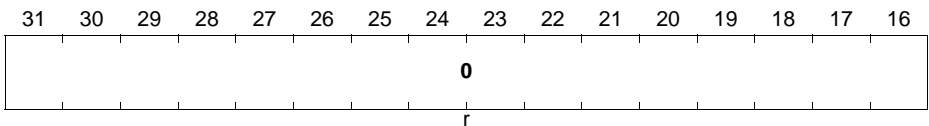
Field	Bits	Type	Function
USBDIV	[2:0]	rw	USB Clock Divider Value PLL clock is divided by USBDIV + 1 Must only be programmed, when clock is not used
USBSEL	16	rw	USB Clock Selection Value 0 _B USB PLL Clock 1 _B PLL Clock
0	[15:3], [31:17]	r	Reserved Read as 0; should be written with 0.

CCUCLKCR

CCUx clock control register.

CCUCLKCR

CCU Clock Control Register (0620_H) Reset Value: 0000 0000_H



Field	Bits	Type	Function
CCUDIV	0	rw	CCU Clock Divider Enable This bit enables division of f_{SYS} clock to produce f_{CCU} clock. $0_B \quad f_{CCU} = f_{SYS}$ $1_B \quad f_{CCU} = f_{SYS} / 2$ <i>Note: Some clock division settings are not allowed.</i> See Table 11-5 for more details.
0	[31:1]	r	Reserved Read as 0; should be written with 0.

WDTCLKCR

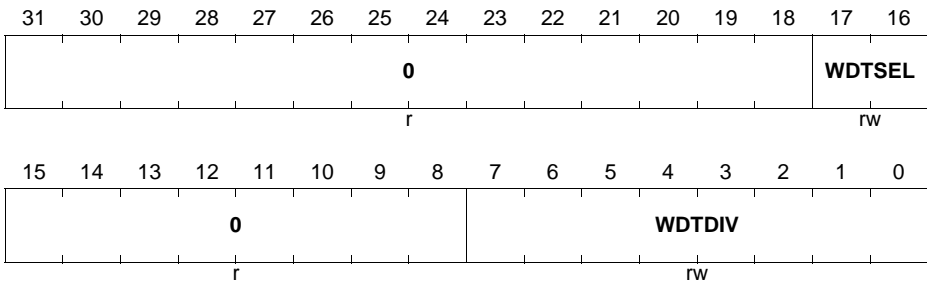
System watchdog (WDT) clock control register.

WDTCLKCR

WDT Clock Control Register

(0624_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Function
WDTDIV	[7:0]	rw	WDT Clock Divider Value WDT is divided by WDTDIV + 1 Must only be programmed, when clock is not used
WDTSEL	[17:16]	rw	WDT Clock Selection Value $00_B \quad f_{OFI}$ clock $01_B \quad f_{STDBY}$ clock $10_B \quad f_{PLL}$ clock 11_B Reserved

System Control Unit (SCU)

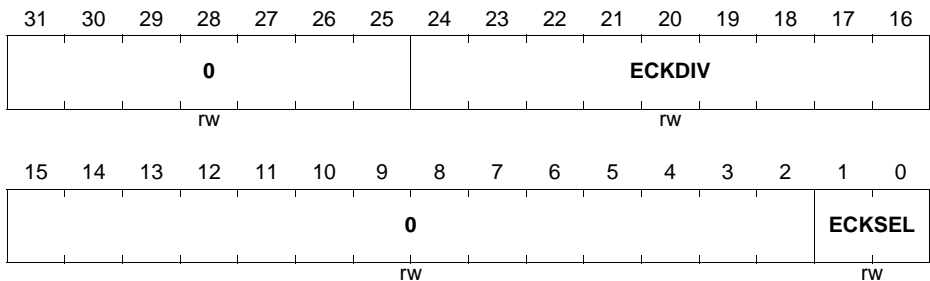
Field	Bits	Type	Function
0	[15:8], [31:18]	r	Reserved Read as 0; should be written with 0.

EXTCLKCR

External clock control register. Use this register to select output clock.

EXTCLKCR

External Clock Control (0628_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
ECKSEL	[1:0]	rw	External Clock Selection Value 00 _B f_{SYS} clock 01 _B Reserved 10 _B f_{USB} clock divided according to ECKDIV bit field configuration 11 _B f_{PLL} clock divided according to ECKDIV bit field configuration
ECKDIV	[24:16]	rw	External Clock Divider Value Selected clock is divided by ECKDIV + 1 Must only be programmed, when clock is not used
0	[15:2], [31:25]	rw	Reserved Read as 0; should be written with 0.

MLINKCLKCR

Multi-link register for central clock control. This register allows consistent reconfiguration of multiple clocks with a single register access and prevents invalid configurations i.e. clock division combinations are not allowed (see [Table 11-5](#) for more details). An

System Control Unit (SCU)

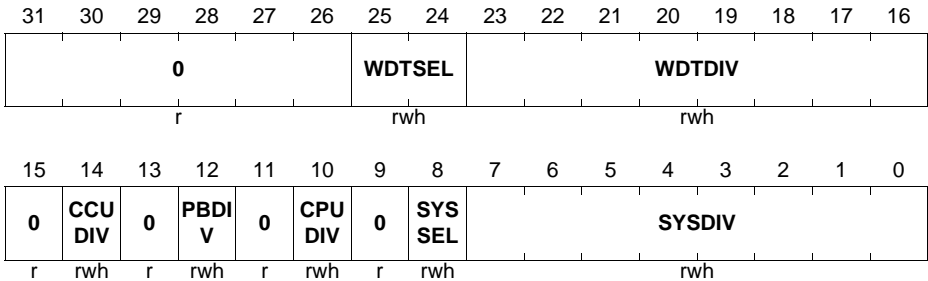
attempt to write invalid value will be rejected and result in generation of bus error response.

MLINKCLKCR

Multi-Link Clock Control

(062C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SYSDIV	[7:0]	rwh	System Clock Division Value The value the divider operates is (SYSDIV+1).
SYssel	8	rwh	System Clock Selection Value 0 _B f_{OFI} clock 1 _B f_{PLL} clock
CPUDIV	10	rwh	CPU Clock Divider Enable This bit enables division of f_{SYS} clock to produce f_{CPU} clock. 0 _B $f_{CPU} = f_{SYS}$ 1 _B $f_{CPU} = f_{SYS} / 2$ <i>Note: Some clock division combinations are not allowed. See Table 11-5 for more details. This register must ignore the write and generate error response if invalid configuration attempt detected.</i>

System Control Unit (SCU)

Field	Bits	Type	Description
PBDIV	12	rwh	<p>PB Clock Divider Enable</p> <p>This bit enables division of f_{SYS} clock to produce f_{PERIPH} clock.</p> $0_B \quad f_{PERIPH} = f_{CPU}$ $1_B \quad f_{PERIPH} = f_{CPU} / 2$ <p><i>Note: Some clock division combinations are not allowed. See Table 11-5 for more details. This register must ignore the write and generate error response if invalid configuration attempt detected.</i></p>
CCUDIV	14	rwh	<p>CCU Clock Divider Enable</p> <p>This bit enables division of f_{SYS} clock to produce f_{CCU} clock.</p> $0_B \quad f_{CCU} = f_{SYS}$ $1_B \quad f_{CCU} = f_{SYS} / 2$ <p><i>Note: Some clock division combinations are not allowed. See Table 11-5 for more details. This register must ignore the write and generate error response if invalid configuration attempt detected.</i></p>
WTDIV	[23:16]	rwh	<p>WDT Clock Divider Value</p> <p>WDT is divided by WTDIV + 1 Must only be programmed, when clock is not used</p>
WDTSEL	[25:24]	rwh	<p>WDT Clock Selection Value</p> $00_B \quad f_{OFI} \text{ clock}$ $01_B \quad f_{STDBY} \text{ clock}$ $10_B \quad f_{PLL} \text{ clock}$ $11_B \quad \text{Reserved}$
0	9, 11,13, 15, [31:26]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

SLEEPSCR

Configuration register that defines some system behavior aspects while in sleep mode. The original clock configuration gets restored upon wake-up from sleep mode.

Note: Switching off a module clock during operation may result in unexpected effects like e.g. clock spikes or protocol violations. Before entering Sleep mode the affected

System Control Unit (SCU)

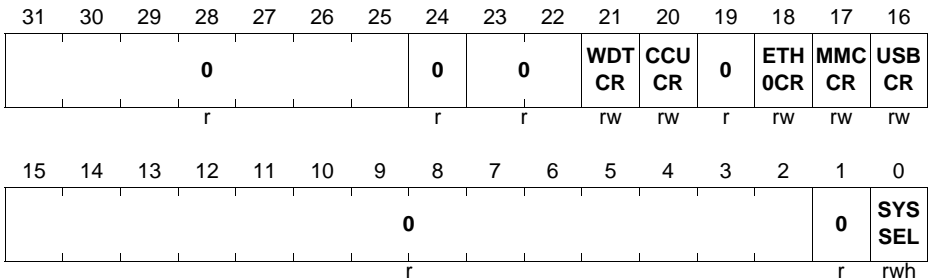
modules should be in reset state. After re-toration of the clocks the modules need to be re-initialized in order to ensure proper function.

SLEEP_CR

Sleep Control Register

(0630_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SYSSEL	0	rwh	System Clock Selection Value 0 _B f_{OFI} clock 1 _B f_{PLL} clock
USBCR	16	rw	USB Clock Control in Sleep Mode 0 _B Disabled 1 _B Enabled
MMCCR	17	rw	MMC Clock Control in Sleep Mode 0 _B Disabled 1 _B Enabled
ETH0CR	18	rw	Ethernet Clock Control in Sleep Mode 0 _B Disabled 1 _B Enabled
CCUCR	20	rw	CCU Clock Control in Sleep Mode 0 _B Disabled 1 _B Enabled
WDTCR	21	rw	WDT Clock Control in Sleep Mode 0 _B Disabled 1 _B Enabled

System Control Unit (SCU)

Field	Bits	Type	Description
0	1, 19, [15:2], [23:22], 24, [31:25]	r	Reserved Read as 0; should be written with 0.

DSLEEPCCR

Configuration register that defines some system behavior aspects while in Deep Sleep mode. The original clock configuration gets restored upon wake-up from sleep mode. The PLL re-initialization is required after wake-up from Deep Sleep mode if was enabled before entering Deep Sleep mode and configured to go into power down while in Deep Sleep mode.

Note: Switching off a module clock during operation may result in unexpected effects like e.g. clock spikes or protocol violations. Before entering Sleep mode the affected modules should be in reset state. After re-toration of the clocks the modules need to be re-initialized in order to ensure proper function.

DSLEEPCCR

Deep Sleep Control Register (0634_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						0		0	WDT CR	CCU CR	0	ETH 0CR	MMC CR	USB CR	
r						r		r	rw	rw	r	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		VCO PDN	PLL PDN	FPD N							0			0	SYS SEL
r		rw	rw	rw							r			r	rwh

Field	Bits	Type	Description
SYSSEL	0	rwh	System Clock Selection Value 0 _B f_{OFI} clock 1 _B f_{PLL} clock
FPDN	11	rw	Flash Power Down 1 _B Flash power down module 0 _B No effect

System Control Unit (SCU)

Field	Bits	Type	Description
PLLDPN	12	rw	PLL Power Down 1 _B Switch off main PLL 0 _B No effect
VCOPDN	13	rw	VCO Power Down 1 _B Switch off VCO of main PLL 0 _B No effect
USBCR	16	rw	USB Clock Control in Deep Sleep Mode 0 _B Disabled 1 _B Enabled
MMCCR	17	rw	MMC Clock Control in Deep Sleep Mode 0 _B Disabled 1 _B Enabled
ETH0CR	18	rw	Ethernet Clock Control in Deep Sleep Mode 0 _B Disabled 1 _B Enabled
CCUCR	20	rw	CCU Clock Control in Deep Sleep Mode 0 _B Disabled 1 _B Enabled
WDTCR	21	rw	WDT Clock Control in Deep Sleep Mode 0 _B Disabled 1 _B Enabled
0	1, [10:2], [15:14], 19, [23:22], 24, [31:25]	r	Reserved Read as 0; should be written with 0.

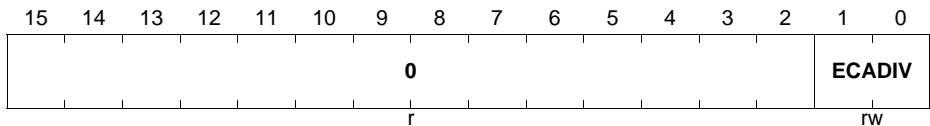
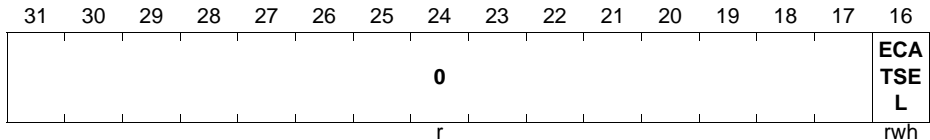
ECATCLKCR

EtherCAT clock control register.

System Control Unit (SCU)

ECATCLKCR

EtherCAT Clock Control Register (0638_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ECADIV	[1:0]	rw	EtherCAT Clock Divider Value Selected clock is divided by ECATDIV + 1 Must only be programmed, when clock is not used
ECATSEL	16	rwh	EtherCAT Clock Selection Value 0 _B f_{PLLUSB} clock 1 _B f_{PLL} clock <i>Note: Configuration of this bit field shall be performed before the affected modules are release from reset state. This bitfield shall not be altered while the modules are in operaton.</i>
0	[15:2], [31:17]	r	Reserved Read as 0; should be written with 0.

CGATSTAT0

Clock gating status for peripherals 0.

System Control Unit (SCU)

CGATSTAT0

Peripheral 0 Clock Gating Status

(0640_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								0	0						ERU 1
r								r	r						r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				USIC 0	0	POSIF0	0	CCU80	0		0	CCU41	CCU40	0	VADC
r				r	r	r	r	r	r		r	r	r	r	r

Field	Bits	Type	Description
VADC	0	r	VADC Gating Status 0 _B Gating de-asserted 1 _B Gating asserted
CCU40	2	r	CCU40 Gating Status 0 _B Gating de-asserted 1 _B Gating asserted
CCU41	3	r	CCU41 Gating Status 0 _B Gating de-asserted 1 _B Gating asserted
CCU80	7	r	CCU80 Gating Status 0 _B Gating de-asserted 1 _B Gating asserted
POSIF0	9	r	POSIF0 Gating Status 0 _B Gating de-asserted 1 _B Gating asserted
USIC0	11	r	USIC0 Gating Status 0 _B Gating de-asserted 1 _B Gating asserted
ERU1	16	r	ERU1 Gating Status 0 _B Gating de-asserted 1 _B Gating asserted

Field	Bits	Type	Description
0	1,4, [6:5], 8, 10, [15:12], [22:17], 23, [31:24]	r	Reserved

CGATSET0

Clock gating enable for peripherals 0. Write one to selected bit to Enable gating of corresponding clock, writing zeros has no effect.

Note: Clock gating shall not be activated unless the module is in reset state.

CGATSET0

Peripheral 0 Clock Gating Set (0644_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			0					0			0				ERU 1
			r					r			r				w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0		USIC 0	0	POSIF0	0	CCU 80	0	0	0	CCU 41	CCU 40	0	VAD C	
	r		w	r	w	r	w	r	r	w	w	r	w		

Field	Bits	Type	Description
VADC	0	w	VADC Gating Set 0 _B No effect 1 _B Enable gating
CCU40	2	w	CCU40 Gating Set 0 _B No effect 1 _B Enable gating
CCU41	3	w	CCU41 Gating Set 0 _B No effect 1 _B Enable gating

Field	Bits	Type	Description
VADC	0	w	VADC Gating Clear 0 _B No effect 1 _B Disable gating
CCU40	2	w	CCU40 Gating Clear 0 _B No effect 1 _B Disable gating
CCU41	3	w	CCU41 Gating Clear 0 _B No effect 1 _B Disable gating
CCU80	7	w	CCU80 Gating Clear 0 _B No effect 1 _B Disable gating
POSIF0	9	w	POSIF0 Gating Clear 0 _B No effect 1 _B Disable gating
USIC0	11	w	USIC0 Gating Clear 0 _B No effect 1 _B Disable gating
ERU1	16	w	ERU1 Gating Clear 0 _B No effect 1 _B Disable gating
0	1,4, [6:5], 8, 10, [15:12], [22:17], 23, [31:24]	r	Reserved Read as 0; should be written with 0.

CGATSET1

Clock gating enable for peripherals 1. Write one to selected bit to Enable gating of corresponding clock, writing zeros has no effect.

Note: Clock gating shall not be activated unless the module is in reset state.

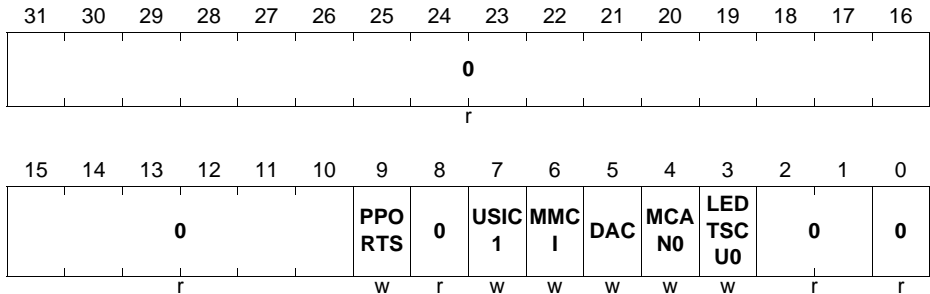
System Control Unit (SCU)

CGATSET1

Peripheral 1 Clock Gating Set

(0650_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
LEDTSCU0	3	w	LEDTS Gating Set 0 _B No effect 1 _B Enable gating
MCANO	4	w	MultiCAN Gating Set 0 _B No effect 1 _B Enable gating
DAC	5	w	DAC Gating Set 0 _B No effect 1 _B Enable gating
MMCI	6	w	MMC Interface Gating Set 0 _B No effect 1 _B Enable gating
USIC1	7	w	USIC1 Gating Set 0 _B No effect 1 _B Enable gating
PPORTS	9	w	PORTS Gating Set 0 _B No effect 1 _B Enable gating
0	0, [2:1], 8, [31:10]	r	Reserved Read as 0; should be written with 0.

System Control Unit (SCU)

CGATCLR1

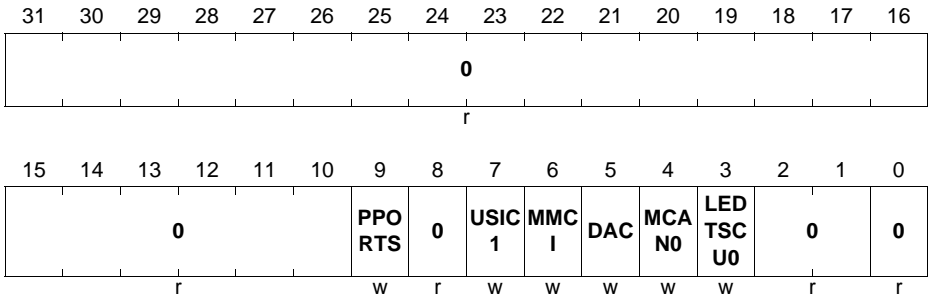
Clock gating disable for peripherals 1. Write one to selected bit to disable gating of corresponding clock, writing zeros has no effect.

CGATCLR1

Peripheral 1 Clock Gating Clear

(0654_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
LEDTSCU0	3	w	LEDTS Gating Clear 0 _B No effect 1 _B Disable gating
MCAN0	4	w	MultiCAN Gating Clear 0 _B No effect 1 _B Disable gating
DAC	5	w	DAC Gating Clear 0 _B No effect 1 _B Disable gating
MMCI	6	w	MMC Interface Gating Clear 0 _B No effect 1 _B Disable gating
USIC1	7	w	USIC1 Gating Clear 0 _B No effect 1 _B Disable gating
PPORTS	9	w	PORTS Gating Clear 0 _B No effect 1 _B Disable gating

Field	Bits	Type	Description
0	0, [2:1], 8, [31:10]	r	Reserved Read as 0; should be written with 0.

CGATSTAT2

Clock gating status for peripherals 2.

CGATSTAT2

Peripheral 2 Clock Gating Status (0658_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				ECA T0	0	USB	FCE	0	DMA 0	0	ETH 0	WDT	0		
r				r	r	r	r	r	r	r	r	r	r		

Field	Bits	Type	Description
WDT	1	r	WDT Gating Status 0 _B Gating de-asserted 1 _B Gating asserted
ETH0	2	r	ETH0 Gating Status 0 _B Gating de-asserted 1 _B Gating asserted
DMA0	4	r	DMA0 Gating Status 0 _B Gating de-asserted 1 _B Gating asserted
FCE	6	r	FCE Gating Status 0 _B Gating de-asserted 1 _B Gating asserted
USB	7	r	USB Gating Status 0 _B Gating de-asserted 1 _B Gating asserted

System Control Unit (SCU)

Field	Bits	Type	Description
ECAT0	10	r	ECAT0 Gating Status 0 _B Gating de-asserted 1 _B Gating asserted
0	0, 3, 5, [9:8], [31:11]	r	Reserved

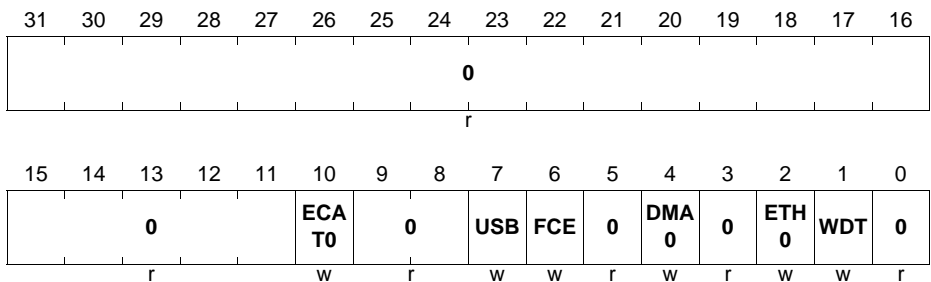
CGATSET2

Clock gating enable for peripherals 2. Write one to selected bit to Enable gating of corresponding clock, writing zeros has no effect.

Note: Clock gating shall not be activated unless the module is in reset state.

CGATSET2

Peripheral 2 Clock Gating Set (065C_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
WDT	1	w	WDT Gating Set 0 _B No effect 1 _B Enable gating
ETH0	2	w	ETH0 Gating Set 0 _B No effect 1 _B Enable gating

System Control Unit (SCU)

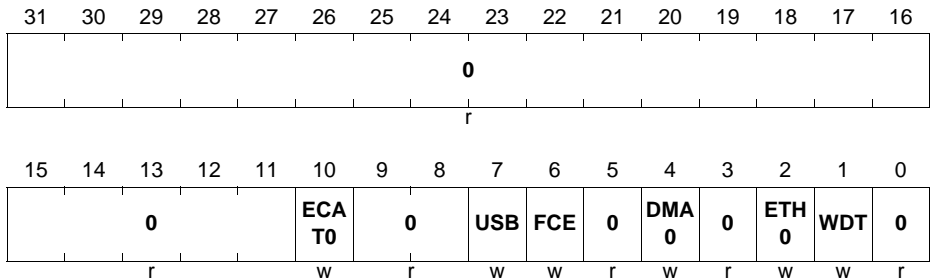
Field	Bits	Type	Description
DMA0	4	w	DMA0 Gating Set 0 _B No effect 1 _B Enable gating
FCE	6	w	FCE Gating Set 0 _B No effect 1 _B Enable gating
USB	7	w	USB Gating Set 0 _B No effect 1 _B Enable gating
ECAT0	10	w	ECAT0 Gating Set 0 _B No effect 1 _B Enable gating
0	0, 3, 5, [9:8], [31:11]	r	Reserved Read as 0; should be written with 0.

CGATCLR2

Clock gating disable for peripherals 2. Write one to selected bit to disable gating of corresponding clock, writing zeros has no effect.

CGATCLR2

Peripheral 2 Clock Gating Clear (0660_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
WDT	1	w	WDT Gating Clear 0 _B No effect 1 _B Disable gating
ETH0	2	w	ETH0 Gating Clear 0 _B No effect 1 _B Disable gating
DMA0	4	w	DMA0 Gating Clear 0 _B No effect 1 _B Disable gating
FCE	6	w	FCE Gating Clear 0 _B No effect 1 _B Disable gating
USB	7	w	USB Gating Clear 0 _B No effect 1 _B Disable gating
ECAT0	10	w	ECAT0 Gating Clear 0 _B No effect 1 _B Disable gating
0	0, 3, 5, [9:8], [31:11]	r	Reserved Read as 0; should be written with 0.

OSCHPSTAT

Status register of the OSC_HP oscillator.

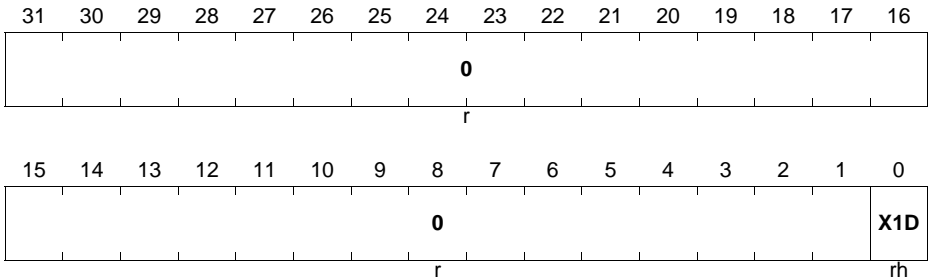
System Control Unit (SCU)

OSCHPSTAT

OSC_HP Status Register

(0700_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
X1D	0	rh	XTAL1 Data Value This bit monitors the value (level) of pin XTAL1. If XTAL1 is not used as clock input it can be used as GPI pin. This bit is only updated if X1DEN is set.
0	[31:1]	r	Reserved

OSCHPCTRL

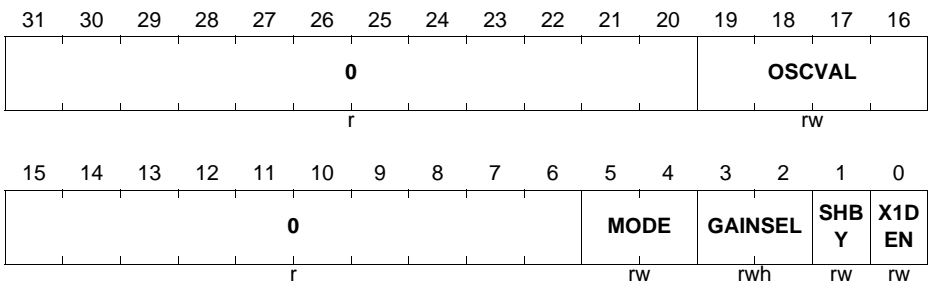
Control register of the OSC_HP oscillator.

OSCHPCTRL

OSC_HP Control Register

(0704_H)

Reset Value: 0000 003C_H



System Control Unit (SCU)

Field	Bits	Type	Description
X1DEN	0	rw	XTAL1 Data Enable 0 _B Bit X1D is not updated 1 _B Bit X1D can be updated
SHBY	1	rw	Shaper Bypass 0 _B The shaper is not bypassed 1 _B The shaper is bypassed
GAINSEL	[3:2]	rwh	Oscillator Gain Selection 00 _B The gain control is configured for frequencies from 4 MHz to 8 MHz 01 _B The gain control is configured for frequencies from 4 MHz to 16 MHz 10 _B The gain control is configured for frequencies from 4 MHz to 20 MHz 11 _B The gain control is configured for frequencies from 4 MHz to 25 MHz
MODE	[5:4]	rw	Oscillator Mode 00 _B External Crystal Mode and External Input Clock Mode. The oscillator Power-Saving Mode is not entered. 01 _B OSC is disabled. The oscillator Power-Saving Mode is not entered. 10 _B External Input Clock Mode and the oscillator Power-Saving Mode is entered 11 _B OSC is disabled. The oscillator Power-Saving Mode is entered.
OSCVL	[19:16]	rw	OSC Frequency Value This bit field defines the divider value that generates the reference clock that is supervised by the oscillator watchdog. f_{OSC} is divided by $OSCVL + 1$ in order to generate f_{OSCREF} .
0	[15:6], [31:20]	r	Reserved Read as 0; should be written with 0.

CLKCALCONST

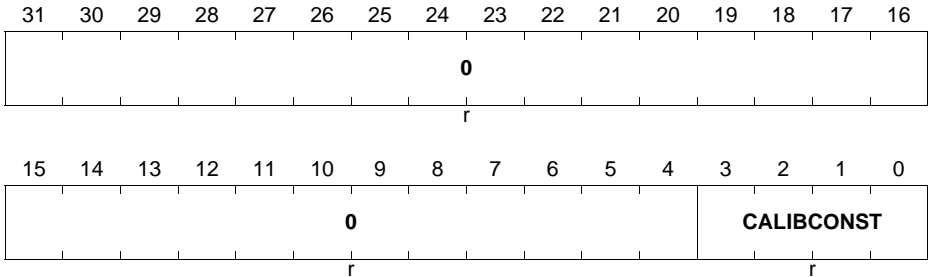
Clock calibration constant for PLL programming.

System Control Unit (SCU)

CLKCALCONST

Clock Calibration Constant Register (070C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CALIBCONST	[3:0]	r	Clock Calibration Constant Value This field contains clock calibration constant value for PLL configuration.
0	[31:4]	r	Reserved Read as 0; should be written with 0.

PLLSTAT

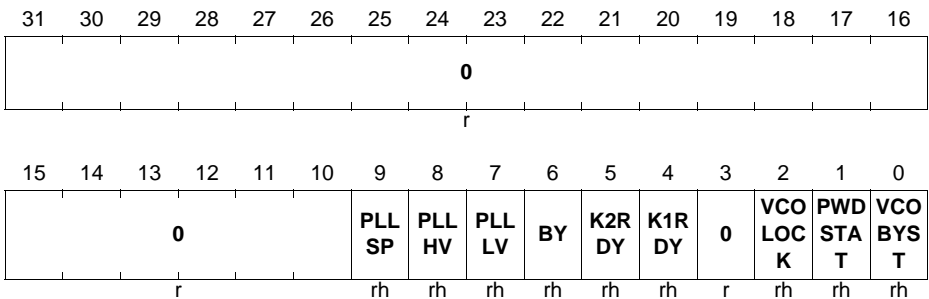
System PLL Status register.

PLLSTAT

PLL Status Register

(0710_H)

Reset Value: 0000 0002_H



System Control Unit (SCU)

Field	Bits	Type	Description
VCOBYST	0	rh	VCO Bypass Status 0 _B Free-running / Normal Mode is entered 1 _B Prescaler Mode is entered
PWDSTAT	1	rh	PLL Power-saving Mode Status 0 _B PLL Power-saving Mode was not entered 1 _B PLL Power-saving Mode was entered
VCOLOCK	2	rh	PLL LOCK Status 0 _B PLL not locked 1 _B PLL locked
K1RDY	4	rh	K1 Divider Ready Status This bit indicates if the K1-divider operates on the configured value or not. This is of interest if the value is changed. 0 _B K1-Divider does not operate with the new value 1 _B K1-Divider operate with the new value
K2RDY	5	rh	K2 Divider Ready Status This bit indicates if the K2-divider operates on the configured value or not. This is of interest if the value is changed. 0 _B K2-Divider does not operate with the new value 1 _B K2-Divider operate with the new value
BY	6	rh	Bypass Mode Status 0 _B Bypass Mode is not entered 1 _B Bypass Mode is entered. Input f_{OSC} is selected as output f_{PLL} .
PLLLV	7	rh	Oscillator for PLL Valid Low Status Bit This bit indicates if the frequency output of OSC is usable for the VCO part of the PLL. This is checked by the Oscillator Watchdog of the PLL. 0 _B The OSC frequency is not usable. Frequency f_{REF} is too low. 1 _B The OSC frequency is usable

System Control Unit (SCU)

Field	Bits	Type	Description
PLLHV	8	rh	Oscillator for PLL Valid High Status Bit This bit indicates if the frequency output of OSC is usable for the VCO part of the PLL. This is checked by the Oscillator Watchdog of the PLL. 0 _B The OSC frequency is not usable. Frequency f_{OSC} is too high. 1 _B The OSC frequency is usable
PLLSP	9	rh	Oscillator for PLL Valid Spike Status Bit This bit indicates if the frequency output of OSC is usable for the VCO part of the PLL. This is checked by the Oscillator Watchdog of the PLL. 0 _B The OSC frequency is not usable. Spikes are detected that disturb a locked operation 1 _B The OSC frequency is usable
0	3, [31:10]	r	Reserved Read as 0.

PLLCON0

System PLL configuration register 0.

PLLCON0

PLL Configuration 0 Register

(0714_H)

Reset Value: 0003 0003_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0											FOT	AOT	RES	OSC	PLL	
r											R	REN	LD	RES	PWD	
r											rw	rw	w	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0										OSC	0	FIND	0	VCO	VCO	VCO
r										DISC	r	IS	r	TR	PWD	BYP
r										rw	r	rwh	r	rw	rw	rw

System Control Unit (SCU)

Field	Bits	Type	Description
VCOBYP	0	rw	VCO Bypass 0_B Normal operation, VCO is not bypassed 1_B Prescaler Mode, VCO is bypassed
VCOPWD	1	rw	VCO Power Saving Mode 0_B Normal behavior 1_B The VCO is put into a Power Saving Mode and can no longer be used. Only the Bypass and Prescaler Mode are active if previously selected.
VCOTR	2	rw	VCO Trim Control 0_B VCO bandwidth is operation in the normal range. VCO output frequency is between 260 and 520 MHz for a input frequency between 8 and 16 MHz. 1_B VCO bandwidth is operation in the test range. VCO output frequency is between 260 and 520 MHz for a input frequency between 8 and 16 MHz. Selecting a VCO trim value of one can result in a high jitter but the PLL is still operable.
FINDIS	4	rwh	Disconnect Oscillator from VCO 0_B connect oscillator to the VCO part 1_B disconnect oscillator from the VCO part.
OSCDISDIS	6	rw	Oscillator Disconnect Disable This bit is used to disable the control FINDIS in a PLL loss-of-lock case. 0_B In case of a PLL loss-of-lock bit FINDIS is set 1_B In case of a PLL loss-of-lock bit FINDIS is cleared
PLLPWD	16	rw	PLL Power Saving Mode 0_B Normal behavior 1_B The complete PLL block is put into a Power Saving Mode and can no longer be used. Only the Bypass Mode is active if previously selected.

System Control Unit (SCU)

Field	Bits	Type	Description
OSCREG	17	rw	Oscillator Watchdog Reset This bit controls signal <i>osc_fail_res_i</i> at the PLL module. 0_B The Oscillator Watchdog of the PLL is not reset and remains active 1_B The Oscillator Watchdog of the PLL is reset
RESLD	18	w	Restart VCO Lock Detection Setting this bit will clear bit PLLSTAT.VCOLOCK and restart the VCO lock detection. Reading this bit returns always a zero.
AOTREN	19	rw	Automatic Oscillator Calibration Enable Setting this bit will enable automatic adjustment of the f_{OF1} clock with f_{STDBY} clock used as reference clock. 0_B Disable 1_B Enable
FOTR	20	rw	Factory Oscillator Calibration Force adjustment of the internal oscillator with the firmware defined value. 0_B No effect 1_B Force fixed-value trimming
0	3, 5, [15:7], [31:21]	r	Reserved Read as 0; should be written with 0.

PLLCON1

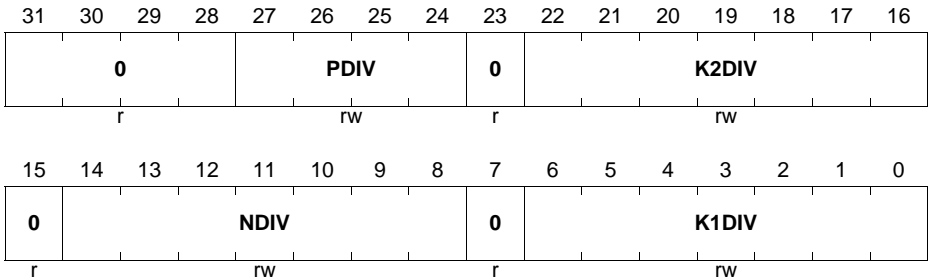
System PLL configuration register 1.

PLLCON1

PLL Configuration 1 Register

(0718_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
K1DIV	[6:0]	rw	K1-Divider Value The value the K1-Divider operates is K1DIV+1.
NDIV	[14:8]	rw	N-Divider Value The value the N-Divider operates is NDIV+1.
K2DIV	[22:16]	rw	K2-Divider Value The value the K2-Divider operates is K2DIV+1.
PDIV	[27:24]	rw	P-Divider Value The value the P-Divider operates is PDIV+1.
0	7, 15, 23, [31:28]	r	Reserved Read as 0; should be written with 0.

PLLCON2

System PLL configuration register 2.

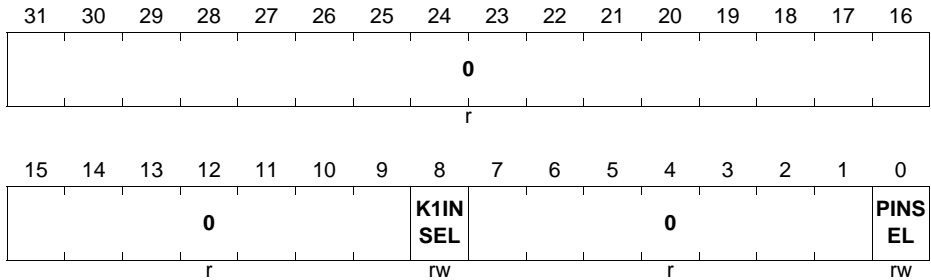
System Control Unit (SCU)

PLLCON2

PLL Configuration 2 Register

(071C_H)

Reset Value: 0000 0001_H



Field	Bits	Type	Description
PINSEL	0	rw	P-Divider Input Selection 0 _B PLL external oscillator selected 1 _B Backup clock f_{ofi} selected
K1INSEL	8	rw	K1-Divider Input Selection 0 _B PLL external oscillator selected 1 _B Backup clock f_{ofi} selected
0	[7:1], [31:9]	r	Reserved Read as 0; should be written with 0.

USBPLLSTAT

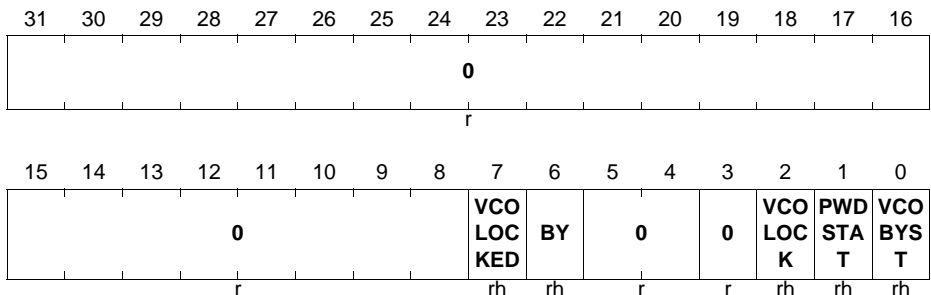
USB PLL Status register.

USBPLLSTAT

USB PLL Status Register

(0720_H)

Reset Value: 0000 0002_H



Field	Bits	Type	Description
VCOPYST	0	rh	VCO Bypass Status 0_B Normal Mode is entered 1_B Prescaler Mode is entered
PWDSTAT	1	rh	PLL Power-saving Mode Status 0_B PLL Power-saving Mode was not entered 1_B PLL Power-saving Mode was entered
VCOLOCK	2	rh	PLL VCO Lock Status 0_B The frequency difference of f_{REF} and f_{DIV} is greater than allowed. The VCO part of the PLL can not lock on a target frequency. 1_B The frequency difference of f_{REF} and f_{DIV} is small enough to enable a stable VCO operation <i>Note: In case of a loss of VCO lock the f_{VCO} goes to the upper boundary of the VCO frequency if the reference clock input is greater than expected.</i> <i>Note: In case of a loss of VCO lock the f_{VCO} goes to the lower boundary of the VCO frequency if the reference clock input is lower than expected.</i>
BY	6	rh	Bypass Mode Status 0_B Bypass Mode is not entered 1_B Bypass Mode is entered. Input f_{OSC} is selected as output f_{PLL} .
VCOLOCKED	7	rh	PLL LOCK Status 0_B PLL not locked 1_B PLL locked
0	3, [5:4], [31:8]	r	Reserved Read as 0.

USBPLLCON

USB PLL configuration register 0.

System Control Unit (SCU)

USBPLLCON

USB PLL Configuration Register

(0724_H)

Reset Value: 0001 0003_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			PDIV				0				RES LD	0	PLL PWD		
r			rw				r				w	r	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	NDIV						0	OSC DISC DIS	0	FIND IS	0	VCO TR	VCO PWD	VCO BYP	
r	rw						r	rw	r	rwh	r	rw	rw	rw	

Field	Bits	Type	Description
VCOBYP	0	rw	VCO Bypass 0 _B Normal operation, VCO is not bypassed 1 _B Prescaler Mode, VCO is bypassed
VCOPWD	1	rw	VCO Power Saving Mode 0 _B Normal behavior 1 _B The VCO is put into a Power Saving Mode
VCOTR	2	rw	VCO Trim Control 0 _B VCO bandwidth is operating in the normal range. VCO output frequency is between 260 and 520 MHz for a input frequency between 8 and 16 MHz. 1 _B VCO bandwidth is operating in the test range. VCO output frequency is between 260 and 520 MHz for a input frequency between 8 and 16 MHz. Selecting a VCO trim value of one can result in a high jitter but the PLL is still operable.
FINDIS	4	rwh	Disconnect Oscillator from VCO 0 _B Connect oscillator to the VCO part 1 _B Disconnect oscillator from the VCO part.

System Control Unit (SCU)

Field	Bits	Type	Description
OSCDISCDIS	6	rw	Oscillator Disconnect Disable This bit is used to disable the control FINDIS in a PLL loss-of-lock case. 0 _B In case of a PLL loss-of-lock bit FINDIS is set 1 _B In case of a PLL loss-of-lock bit FINDIS is cleared
NDIV	[14:8]	rw	N-Divider Value The value the N-Divider operates is NDIV+1.
PLLPWD	16	rw	PLL Power Saving Mode 0 _B Normal behavior 1 _B The complete PLL block is put into a Power Saving Mode. Only the Bypass Mode is active if previously selected.
RESLD	18	w	Restart VCO Lock Detection Setting this bit will clear bit PLLSTAT.VCOLOCK and restart the VCO lock detection. Reading this bit returns always a zero.
PDIV	[27:24]	rw	P-Divider Value The value the P-Divider operates is PDIV+1.
0	3, 5, 7, 15, 17, [23:19], [31:28]	r	Reserved Read as 0; should be written with 0.

CLKMXSTAT

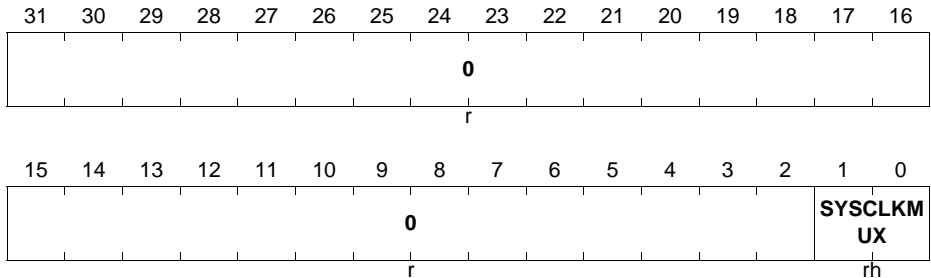
Clock Multiplexer Switching Status

This register shows status of clock multiplexing upon switching from one clock source to another. This register should be checked before disabling any of the multiplexer input clock sources after switching. Bits of this registers indicate which of the corresponding input clocks must not be switched off under any circumstances until indicated as inactive. The clocks sources that are indicated as active are still contributing in driving the output clock from respective multiplexer. This is a side effect of glitch-free clock switching mechanism.

CLKMXSTAT

Clock Multiplexing Status Register (0738_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SYSCLKMUX	[1:0]	rh	Status of System Clock Multiplexing Upon Source Switching Clock sources that are indicated active are still contributing in glitch-free switching x1 _B f _{OFl} clock active 1x _B f _{PLL} clock active
0	[31:2]	r	Reserved

11.11 Interconnections

The system control unit SCU is connected towards the system via AHB-lite bus interface. The AHB bus interface provides register access for monitoring and controlling operation of all sub-modules of the SCU.

Communication Peripherals

12 LED and Touch-Sense (LEDTS)

The LED and Touch-Sense (LEDTS) drives LEDs and controls touch pads used as human-machine interface (HMI) in an application.

Table 12-1 Abbreviations in chapter

Abbreviation	Meaning
LEDTS	LED and Touch-sense
TSD	time slice duration
TFD	time frame duration
TPD	time period duration

12.1 Overview

The LEDTS can measure the capacitance of up to 8 touch pads using the relaxation oscillator (RO) topology. The pad capacitance is measured by generating oscillations on the pad for a fixed time period and counting them. The module can also drive up to 64 LEDs in an LED matrix. Touch pads and LEDs can share pins to minimize the number of pins needed for such applications. This configuration is realized by the module controlling the touch pads and driving the LEDs in a time-division multiplexed manner.

The LEDs in the LED matrix are organized into columns and lines. Every line can be shared between up to 8 LEDs and one touch pad. Certain functions such as column enabling, function selection and control are controlled by hardware. Application software is required to update the LED lines and evaluate the touch pad measurement results.

12.1.1 Features

This device contains 1 LEDTS kernel. Each kernel has an LED driving function and a touch-sensing function.

For the LED driving function, an LEDTS kernel provides these features:

- Selection of up to 8 LED columns; Up to 7 LED columns if touch-sense function is also enabled
- Configurable active time in LED columns to control LED brightness
- Possibility to drive up to 8 LEDs per column, common-anode or common-cathode
- Shadow activation of line pattern for LED column time slice; LED line patterns are updated synchronously to column activation
- Configurable interrupt enable on selected event
- Line and column pins controlled by port SFR setting

For the touch-sensing function, an LEDTS kernel provides these features:

- Up to 8 touch input lines

LED and Touch-Sense (LEDTS)

- Only one pad can be measured at any time; selection of active pad controllable by software or hardware round-robin
- Flexible measurement time on touch pads
- Pin oscillation control circuit with adjustments for oscillation
- 16-bit counter: For counting oscillations on pin.
- Configurable interrupt enable on selected event
- Pin over-rule control for active touch input line (pin)

Note: This chapter refers to the LED or touch-sense pins, e.g. 'pin COL[x]', 'pin TSIN[x]'. In all instances, it refers to the user-configured pin(s) which selects the LED/touch-sense function. Refer to [Section 12.9.5](#) for more elaboration.

Note: This chapter describes the full capability of the LEDTS. The amount of features available on the device is limited by the device pins assigned. Please refer to the [Section 12.11](#) for the pins assigned to the LEDTS.

Table 12-2 LEDTS Applications

Use Case	Application
Non-mechanical switch	HMI
LED feedback	HMI
Simple PWM	PWM

12.1.2 Block Diagram

The LEDTS block diagram is shown in [Figure 12-1](#).

LED and Touch-Sense (LEDTS)

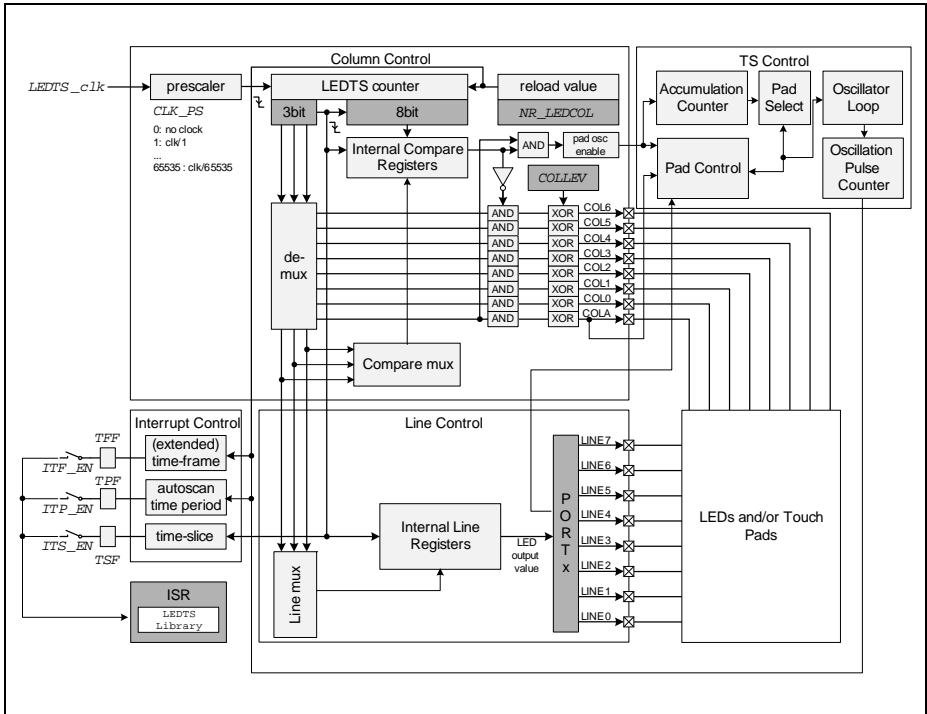


Figure 12-1 LEDTS Block Diagram

12.2 Functional Overview

The same pin can support LED & touch-sense functions in a time-multiplexed manner. LED mode or touch-sense mode can be enabled by hardware for respective function controls.

Time-division multiplexing is done by dividing the time domain into time slots. This basic time slot is called a **time slice**. In one time slice, one LED column is activated or the capacitance of one touch pad is measured.

A **time frame** is composed of 1 or more time slices, up to a maximum of eight. There is one time slice for every LED column enabled. If only LED function is enabled, a time frame can compose up to 8 LED time slices. However, if touch sense function is enabled, the last time slice in every time frame is reserved for touch-sense function. This reduces the maximum number of time slices that can be used for LED function in each time frame to 7. Only one time slice is used for touch sense function in every time frame. This is regardless of the number of touch pads enabled.

In each time slice used for LED function, only one LED column is enabled at a time. In the time slice reserved for touch-sense function, oscillations are enabled and measured on the pin which is activated. No LED column is active during this time slice. A touch pad input line (TSIN[x] pin) is active when its pad turn is enabled. If more than one touch pad input lines are enabled, the enabling and measurement on the touch pads is performed in a round robin manner. Only one touch pad is measured in every time frame.

The resolution of oscillation measurement can be increased by accumulating oscillation counts on each touch input line. When enabled by configuration of “Accumulate Count” (ACCCNT), the pad turn can be extended on consecutive time frames by up to 16 times. This also means that the same touch pad will be measured in consecutive time frames. This control will be handled by hardware. Otherwise it is also possible to enable for software control where the active pad turn is fully under user control.

The number of consecutive time frames, for which a pad turn has been extended, forms an **extended time frame**. When touch-sense function is enabled for automatic hardware pad turn control, several (extended) time frames make up one **autoscan time period** where all pad turns are completed. The time slice duration is configured centrally for the LED and/or touch-sense functions, using the LEDTS-counter. Refer to the description in [Section 12.3](#), [Section 12.9.3](#) and [Figure 12-4](#).

If enabled, a time slice interrupt is triggered on overflow of the 8LSBs of the LEDTS-counter for each new time slice started. The (extended) time frame interrupt may also be enabled. It is triggered on (the configured counts of) overflow of the whole LEDTS-counter. The autoscan time period interrupt may also be enabled. However, this interrupt will require that the hardware pad turn control is enabled. It is triggered when hardware completes the last pad turn on the highest enabled touch input line TSIN[NR_TSIN].

The column activation and pin oscillation duty cycles can be configured for each time slice. This allows the duration of activation of LED columns and/or touch-sense

LED and Touch-Sense (LEDTS)

oscillation counting to be flexible. This is also how the relative brightness of the LEDs can be controlled. In case of touch pads, the activation time is called the oscillation window.

Figure 12-2 shows an example for an LED matrix configuration with touch pads. The configuration in this example is 8 X 4 LED matrix with 4 touch input lines enabled in sequence by hardware. Here no pad turn is extended by ACCCNT, so four time frames complete an autoscan time period.

In the time slice interrupt, software can:

- set up line pattern for next time slice
- set up compare value for next time slice
- evaluate current function in time slice (especially for analysis/debugging)

Refer to **Section 12.9.1** for **Interpretation of Bit Field FNCOL** to determine the currently active time slice.

The (extended) time frame interrupt indicates one touch input line TSIN[x] has been sensed (once or number of times in consecutive frames), application-level software can, for example:

- start touch-sense processing (e.g. filtering) routines and update status
- update LED display data to SFR

In the autoscan time period interrupt which indicates all touch-sense input TSIN[x] have been scanned one round, application-level software can:

- evaluate touch detection result & action
- update LED display data to SFR

LED and Touch-Sense (LEDTS)

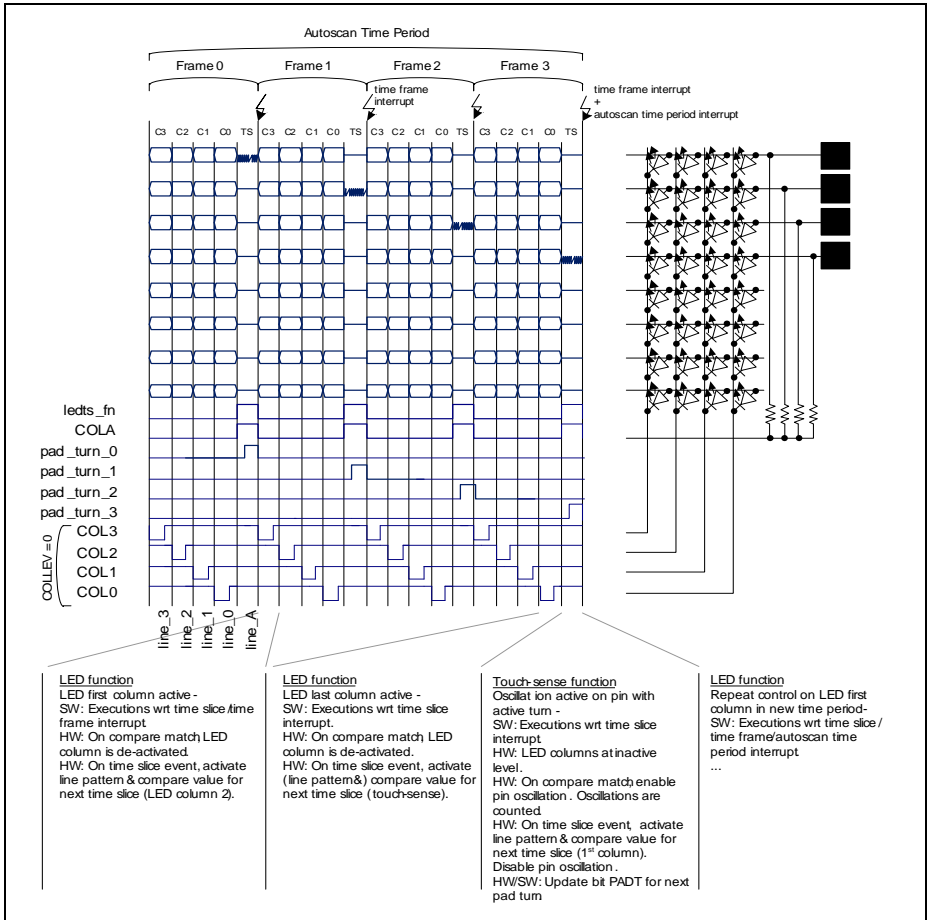


Figure 12-2 Time-Multiplexed LEDTS Functions on Pin (Example)

12.3 LED Drive Mode

LED driving is supported mainly for LED column selection and line control. At one time, only one column is active. The corresponding line level at high or low determines if the associated LED on column is lit or not. Up to eight columns are supported (if only LED function is enabled), and up to eight LEDs can be controlled per column.

With direct LED drive, adjustment of luminence for different types of LEDs with different forward voltages is supported. A compare register for LEDTS-counter is provided so that the duty cycle for column enabling per time slice can be adjusted. The LED column is enabled from the beginning of the time slice until compare match event. For 100% duty cycle for LED column enable in time slice, the compare value should be set to $FF_{H,}$. If the compare value is set to $00_{H,}$, the LED column will stay at passive level during the time slice. The internal compare register for each time slice is updated by shadow transfer from their corresponding compare SFR. This takes place automatically at the beginning of each time slice, refer to **Figure 12-3**.

Updating of LED line pattern (LED enabling) per column (time slice) is performed via a similar shadow transfer mechanism, as illustrated in **Figure 12-3**. This shadow transfer of the corresponding line pattern to the internal line SFR takes place automatically at the beginning of each new time slice.

Note: Any write to any compare or line SFR within the time slice does not affect the internal latched configuration of current time slice.

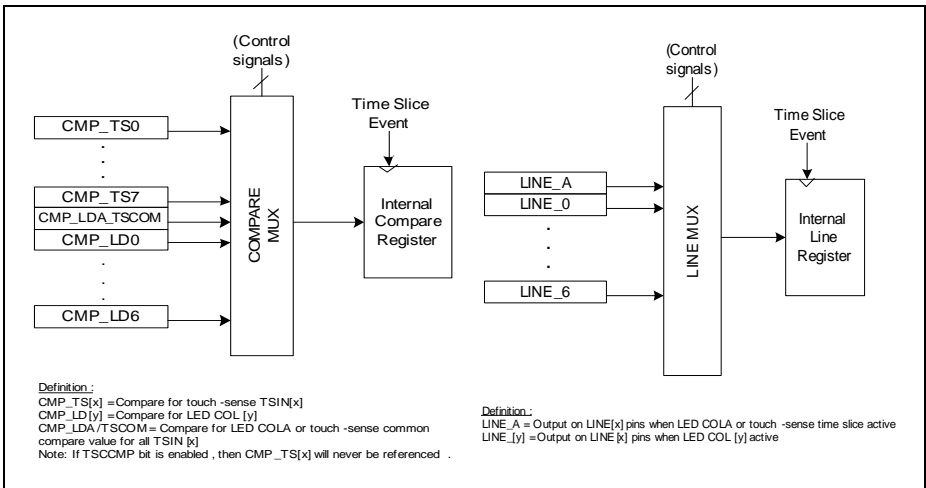


Figure 12-3 Activate Internal Compare/Line Register for New Time Slice

When the LEDTS-counter is first started (enable input clock by CLK_PS), a shadow transfer of line pattern and compare value is activated for the first time slice (column).

LED and Touch-Sense (LEDTS)

A time slice interrupt can be enabled. A new time slice starts on the overflow of the 8LSBs of the LEDTS-counter.

Figure 12-4 shows the LED function control circuit. This circuit also provides the control for enabling the pad oscillator. A 16-bit divider provides pre-scale possibilities to flexibly configure the internal LEDTS-counter count rate, which overflows in one time frame. During a time frame comprising a configurable number of time slices, the configured number of LED columns are activated in sequence. In the last time slice of the time frame, touch-sense function is activated if enabled.

The LEDTS-counter is started when bit CLK_PS is set to any value other than 0 and either the LED or touch-sense function is enabled. It does not run when both functions are disabled. To avoid over-write of function enable which disturbs the hardware control during LEDTS-counter running, the TS_EN and LD_EN bits can only be modified when bit CLK_PS = 0. It is nonetheless possible to set the bits TS_EN and LD_EN in one single write to SFR **GLOBCTL** when setting CLK_PS from 0 to 1, or from 1 to 0.

When started, the counter starts running from a reset/reload value based on enabled function(s): 1) the number of columns (bit-field NR_LED COL) when LED function is enabled, 2) add one time slice at end of time frame when touch-sense function is enabled. The counter always counts up and overflows from $7FF_H$ to the reload value which is the same as the reset value. Within each time frame, the sequence of LED column enabling always starts from the most-significant enabled column (column with highest numbering).

To illustrate this point, in the case of four LED columns and one touch pad input enabled, the column enabling sequence will be in as follows:

- Start with COL3,
- followed by COL2,
- followed by COL1,
- followed by COL0,
- then COLA for touch sense function.

If the touch-sense function is not enabled, COLA will be available for LED function as the last LED column time slice of a time frame. The column enabling sequence will then be as follows:

- Start with COL2,
- followed by COL1,
- followed by COL0,
- then COLA.

LED and Touch-Sense (LEDTS)

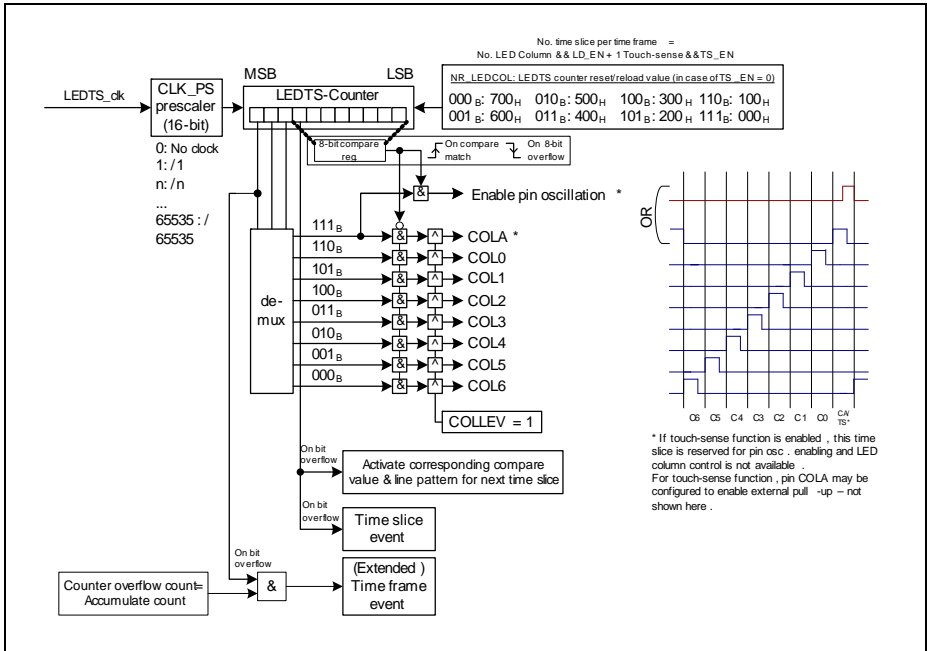


Figure 12-4 LED Function Control Circuit (also provides pad oscillator enable)

In [Section 12.9.3](#), the time slice duration and formulations for LEDTS related timings are provided.

12.3.1 LED Pin Assignment and Current Capability

One LED column pin is enabled within each configured time slice duration to control up to eight LEDs at a time. The assignment of COL[x] to pins is configurable to provide options for application pin usage. The current capability of device pins is also a consideration factor for deciding pin assignment to LED function.

The product data-sheet provides data for all I/O parameters relevant to LED drive.

12.4 Touch-Sense Mode

Figure 12-5 shows the pin oscillation control unit, which is integrated with the standard PORTS pad. The active pad turn (pad_turn_x) for a touch input line is defined as the time duration within the touch-sense time slice (COL A) where the TS-counter is counting oscillations on the TSIN[x] pin. In the case of hardware pad turn control (default), the same TS-counter is connected sequentially on enabled touch input lines to execute a round-robin touch-sensing of TSIN[x] pins.

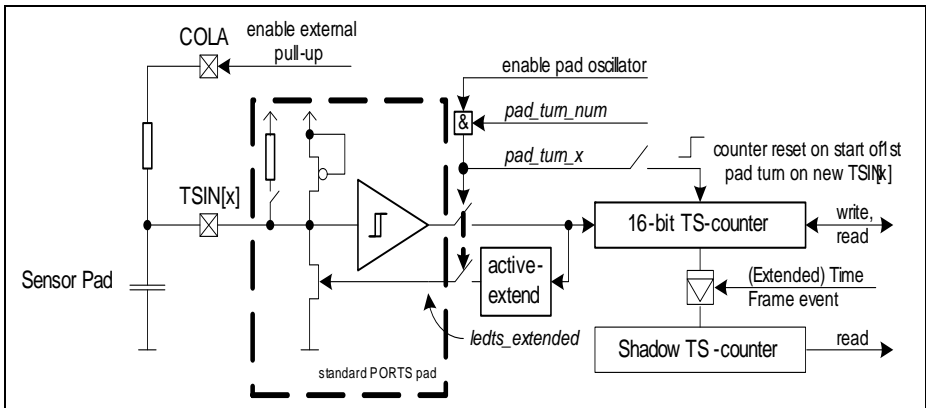


Figure 12-5 Touch-Sense Oscillator Control Circuit

In an example of four touch input lines enabled, the sequence order of touch-sense time slice (COL A) in sequential frames is as follows:

- Always starts with TSIN0,
- followed by TSIN1 in touch-sense time slice of next frame,
- followed by TSIN2 in next touch-sense time slice,
- then TSIN3, and repeat cycle.

It is possible to enable the touch-sense time slice on the same touch input line for consecutive frames (up to 16 times), to accumulate the oscillation count in TS-counter (see **Figure 12-6**). To illustrate this point, in the same case of four touch input lines enabled, and 2 accumulation counts configured, is as follows:

- Always starts with TSIN0,
- followed by TSIN0 again in touch-sense time slice of next frame,
- followed by TSIN1 in touch-sense time slice of next frame,
- followed by TSIN1 again in touch-sense time slice of next frame,
- followed by TSIN2 in touch-sense time slice of next frame,
- followed by TSIN2 again in touch-sense time slice of next frame,
- followed by TSIN3 in touch-sense time slice of next frame,
- then TSIN3 again, and repeat cycle.

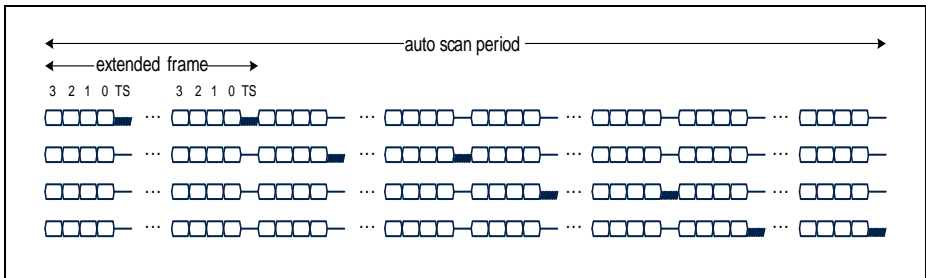


Figure 12-6 Hardware-Controlled Pad Turns for Autoscan of Four TSIN[x] with Extended Frames

There is a 16-bit TS-counter register and there is a 16-bit shadow TS-counter register. The former is both write- and read-accessible, while the latter is only read-accessible. The actual TS-counter counts the latched number of oscillations and can only be written when there is no active pad turn. The content of the TS-counter is latched to the shadow register on every (extended) time frame event. Reading from the shadow register therefore shows the latest valid oscillation count on one TSIN[x] input, ensuring for the application SW there is at least one time slice duration to get the valid oscillation count and meanwhile the actual TS-counter could continually update due to enabled pin oscillations in current time slice.

The TS-counter and shadow TS-counter have another user-enabled function on (extended) time frame event, which is to validate the counter value differences. When this function is enabled by the user and in case the counter values do not differ by 2^n LSB bits ('n' is configurable), the (extended) time frame interrupt request is gated (no interrupt) and the time frame event flag TFF is not set. This gating is on top of the time frame interrupt enable/disable control.

The TS-counter may be enabled for automatic reset (to 00_{H}) on the start of a new pad turn on the next TSIN[x], i.e. resets in the first touch-sense time slice of each (extended) time frame. Bit TSCTROVF indicates that the counter has overflowed. Alternatively, it can be configured such that the TS-counter stops counting in touch-sense time slice(s) of the same extended frame when the count value saturates, i.e. does not overflow & stops at FFFF_{H} . In this case, the TS-counter starts running again only in a new (extended) frame on the start of a new pad turn on the next TSIN[x].

A configurable pin-low-level active extension is provided for adjustment of oscillation per user system. The extension is active during the discharge phase of oscillation, and can be configured to be extended by a number of peripheral clocks. This function is very useful if there is a series resistor between the pin and the touch pad which makes the discharge slower. [Figure 12-7](#) illustrates this function.

The configuration of the active touch-sense pin TSIN[x] is over-ruled by hardware in the active duration to enable oscillations, reference [Section 12.9.5](#). In particular, the weak

LED and Touch-Sense (LEDTS)

internal pull-up enable over-rule can be optionally de-activated (correspondingly internal pull-down disable over-rule is also de-activated; PORTS pin SFR setting for pull applies instead), such as when the user system utilize external resistor for pull-up instead. In the whole duration of the touch-sense time slice, COLA is activated high. This activates a pull-up via an external resistor connected to pin COLA. This configuration provides some flexibility to adjust the pin oscillation rate for adaptation to user system.

The touch-sense function is time-multiplexed with the LED function on enabled LINE[x]/TSIN[x] pins. During the touch-sense time slice for the other TSIN pins which are not on active pad turn, the corresponding LINE[x] output remains active. It is recommended that software takes care to set the line bits to 1 to avoid current sink from pin COLA.

The touch-sense function is active in the last time slice of a time frame. Refer to [Section 12.2](#), and [Section 12.3](#) for more details on time slice allocation and configuration.

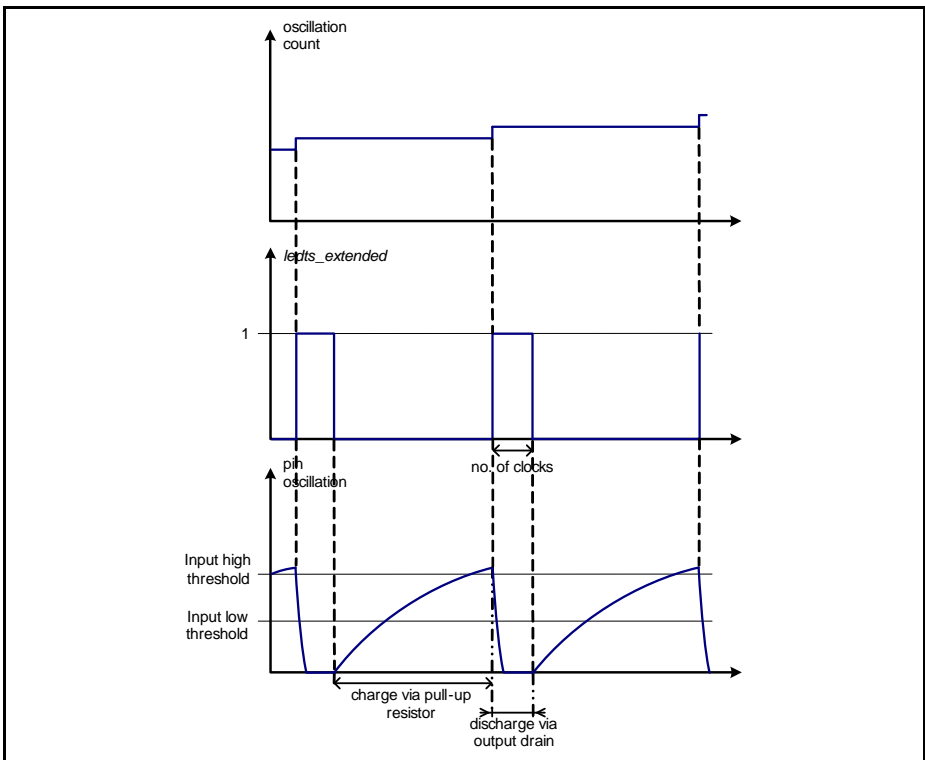


Figure 12-7 Pin-Low-Level Extension Function

LED and Touch-Sense (LEDTS)

The oscillation is enabled on the pin with valid turn for a configurable duration. A compare value provides the means to adjust the duty cycle within the time slice. The pin oscillation is enabled (TS-counter is counting) only on compare match until the end of the time slice. The time interval, in which the TS-counter is counting, is called the oscillation window. For a 100% duty cycle, the compare value has to be set to 00_{H} . In this case, the oscillation window fills in the entire time slice. Setting the compare value to FF_{H} results in no pin oscillation in time slice.

The time slice interrupt, (extended) time frame interrupt and/or autoscan time period interrupt may be enabled as required for touch-sense control.

12.4.1 Finger Sensing

When a finger is placed on the sensor pad, it increases the pin capacitance and frequency of oscillation on pin is reduced. The pin oscillation frequency without finger touch is typically expected to be in the range of 100 kHz to 5 MHz. Various factors affect the oscillation frequency including the size of touch pad, ground planes around and below the pad, the material and thickness of the overlay cover, the trace length and the individual pin itself (every pin has a different pull-up resistance).

In a real-world application, the printed circuit board (PCB) will not be touched directly. Instead, there is usually some sort of a transparent cover material, like a piece of plexiglass sheet, glued onto the PCB. In most of these applications, the oscillation frequency will change by about 2-10% when touched. This change in oscillation frequency can be considered to be very small, and therefore, further signal processing is necessary for reliable detection. Typically, this processing takes the form of a moving average calculation. It is never recommended to try to detect touches based on the raw oscillation count value.

As described in above section, some flexibility is provided to adjust the oscillation frequency in the user system: 1) Configurable pin low-level active extension, 2) Alternative enabling of external pull-up with resistance selectable by user. With a configurable time slice duration, the software can configure the duration of the active pad turn (adjustable within time slice using compare function) and set a count threshold for oscillations to detect if there is a finger touch or not.

To increase touch-sensing oscillation count accuracy, the input clock to LEDTS kernel should be set as high as possible.

12.5 Operating both LED Drive and Touch-Sense Modes

It is possible to enable both LED driving and touch-sense functions in a single time frame. If both functions are enabled, up to 7 time slices are configurable for the LED function, and the last time slice is reserved for touch-sensing function.

The touch-sense function is time-multiplexed with the LED function on enabled LINE[x]/TSIN[x] pins. During the touch-sense time slice (COLA), the corresponding LINE[s] output remains active for the other TSIN pins which are not on active pad turn. In a typical application, COLA is not used and the oscillation is generated by the internal pad structure only. The bits in LINE_A will determine whether the pads, that are not being measured in the given COLA time slice, have a floating or 0V value. This setting usually has a serious effect on the sensitivity and noise robustness of the touch pads.

Refer to [Section 12.2](#) and [Section 12.3](#) for more details on time slice allocation and configuration.

12.6 Service Request Processing

There are three interrupts triggered by LEDTS kernel, all assigned on same node: 1) time slice event, 2) (extended) time frame event, 3) autoscan time period event. The flags are set on event or when CLK_PS is set from 0 regardless of whether the corresponding interrupt is enabled or not. When enabled, the event (including setting of CLK_PS from 0) activates the SR0 interrupt request from the kernel.

[Table 12-3](#) lists the interrupt event sources from the LEDTS, and the corresponding event interrupt enable bit and flag bit.

Table 12-3 LEDTS Interrupt Events

Event	Event Interrupt Enable Bit	Event Flag Bit
Start of Time Slice	GLOBCTL.ITS_EN	EVFR.TSF
Start of (Extended) Time Frame ¹⁾	GLOBCTL.ITF_EN	EVFR.TFF
Start of Autoscan Time Period	GLOBCTL.ITP_EN	EVFR.TPF

1) In case of consecutive pad turns enabled on same TSIN[x] pin by ACCCNT bit-field, interrupt is not triggered on a time frame – but on the extended time frame.

[Table 12-4](#) shows the interrupt node assignment for each LEDTS interrupt source.

Table 12-4 LEDTS Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Node ID
Start of Time Slice	LEDTS0.SR0	LEDTS0.SR0	102
Start of (Extended) Time Frame			
Start of Autoscan Time Period			

12.7 Debug Behavior

The LEDTS timers/counters LEDTS-counter and TS-counter can be enabled (together) for suspend operation when debug mode becomes active (indicated by HALTED signal from CPU).

In debug suspend mode, write and read access are possible. Writing to bit fields that require **GLOBCTL.CLK_PS = 0** will still be ignored.

At the onset of debug suspend, these counters stop counting (retains the last value) for the duration of the device in debug mode. The function that was active in current time slice on the onset of debug suspend, continues to be active. When debug suspend is revoked, the kernel would resume operation according to latest SFR settings.

12.8 Power, Reset and Clock

The LEDTS kernel is clocked and accessible on the peripheral bus frequency. User should set up consistent time-slice durations for correct function by ensuring a constant frequency input clock when the kernel is in operation. It is recommended to set the input clock ledts_clk to highest frequency where possible, for optimal touch-sensing accuracy.

Kernel is in operation in active mode except power-down modes where touch-sensing and LED functions are not available.

12.9 Initialisation and System Dependencies

This section provides hints for enabling the LEDTS functions and using them.

12.9.1 Function Enabling

It is recommended to set up all configuration for the LEDTS in all SFRs before write **GLOBCTL** SFR to enable and start LED and/or touch-sense function(s).

Note: SFR bits especially affecting the LEDTS-counter configuration for LED/touch-sense function can only be written when the counter is not running i.e. CLK_PS = 0. Refer to SFR bit description [Section 12.10](#).

Enable LED Function Only

To enable LED function only: set LD_EN, clear TS_EN.

Initialization after reset:

```
GLOBCTL = 0bXXXXXXXX XXXXXXXX XXX00000 0000XX10;
//set LD_EN and start LEDTS-counter on prescaled clock
//(CLK_PS != 0)
```

Re-configuration during run-time:

```
GLOBCTL &= 0x0000X00X;//stop LEDTS-counter by clearing prescaler
GLOBCTL = 0bXXXXXXXX XXXXXXXX XXX00000 0000XX10;
```

Enable Touch-Sense Function Only

To enable touch-sense function only: clear LD_EN, set TS_EN.

Initialization after reset:

```
GLOBCTL = 0bXXXXXXXX XXXXXXXX XXX00000 0000XX01;
//set TS_EN and start LEDTS-counter on prescaled clock
//(CLK_PS != 0)
```

Re-configuration during run-time:

```
GLOBCTL &= 0x0000X00X;//stop LEDTS-counter by clearing prescaler
GLOBCTL = #0bXXXXXXXX XXXXXXXX XXX00000 0000XX01;
```

Enable Both LED and Touch-Sense Function

To enable both functions: set LD_EN, set TS_EN.

Initialization after reset:

```
GLOBCTL = 0bXXXXXXXX XXXXXXXX XXX00000 0000XX11;
//set TS_EN and start LEDTS-counter on prescaled clock
//(CLK_PS != 0)
```

Re-configuration during run-time:

```
GLOBCTL &= 0x0000X00X;//stop LEDTS-counter by clearing prescaler
GLOBCTL = 0bXXXXXXXX XXXXXXXX XXX00000 0000XX11;
```

12.9.2 Interpretation of Bit Field FNCOL

The interpretation of the FNCOL bit field can be handled by software. The following example where six time slices are enabled (per time frame), with five LED columns and touch-sensing enabled, illustrates this ([Table 12-5](#)).

The FNCOL bit field provides information on the function/column active in the previous time slice. With this information, software can determine the active function/column in

LED and Touch-Sense (LEDTS)

current time slice and prepare the necessary values (to be shadow-transferred) valid for the next time slice.

Referring to the example below, when the FNCOL bit field is 111_B, it can be derived that the touch-sensing function/column was active in the previous time slice and therefore the current active column is LED COL[4]. Hence, the software can update the shadow line and compare registers for LED COL[3] so that these changes will be reflected when LED COL[3] gets activated in the next time slice.

Table 12-5 Interpretation of FNCOL Bit Field

FNCOL	Active Function / Column in Current Time Slice	SW Prepare via “Shadow” Registers for Function / Column of Next Time Slice
111 _H	LED COL[4]	LED COL[3]
010 _H	LED COL[3]	LED COL[2]
011 _H	LED COL[2]	LED COL[1]
100 _H	LED COL[1]	LED COL[0]
101 _H	LED COL[0]	Touch Input Line TSIN[PADT]
110 _H	Touch Input Line TSIN[PADT]	LED COL[4]

12.9.3 LEDTS Timing Calculations

LEDTS main timing or duration formulation are provided in following.

Count-Rate (CR):

$$CR = (fCLK) \div (PREscaler) \tag{12.1}$$

where fCLK = LEDTS module input clock; PREscaler = **GLOBCTL.CLK_PS**

Time slice duration (TSD):

$$TSD = 2^8 \div (CR) \tag{12.2}$$

Time frame duration (TFD):

$$TFD = (\text{Number of time slices}) \times TSD \tag{12.3}$$

Extended TFD:

$$\text{ExtendedTFD} = \text{ACCCNT} \times \text{TFD} \tag{12.4}$$

where ACCCNT = **FNCTL.ACCCNT**

Autoscan time period duration (TPD):

$$\text{TPD} = (\text{Number of touch-sense inputs TSIN[x]}) \times \text{TFD} \tag{12.5}$$

LED and Touch-Sense (LEDTS)

LED drive active duration:

$$\text{LED Drive Active Duration} = \text{TSD} \times \text{Compare_VALUE} \div 2^8 \quad (12.6)$$

Touch-sense drive active duration:

$$\text{Touch-sense Drive Active Duration} = \text{TSD} \times (2^8 - \text{Compare_VALUE}) \div 2^8 \quad (12.7)$$

12.9.4 Time-Multiplexed LED and Touch-Sense Functions on Pin

Some hints are provided regarding the time-multiplexed usage of a pin for LED and touch-sense function:

- The maximum number of LED columns = 7 when touch-sense function is also enabled.
- If enabled by pin, COLA outputs HIGH to enable external R (resistor) as pull-up for touch-sense function.
- During touch-sense time slice, it is recommended to set LED lines to output LOW.
- During LED time slice, COLA outputs LOW and will sink current if connected lines output HIGH.
- The effective capacitance for each TSIN[x] depends largely on what is connected to the pin and the application board layout. All touch-pads for the application should be calibrated for robust touch-detection.

12.9.5 LEDTS Pin Control

The user may flexibly assign pins as provided by PORTS SFRs, for the LEDTS functions:

- COL[x] (for LED column control)
- LINE[x] (for LED line control)
- TSIN[x] (for touch-sensing)

Refer also to [Section 12.3](#) for more considerations with regards to which COL[x] and/or LINE[x]/TSIN[x] will be active based on user configuration.

User code must configure the assigned LED pin PORTS SFR alternate output selection for the LED function, see [Table 12-6](#) and [Figure 12-8](#).

For the touch-sense function, it is also required to configure the PORTS SFRs to enable the hardware function on TSIN[x] pin (similarly alternate output COLA). However, the LEDTS will provide some pin over-rule controls to the assigned touch-sense pin with active pad turn, see [Table 12-6](#) and [Figure 12-8](#).

LED and Touch-Sense (LEDTS)

Table 12-6 LEDTS Pin Control Signals

Function	ld/ts_en	ledts_fn	Pin	Control of Assigned Pin
LED column	LD_EN = 1	0 = LED	Enable COL[x]; Passive level on COL[the rest]. If TS_EN = 1, COLA = 0.	PORTS SFR setting
LED line	LD_EN = 1	0 = LED	LINE[x] = internal line register value latched from bit field LINE_[x]	PORTS SFR setting
Touch-sense	TS_EN = 1	1 = Touch-sense	Enable TSIN[x] for oscillation. All other TSIN pins output line value. Passive level on COL[the rest] except COLA = 1.	Hardware over-rule on pad_turn_x ¹⁾ for active duration: - Enable pull-up, disable pull-down (pull over-rule can be disabled by bit EPULL) - Set to output mode (both input, output stages enabled) - Enable open-drain

1) For the other pad inputs not on turn, there is no HW over-rule which means the PORTS SFR setting is active.

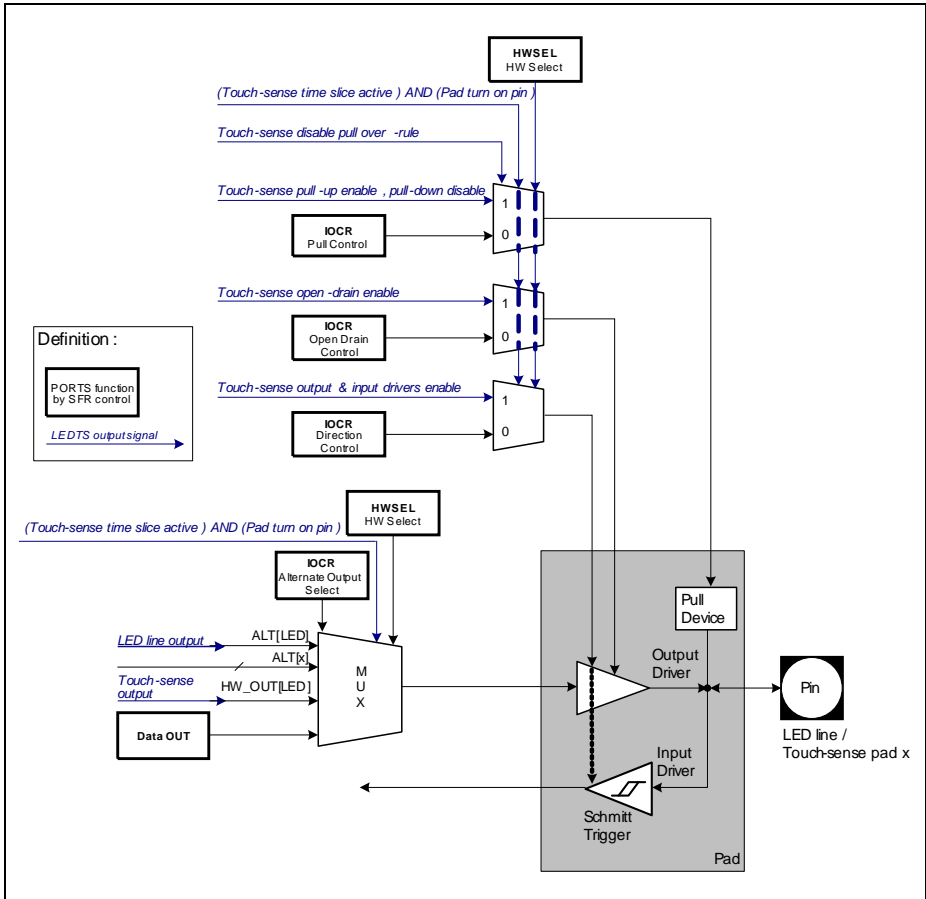


Figure 12-8 Over-rule Control on Pin for Touch-Sense Function

12.9.6 Software Hints

This section provides some useful software hints:

- Compare value 00_H enables oscillation for the full duration of the time slice, whereas FF_H disables oscillation.
- In order to maximize the resolution of the oscillation window, compare value should be selected to maximize the oscillation count without overflowing the TS-counter.
- Valid pad detection period (the time required to detect a valid touch on a pad) can be extended by:

LED and Touch-Sense (LEDTS)

- enabling dummy LED columns (without assigning/setting the LED column pins)
- selecting bigger pre-scale factor (**GLOBCTL.CLK_PS**)
- accumulating the number of pad oscillations (**FNCTL.ACCCNT**)
- Valid pad detection period can be reduced by:
 - selecting smaller pre-scale factor (**GLOBCTL.CLK_PS**)
 - reducing the number of accumulations for pad oscillations (**FNCTL.ACCCNT**)

12.9.7 Hardware Design Hints

This section provides some hardware design hints:

- Touch button oscillation frequency changes when the value of the external pull-up resistor (connected to COLA pin) changes. This results in different sensitivity of the touch button as well as the crosstalk between the adjacent touch buttons.
 - A suitable pull-up resistor should be selected to balance the sensitivity of the touch button and the accuracy of the detection.
- The presence of LEDs modifies the equivalent capacitance for a touch pad. A larger number of LEDs connected to a touch pad will increase the self-capacitance of the pad. This makes the pad less sensitive.
- If possible, LEDs should be located near to the touch pads, to reduce the additional parasitic capacitance introduced by the traces.

12.10 Registers

Registers Overview

The absolute register address is calculated by adding:

Module Base Address + Offset Address

Table 12-7 Registers Address Space

Module	Base Address	End Address	Note
LEDTS0	4801 0000 _H	4801 00FF _H	

The prefix “**LEDTSx_**” is added to the register names in this table to indicate they belong to the LEDTS kernel.

Note: Register bits marked “w” always deliver 0 when read.

Access rights within the address range of an LEDTS kernel:

- Read or write access to defined register addresses: U, PV
- Accesses to empty addresses: nBE

Table 12-8 Register Overview of LEDTS

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
ID	Module Identification Register	0000 _H	U, PV	U, PV	Page 12-23
GLOBCTL	Global Control Register	0004 _H	U, PV	U, PV	Page 12-24
FNCTL	Function Control Register	0008 _H	U, PV	U, PV	Page 12-26
EVFR	Event Flag Register	000C _H	U, PV	U, PV	Page 12-30
TSVAL	Touch-Sense TS-Counter Value	0010 _H	U, PV	U, PV	Page 12-31
LINE0	Line Pattern Register 0	0014 _H	U, PV	U, PV	Page 12-32
LINE1	Line Pattern Register 1	0018 _H	U, PV	U, PV	Page 12-33
LDCMP0	LED Compare Register 0	001C _H	U, PV	U, PV	Page 12-33
LDCMP1	LED Compare Register 1	0020 _H	U, PV	U, PV	Page 12-34
TSCMP0	Touch-Sense Compare Register 0	0024 _H	U, PV	U, PV	Page 12-35

Table 12-8 Register Overview of LEDTS

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
TSCMP1	Touch-Sense Compare Register 1	0028 _H	U, PV	U, PV	Page 12-35
Reserved	Reserved	002C _H - 1FFC _H	nBE	nBE	

12.10.1 Registers Description

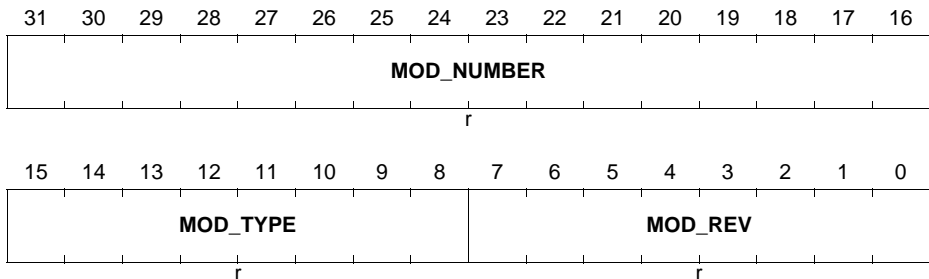
The LEDTS SFRs are organized into registers for global initialization control, functional control comprising TS-counter value, line pattern and compare value registers.

LEDTSx_ID

The module identification register indicate the function and the design step of the peripheral.

ID

Module Identification Register (00_H) **Reset Value:00AB C0XX_H**



Field	Bits	Type	Description
MOD_REV	[7:0]	r	Module Revision Number MOD_REV defines the revision number. The value of a module revision starts with a 00 _H (first revision).
MOD_TYPE	[15:8]	r	Module Type This bit field is C0 _H . It defines the module as a 32-bit module.

LED and Touch-Sense (LEDTS)

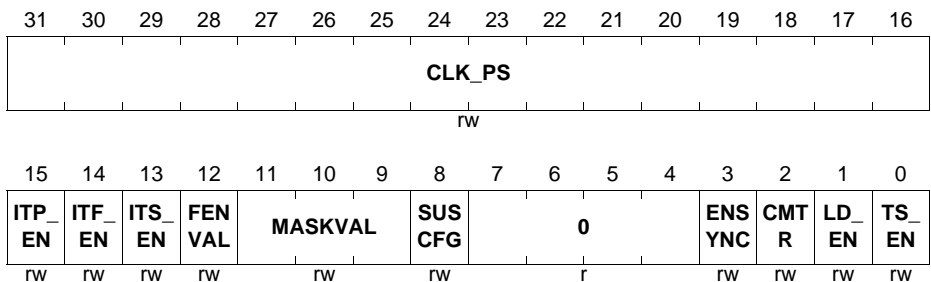
Field	Bits	Type	Description
MOD_NUMBER	[31:16]	r	Module Number Value This bit field defines the module identification number.

GLOBCTL

The GLOBCTL register is used to initialize the LEDTS global controls.

GLOBCTL

Global Control Register (04_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TS_EN¹⁾	0	rw	Touch-Sense Function Enable Set to enable the kernel for touch-sense function control when CLK_PS is set from 0.
LD_EN¹⁾	1	rw	LED Function Enable Set to enable the kernel for LED function control when CLK_PS is set from 0.
CMTR¹⁾	2	rw	Clock Master Disable 0 _B Kernel generates its own clock for LEDTS-counter based on SFR setting 1 _B LEDTS-counter takes its clock from another master kernel If this bit is set, the bit field GLOBCTL.CLK_PS needs to be set to a non-zero value to activate the kernel, even though it is taking its clock from the master kernel.

LED and Touch-Sense (LEDTS)

Field	Bits	Type	Description
ENSYNC ¹⁾	3	rw	Enable Autoscan Time Period Synchronization 0 _B No synchronization 1 _B Synchronization enabled on Kernel0 autoscan time period
SUSCFG	8	rw	Suspend Request Configuration 0 _B Ignore suspend request 1 _B Enable suspend according to request This bit is restored to default with Debug Reset.
MASKVAL	[11:9]	rw	Mask Number of LSB Bits for Event Validation This defines the number of LSB bits to mask for TS-counter and shadow TS-counter comparison when Time Frame validation is enabled. 0 _D Mask LSB bit 1 _D Mask 2 LSB bits ... 7 _D Mask 8 LSB bits
FENVAL	12	rw	Enable (Extended) Time Frame Validation When enabled, TS-counter and shadow TS-counter values are compared to validate a Time Frame event for set flag and interrupt request. When validation fail, TFF flag does not get set and interrupt is not requested. 0 _B Disable 1 _B Enable
ITS_EN	13	rw	Enable Time Slice Interrupt 0 _B Disable 1 _B Enable
ITF_EN	14	rw	Enable (Extended) Time Frame Interrupt 0 _B Disable 1 _B Enable
ITP_EN	15	rw	Enable Autoscan Time Period Interrupt 0 _B Disable 1 _B Enable (valid only for case of hardware-enabled pad turn control)

LED and Touch-Sense (LEDTS)

Field	Bits	Type	Description
CLK_PS	[31:16]	rw	<p>LEDTS-Counter Clock Pre-Scale Factor The constant clock input is prescaled according to setting.</p> <p>0_D No clock 1_D Divide by 1 ... 65535_D Divide by 65535</p> <p>This bit can only be set to any other value (from 0) provided at least one of touch-sense or LED function is enabled. The LEDTS-counter starts running on the input clock from reset/reload value based on enabled function(s) (and NR_LEDCOL). Refer Section 12.3 for details.</p> <p>When this bit is clear to 0 from other value, the LEDTS-counter stops running and resets.</p>
0	[7:4]	r	<p>Reserved Read as 0; should be written with 0.</p>

1) This bit can only be modified when bit CLK_PS = 0.

FNCTL

The FNCTL control register provides control bits for the LED and Touch Sense functions.

FNCTL

Function Control Register

(08_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NR_LEDCOL		COL LEV	NR_TSIN		TSC TRS AT	TSC TRR	TSOEXT	TSC CMP	ACCCNT						
rw		rw	rw		rw	rw	rw	rw	rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								FNCOL	EPU LL	PAD TSW	PADT				
r								rh	rw	rw	rwh				

LED and Touch-Sense (LEDTS)

Field	Bits	Type	Description
PADT	[2:0]	rw	<p>Touch-Sense TSIN Pad Turn</p> <p>This is the TSIN[x] pin that is next or currently active in pad turn. When PADTSW = 0, the value is updated by hardware at the end of touch-sense time slice. Software write is always possible. However, PADT must not be written during an active pad turn.</p> <p>0_D TSIN0 ... 7_D TSIN7</p>
PADTSW¹⁾	3	rw	<p>Software Control for Touch-Sense Pad Turn</p> <p>0_B The hardware automatically enables the touch-sense inputs in sequence round-robin, starting from TSIN0.</p> <p>1_B Disable hardware control for software control only. The touch-sense input is configured in bit PADT.</p>
EPULL	4	rw	<p>Enable External Pull-up Configuration on Pin COLA</p> <p>When set, the internal pull-up over-rule on active touch-sense input pin is disabled.</p> <p>0_B HW over-rule to enable internal pull-up is active on TSIN[x] for set duration in touch-sense time slice. With this setting, it is not specified to assign the COLA to any pin.</p> <p>1_B Enable external pull-up: Output 1 on pin COLA for whole duration of touch-sense time slice.</p> <p><i>Note: Independent of this setting, COLA always outputs 1 for whole duration of touch-sense time slice.</i></p>
FNCOL	[7:5]	rh	<p>Previous Active Function/LED Column Status</p> <p>Shows the active function / LED column in the previous time slice. Updated on start of new time-slice when LEDTS-counter 8LSB over-flows. Controlled by latched value of the internal DE-MUX, see Figure 12-4.</p>

LED and Touch-Sense (LEDTS)

Field	Bits	Type	Description
ACCNT¹⁾	[19:16]	rw	<p>Accumulate Count on Touch-Sense Input Defines the number of times a touch-sense input/pin is enabled in touch-sense time slice of consecutive frames. This provides to accumulate oscillation count on the TSIN[x].</p> <p>0_D 1 time 1_D 2 times ... 15_D 16 times</p>
TSCCMP	20	rw	<p>Common Compare Enable for Touch-Sense 0_B Disable common compare for touch-sense 1_B Enable common compare for touch-sense</p>
TSEOEXT	[22:21]	rw	<p>Extension for Touch-Sense Output for Pin-Low-Level 00_B Extend by 1 ledts_clk 01_B Extend by 4 ledts_clk 10_B Extend by 8 ledts_clk 11_B Extend by 16 ledts_clk</p>
TSCTRR	23	rw	<p>TS-Counter Auto Reset 0_B Disable TS-counter automatic reset 1_B Enable TS-counter automatic reset to 00H on the first pad turn of a new TSIN[x]. Triggered on compare match in time slice.</p>
TSCTRSAT	24	rw	<p>Saturation of TS-Counter 0_B Disable 1_B Enable. TS-counter stops counting in the touch-sense time slice(s) of the same (extended) frame when it reaches FFH. Counter starts to count again on the first pad turn of a new TSIN[x], triggered on compare match.</p>
NR_TSIN¹⁾	[27:25]	rw	<p>Number of Touch-Sense Input Defines the number of touch-sense inputs TSIN[x]. Used for the hardware control of pad turn enabling.</p> <p>0_D 1 ... 7_D 8</p>

LED and Touch-Sense (LEDTS)

Field	Bits	Type	Description
COLLEV	28	rw	Active Level of LED Column 0 _B Active low 1 _B Active high
NR_LEDCOL ¹⁾	[31:29]	rw	Number of LED Columns Defines the number of LED columns. 000 _B 1 LED column 001 _B 2 LED columns 010 _B 3 LED columns 011 _B 4 LED columns 100 _B 5 LED columns 101 _B 6 LED columns 110 _B 7 LED columns 111 _B 8 LED columns (if bit TS_EN = 1, a maximum number of 7 LED columns is allowed. Combination of TS_EN = 1 and NR_LEDCOL = 111 _B is disallowed) <i>Note: LED column is enabled in sequence starting from highest column number. If touch-sense function is not enabled, COLA is activated in last time slice.</i>
0	[15:8]	r	Reserved Read as 0; should be written with 0.

1) This bit can only be modified when bit CLK_PS = 0.

EVFR

The EVFR register contains the status flags for events and control bits for requesting clearance of event flags.

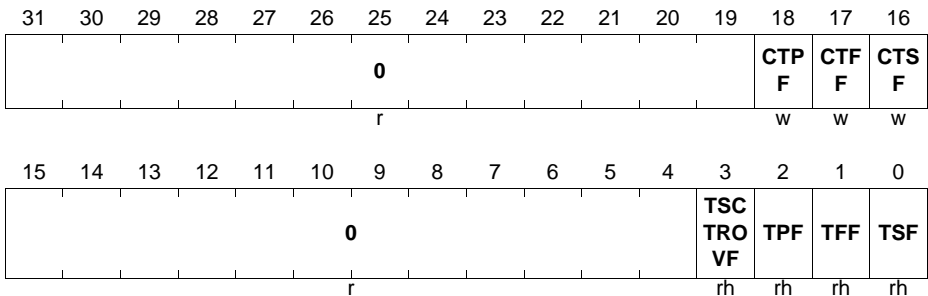
LED and Touch-Sense (LEDTS)

EVFR

Event Flag Register

(0C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TSF	0	rh	Time Slice Interrupt Flag Set on activation of each new time slice, including when bit CLK_PS is set from 0. To be cleared by software.
TFF	1	rh	(Extended) Time Frame Interrupt Flag Set on activation of each new (extended) time frame, including when bit CLK_PS is set from 0.
TPF	2	rh	Autoscan Time Period Interrupt Flag Set on activation of each new time period, including when bit CLK_PS is set from 0. This bit will never be set in case of hardware pad turn control is disabled (bit PADTSW = 1).
TSCTROVF	3	rh	TS-Counter Overflow Indication This bit indicates whether a TS-counter overflow has occurred. This bit is cleared on new pad turn, triggered on compare match. 0 _B No overflow has occurred. 1 _B The TS-counter has overflowed at least once.
CTSF	16	w	Clear Time Slice Interrupt Flag 0 _B No action. 1 _B Bit TSF is cleared. Read always as 0.

LED and Touch-Sense (LEDTS)

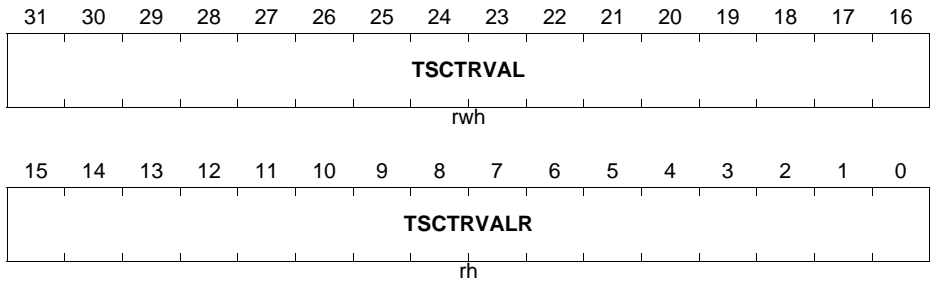
Field	Bits	Type	Description
CTFF	17	w	Clear (Extended) Time Frame Interrupt Flag 0 _B No action. 1 _B Bit TFF is cleared. Read always as 0.
CTPF	18	w	Clear Autoscan Time Period Interrupt Flag 0 _B No action. 1 _B Bit TPF is cleared. Read always as 0.
0	[31:19], [15:4]	r	Reserved Read as 0; should be written with 0.

TSVAL

The TSVAL register holds the current and shadow touch sense counter values.

TSVAL

Touch-sense TS-Counter Value (10_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TSCTRVALR	[15:0]	rh	Shadow TS-Counter Value (Read) This is the latched value of the TS-counter (on every extended time frame event). It shows the latest valid oscillation count from the last completed time slice.

LED and Touch-Sense (LEDTS)

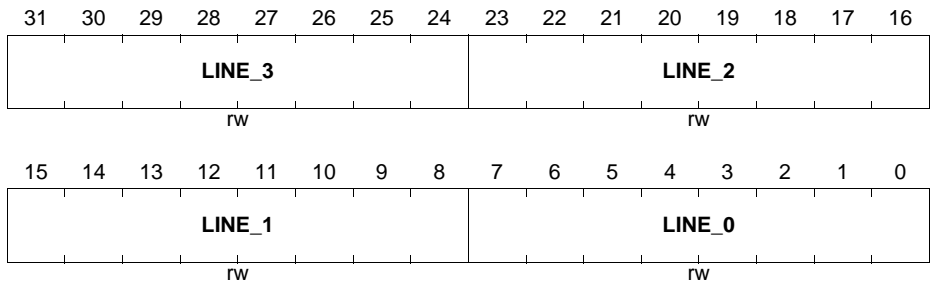
Field	Bits	Type	Description
TSCTRVAL	[31:16]	rw	TS-Counter Value This is the actual TS-counter value. It can only be written when no pad turn is active. The counter may be enabled for automatic reset once per (extended) frame on the start of a new pad turn on the next TSIN[x] pin.

LINE_x (x = 0-1)

The LINE_x registers hold the values that are output to the respective line pins during their active column period.

LINE0

Line Pattern Register 0 (14_H) **Reset Value: 0000 0000_H**

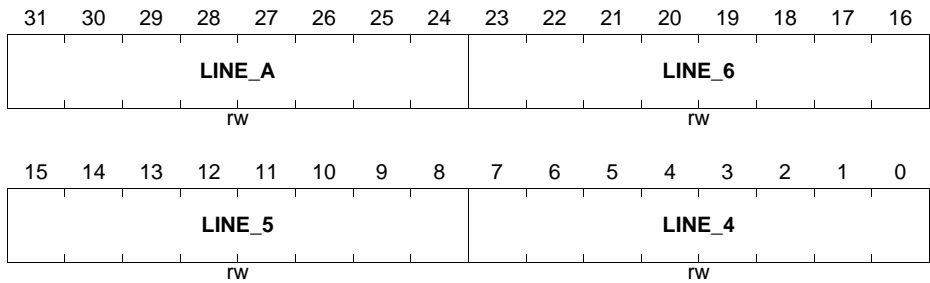


Field	Bits	Type	Description
LINE_0, LINE_1, LINE_2, LINE_3	[7:0], [15:8], [23:16], [31:24]	rw	Output on LINE[x] This value is output on LINE[x] to pin when LED COL[x] is active.

LED and Touch-Sense (LEDTS)

LINE1

Line Pattern Register 1 (18_H) **Reset Value: 0000 0000_H**



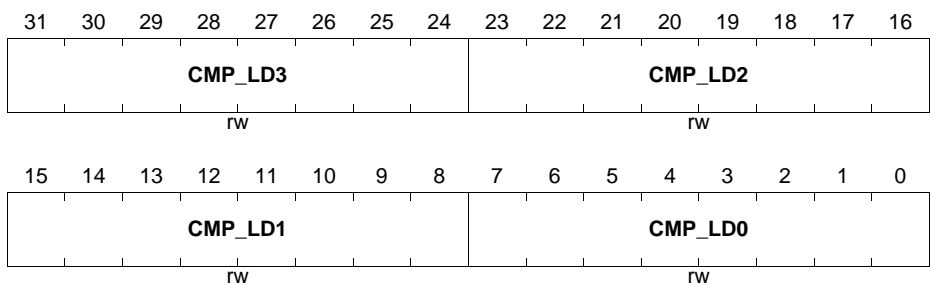
Field	Bits	Type	Description
LINE_4, LINE_5, LINE_6	[7:0], [15:8], [23:16]	rw	Output on LINE[x] This value is output on LINE[x] to pin when LED COL[x] is active.
LINE_A	[31:24]	rw	Output on LINE[x] This value is output on LINE[x] to pin when LED COLA or touch-sense time slice is active.

LDCMPx (x = 0-1)

The LDCMPx registers hold the COMPARE values for their respective LED columns. These values are used for LED brightness control.

LDCMP0

LED Compare Register 0 (1C_H) **Reset Value: 0000 0000_H**



LED and Touch-Sense (LEDTS)

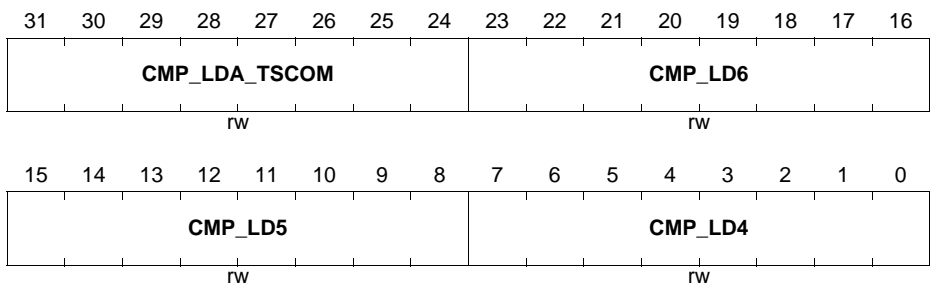
Field	Bits	Type	Description
CMP_LD0, CMP_LD1, CMP_LD2, CMP_LD3	[7:0], [15:8], [23:16], [31:24]	rw	Compare Value for LED COL[x]

LDCMP1

LED Compare Register 1

(20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CMP_LD4, CMP_LD5, CMP_LD6	[7:0], [15:8], [23:16]	rw	Compare Value for LED COL[x]
CMP_LDA_TS COM	[31:24]	rw	Compare Value for LED COLA / Common Compare Value for Touch-sense Pad Turns LED function The compare value for LED COLA is only valid when touch-sense function is not enabled. Touch-sense function The common compare value for touch-sense pad turns is enabled by set TSCCMP bit. When enabled for common compare, settings in SFRs LEDTS_TSCMP0,1 are not referenced.

TSCMPx (x = 0-1)

The TSCMPx registers hold the COMPARE values for their respective touch pad input lines. These values determine the size of the pad oscillation window for each pad input lines during their pad turn.

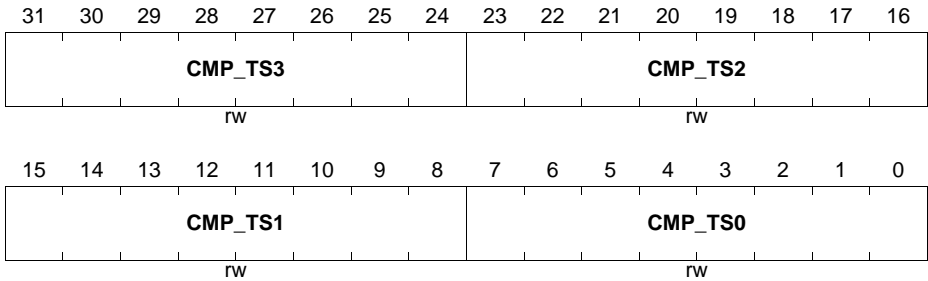
LED and Touch-Sense (LEDTS)

TSCMP0

Touch-sense Compare Register 0

(24_H)

Reset Value: 0000 0000_H



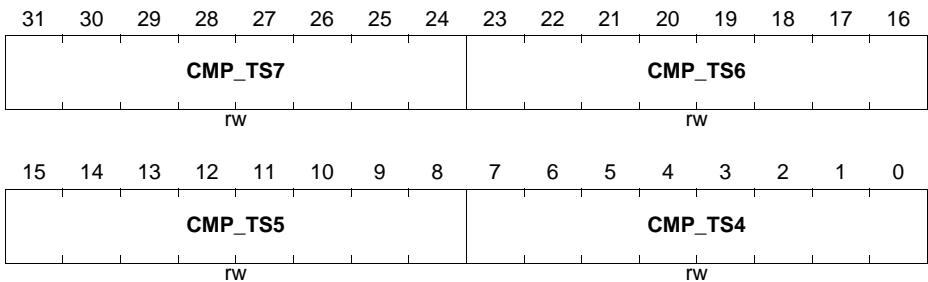
Field	Bits	Type	Description
CMP_TS0, CMP_TS1, CMP_TS2, CMP_TS3	[7:0], [15:8], [23:16], [31:24]	rw	Compare Value for Touch-Sense TSIN[x]

TSCMP1

Touch-sense Compare Register 1

(28_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CMP_TS4, CMP_TS5, CMP_TS6, CMP_TS7	[7:0], [15:8], [23:16], [31:24]	rw	Compare Value for Touch-Sense TSIN[x]

LED and Touch-Sense (LEDTS)

12.11 Interconnects

The LEDTS has interconnection to other peripherals enabling higher level of automation without requiring software. [Table 12-9](#) provides a list of the pin connections.

LEDTSx.FN is an output signal denoting LEDTS active function. This signal can be used as a source for VADC request gating and background gating. LEDTSx.SR0 is an output signal denoting LEDTS kernel service request. This signal can be used as an interrupt trigger .

Table 12-9 Pin Connection

Input/Output	I/O	Connected To	Description
LEDTS Kernel 0 Connectivity			
LEDTS0_clk	I	f_{PERIPH}	LEDTS0 kernel clock
SUSCFG	I	CPU Halted	Suspend signal
LEDTS0.TSIN0	I	P2.2	LEDTS Touch-sense 0 input
LEDTS0.TSIN1	I	P2.3	LEDTS Touch-sense 1 input
LEDTS0.TSIN2	I	P2.4	LEDTS Touch-sense 2 input
LEDTS0.TSIN3	I	P2.5	LEDTS Touch-sense 3 input
LEDTS0.TSIN4	I	P2.8	LEDTS Touch-sense 4 input
LEDTS0.TSIN5	I	P2.9	LEDTS Touch-sense 5 input
LEDTS0.TSIN6	I	P2.15	LEDTS Touch-sense 6 input
LEDTS0.TSIN7	I	not connected	LEDTS Touch-sense 7 input
LEDTS0.LINE0	O	P2.2	LEDTS Line 0 output
LEDTS0.LINE1	O	P2.3	LEDTS Line 1 output
LEDTS0.LINE2	O	P2.4	LEDTS Line 2 output
LEDTS0.LINE3	O	P2.5	LEDTS Line 3 output
LEDTS0.LINE4	O	P2.8	LEDTS Line 4 output
LEDTS0.LINE5	O	P2.9	LEDTS Line 5 output
LEDTS0.LINE6	O	P2.15	LEDTS Line 6 output
LEDTS0.LINE7	O	not connected	LEDTS Line 7 output
LEDTS0.COLA	O	P3.2 P5.7	LEDTS Column A output
LEDTS0.COL0	O	P0.9 P2.1	LEDTS Column 0 output

LED and Touch-Sense (LEDTS)

Table 12-9 Pin Connection (cont'd)

Input/Output	I/O	Connected To	Description
LEDTS0.COL1	O	P0.10 P2.0	LEDTS Column 1 output
LEDTS0.COL2	O	P0.0 P2.7	LEDTS Column 2 output
LEDTS0.COL3	O	P0.1 P2.6	LEDTS Column 3 output
LEDTS0.EXTENDED0	O	P2.2.HWO0	
LEDTS0.EXTENDED1	O	P2.3.HWO0	
LEDTS0.EXTENDED2	O	P2.4.HWO0	
LEDTS0.EXTENDED3	O	P2.5.HWO0	
LEDTS0.EXTENDED4	O	P2.8.HWO0	
LEDTS0.EXTENDED5	O	P2.9.HWO0	
LEDTS0.EXTENDED6	O	P2.15.HWO0	
LEDTS0.EXTENDED7	O	not connected	
LEDTS Kernel 0 Service Request Connectivity			
LEDTS0.FN	O	VADC.GxREQGTJ	1) VADC request gating
		VADC.BGREQGTJ	2) VADC background gating input J
LEDTS0.SR0	O	NVIC	Interrupt trigger



13 SD/MMC Interface (SDMMC)

This chapter describes the SD/MMC module. The XMC4300 uses the following SD and MMC card standard specification. For more detailed information on how to operate the SDMMC interface, please refer to the SD and MMC specification referenced below.

References

- [10] SD Specifications Part A2, SD Host Controller Standard Specification, Version 2.00, February 2007
https://www.sdcard.org/developers/overview/host_controller/simple_spec
- [11] SD Specifications Part 1, Physical Layer Specification, Version 2.00, May 2006
<https://www.sdcard.org/downloads/pls>
- [12] SD Specifications Part E1, SDIO Specification, Version 2.00, January 2007
https://www.sdcard.org/developers/overview/sdio/sdio_spec
- [13] SD Memory Card Security Specification, Version 1.01
- [14] The MultiMediaCard System Specification, Version 3.31, 4.2 and 4.4

13.1 Overview

The **Secure Digital/ MultiMediaCard** interface (SDMMC) of the XMC4300 provides an interface between SD/SDIO/MMC cards and the AHB bus. The CPU is programmed to support SD, SDIO, SDHC and MMC cards, and can operate up to 48 MHz. The SDMMC module is able to transfer a maximum of 24 MB/sec for SD cards and 48 MB/sec for MMC cards.

13.1.1 Features

The SDMMC Host Controller handles SDIO/SD protocol at transmission level, packing data, adding cyclic redundancy check (CRC), start/end bit, and checking for transaction format correctness. Some useful applications of the SDMMC includes memory extension, data logging, and firmware update.

The SDMMC is compliant with the following specifications:

- SD Card Host Controller Version 2.0
- SD Physical Layer Specification Version 2.0
- SDIO Card Specification Version 2.0
- SD Memory Card Security Specification Version 1.01
- MMC Specification version 3.31, 4.2 and 4.4
- Fully compatible with earlier versions of MMC

The following functionalities are supported by the SDMMC module:

- **System Interface**
 - Data transfer using Programmed IO mode on AHB Slave interface
- **SD/SDIO/MMC Card Interface**
 - Transfers data in 1 bit and 4 bit SD modes
 - Cyclic Redundancy Check CRC7 for command and CRC16 for data integrity
 - Variable-length data transfers for SD/SDIO cards
 - Designed to work with SD I/O cards, Read-only cards and Read/Write cards
 - Supports Read wait Control, and Suspend/Resume operation for SD/SDIO cards
 - Supports MMC Plus and MMC Mobile
 - MMC Card detection for insertion/removal
 - Error Correction Codes (ECC) for eMMC 4.4 cards
 - Password protection for MMC cards
 - Two 512 byte buffer for data transfers between core and cards
 - Handles FIFO overrun and underrun conditions

Table 13-1 SDMMC Applications

Use Case SDMMC	Application
Memory Extension	All
Data Logging	All
Firmware Update	All
Data Transfer (PC and Application)	All

13.1.2 Block Diagram

The SDMMC block diagram is shown in [Figure 13-1](#).

SD/MMC Interface (SDMMC)

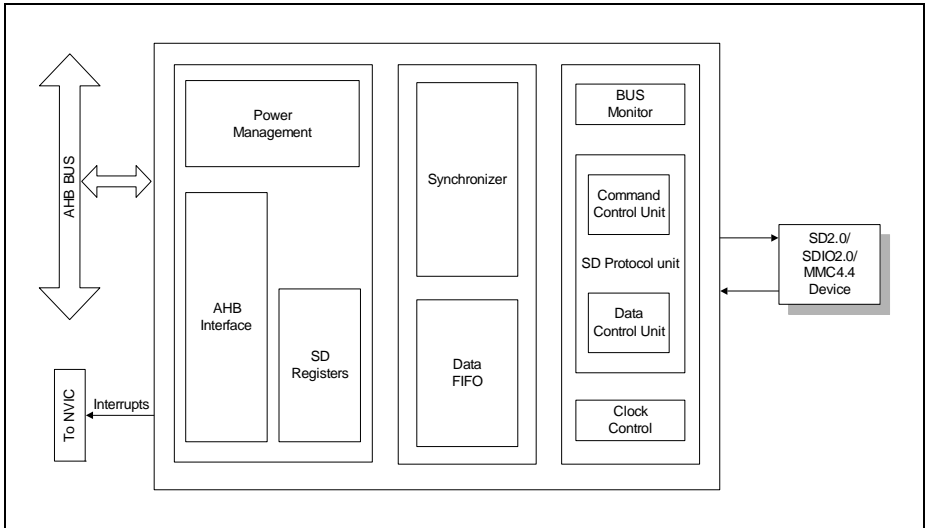


Figure 13-1 SDMMC Block Diagram

13.2 Functional Description

This section describes the functional blocks of the SDMMC.

AHB Interface

Host AHB interface acts as a bridge between AHB and the host controller. The SDMMC host controller provides Programmed IO method in which the ARM Host Driver transfers data using the Buffer Data Port Register **SDMMC_DATA_BUFFER**. The AHB target is having the Host control register **SDMMC_HOST_CTRL** and these registers are programmed by the CPU through the AHB target interface. The data transaction is performed through the AHB target interface in case of Programmed IO method of data transfer.

Interrupt controller

The SDMMC host controller generates interrupt to the Nested Vectored Interrupt Controller (NVIC) if any of the interrupt bits are set in the interrupt status register **SDMMC_INT_STATUS_NORM**.

DATA FIFO

The SDMMC host controller uses two 512 bytes dual port fifo for performing both read and write transactions. During a write transaction (data transferred from CPU to SD/SDIO/MMC card), the data will be filled in to the first and second fifo alternatively. When data from first fifo is transferring to the SD/SDIO/MMC card, the second fifo will be filled and vice versa. The two fifo's are alternatively used to store data which will give maximum throughput. During a read transaction (data transferred from SD/SDIO/MMC card to CPU), the data from SD/SDIO/MMC card will be written in to the two fifo's alternatively. When data from one fifo is transferring to the CPU, the second fifo will be filled and vice versa and thereby the throughput will be maximum. If the host controller cannot accept any data from SD/SDIO/MMC card, it will issue read wait (if card supports read wait mechanism) to stop the data coming from card or through stopping clock.

DAT[0-7] Control Logic

The DAT[0-7] control logic block transmits data in the data lines during write transaction and receives data in the data lines during read transaction.

BUS Monitor

Bus monitor will check for any violations occurring in the SD bus and time-out conditions.

Command Control Logic

The Command control logic block sends the command in the cmd line and receives the response coming from the SD/SDIO/MMC card.

Power Control

The SDMMC host controller controls the SD Bus Power depending on the value programmed in the Power Control Register **SDMMC_POWER_CTRL** by the CPU. The system has the responsibility to supply SD Bus Voltage according to card OCR and supply voltage capabilities depending on the host controller. If SD Bus power **SDMMC_POWER_CTRL.SD_BUS_POWER** = 1, the system shall supply voltage to the Card. If an unsupported voltage is selected in the SD Bus Voltage Select **SDMMC_POWER_CTRL.SD_BUS_VOLTAGE_SEL** field, the system may ignore write to SD Bus Power and keep its value at 0.

Clock Control

The Clock generation block will generate the SD clock depending on the value programmed by the CPU in the Clock Control Register **SDMMC_CLOCK_CTRL**.

Stream write and read transaction

SDMMC host controller will switch to second fifo after writing/reading a block of data to the first fifo, but in stream transaction, block size will not be programmed by the driver.

For both stream write and read transaction, it is recommended to write the maximum fifo size value to Block Size Register **SDMMC_BLOCK_SIZE**. For example if fifo size is 512 bytes, host driver needs to write 512 bytes to the **SDMMC_BLOCK_SIZE**. Fifo switching will occur after writing/ reading 512 bytes of data (fifo size).

13.3 Card Detection

When card insertion or removal in the slot is detected, the status will be sent to the CPU via interrupt methodology. The active low card signal `SDCD_n` is set to 0 during card detection. The `SDMMC_PRESENT_STATE.CARD_INSERTED` bit indicates whether a card has been inserted. A change from 0 to 1 generates a Card Insertion interrupt in the Normal Interrupt Status register `SDMMC_INT_STATUS_NORM.CARD_INS = 1`, and a change from 1 to 0 generates a Card Removal Interrupt in the Normal Interrupt Status register `SDMMC_INT_STATUS_NORM.CARD_REMOVAL = 1`.

Note: LED light indicates that card is being accessed. Do not remove card when LED light is ON.

13.4 Data Transfer Modes

SDMMC transfers are classified into following three modes, according to how the number of blocks is specified:

Single Block Transfer

Single block transfer mode can be selected by setting `SDMMC_TRANSFER_MODE.MULTI_BLOCK_SELECT = 0`. The number of blocks is specified to the host controller before the transfer via Block Count Register, and it is always set to 1. `SDMMC_BLOCK_COUNT.BLOCK_COUNT = 0001H`.

Multiple Block Transfer

Multiple block transfer mode can be selected by setting `SDMMC_TRANSFER_MODE.MULTI_BLOCK_SELECT = 1`. The number of blocks is specified to the host controller before the transfer via Block Count Register `SDMMC_BLOCK_COUNT.BLOCK_COUNT`, and it can be set to 1 or more.

Infinite Block Transfer

The number of blocks is not specified to the host controller before the transfer. This transfer is continued until an abort transaction is executed. Refer to [Section 13.5.3](#) for details on Abort Transaction.

13.5 Read/ Write Operation

The controller will be configured to work with buffer data port registers **SDMMC_DATA_BUFFER** without internal DMA. The CPU will act as a master and start writing / reading data via **SDMMC_DATA_BUFFER**.

13.5.1 Write Operation

On receiving the Buffer Write Ready interrupt the CPU will act as a master and start transferring the data via Buffer data port register **SDMMC_DATA_BUFFER** (fifo_1). Transmitter starts sending the data in SD bus when a block of data is ready in fifo_1. While transmitting the data in sd bus, the buffer write ready interrupt is sent to the interrupt controller for the second block of data. The CPU will act as a master and start sending the second block of data via **SDMMC_DATA_BUFFER** to fifo_2. Buffer write ready interrupt will be asserted only when a fifo is empty to receive a block of data.

During write transaction the host controller will transmit data to card only when a block of data is ready to transmit and also the card is not driving busy. So underrun condition will not occur in SD side. The controller will assert buffer write ready interrupt **SDMMC_INT_STATUS_NORM.BUFF_WRITE_READY = 1** only if space is available to accept a block of data.

13.5.2 Read Operation

Buffer Read Ready interrupt is asserted whenever a block of data is ready in one of the fifo's. On receiving the Buffer Read Ready interrupt, the CPU will act as a master and start reading the data via Buffer data port register **SDMMC_DATA_BUFFER** (fifo_1). Receiver start reading the data from SD bus only when a fifo is empty to receive a block of data. When both the fifo's are full the host controller will stop the data coming from the card through read wait mechanism (if card supports read wait) or through clock stopping. During read transaction when fifo is full, that is, no space to accept one block of data from card, the host controller will stop the clock to card so overrun condition will not occur in SD side. The controller will assert buffer read ready interrupt **SDMMC_INT_STATUS_NORM.BUFF_READ_READY = 1** only on reception of block of data from card.

13.5.3 Abort Transaction

There are two cases where the host driver needs to perform an Abort Transaction:

1. When the host driver stops infinite block transfers.
2. When host driver stops transfers while a multiple block transfer is exacting.

There are two ways to issue an Abort command; Asynchronous Abort and Synchronous Abort.

Asynchronous Abort

In an Asynchronous Abort sequence, the host driver can issue an Abort Command at anytime unless Command Inhibit (CMD) in the Present State Register is set to 1. **SDMMC_PRESENT_STATE.COMMAND_INHIBIT_CMD = 1.**

Synchronous Abort

In a Synchronous Abort, the host driver shall issue an Abort command after the data transfer stopped by using Stop At Block Gap Request in the Block Gap Control register. **SDMMC_BLOCK_GAP_CTRL.STOP_AT_BLOCK_GAP = 1.**

13.6 Special Command Types

There are three types of special commands. Suspend, Resume and Abort. These bits shall be set to 00_B for all other commands.

Suspend Command

Suspend command can be selected by setting **SDMMC_COMMAND.CMD_TYPE** = 01_B.

If the Suspend command succeeds, the host controller shall assume the SD Bus has been released and that it is possible to issue the next command which uses the DAT line. The controller shall de-assert Read Wait for read transactions and stop checking busy for write transactions. The Interrupt cycle shall start, in 4-bit mode. If the Suspend command fails, the controller shall maintain its current state, and the host driver shall restart the transfer by setting Continue Request in the Block Gap Control Register **SDMMC_BLOCK_GAP_CTRL.CONTINUE_REQ** = 1 to restart the transfer.

*Note: Suspend / Resume cannot be supported if Read Wait Control is disabled. Set **SDMMC_BLOCK_GAP_CTRL.READ_WAIT_CTRL** = 1 to enable Read Wait Control if the SD/SDIO card supports read wait function.*

Resume Command

Resume command can be selected by setting **SDMMC_COMMAND.CMD_TYPE** = 10_B.

The host driver re-starts the data transfer by restoring the registers in the range of 000_H - 00D_H. The host controller shall check for busy before starting write transfers.

*Note: Suspend / Resume cannot be supported if Read Wait Control is disabled. Set **SDMMC_BLOCK_GAP_CTRL.READ_WAIT_CTRL** = 1 to enable Read Wait Control if the SD/SDIO card supports the read wait function.*

Abort Command

Abort command can be selected by setting **SDMMC_COMMAND.CMD_TYPE** = 11_B.

If this command is set when executing a read transfer, the host controller shall stop reads to the buffer. If this command is set when executing a write transfer, the host controller shall stop driving the DAT line. After issuing the Abort command, the controller should issue a software reset

13.7 Error Detection

This section describes data errors or defects detection methods.

Cyclic Redundancy Check (CRC)

The CRC7 and CRC16 generators calculate the CRC for Command and Data respectively to send the CRC to the SD/SDIO/MMC card. The CRC7 and CRC16 checker checks for any CRC error in the Response and Data sent by the SD/SDIO/MMC card. When a CRC error is generated, an interrupt will be triggered if the Error Interrupt Signal Enable is enabled. [SDMMC_EN_INT_SIGNAL_ERR.DATA_CRC_ERR_EN = 1](#) for data CRC error, and [SDMMC_EN_INT_STATUS_ERR.COMD_CRC_ERR_EN = 1](#) for command CRC error.

Error Correction Code (ECC)

Error correction codes (ECC) may be included in the payload data to detect data defects on the cards. An ECC code is used to store data on the MMC card. This ECC code is used by the SDMMC or application to decode the user data.

13.8 Service Request Generation

The SDMMC module provides one service request output. The service request output MMCI.SR0 is connected to interrupt node in the Nested Vectored Interrupt Controller (NVIC). For details on the service request and interrupt node, please refer to “Service Request Processing” and “Nested Vectored Interrupt Controller” chapters in the reference manual.

13.9 Debug Behavior

The SDMMC module is not affected when the CPU enters HALT mode.

13.10 Power, Reset and Clocks

This section describes the behaviour of power, reset and clocks.

Power

The SD/MMC card power supply can be controlled by the signal bus_pow. The SD bus voltage supported by the SDMMC is 3.3V. If the SD Bus power is set to 1 in the Power Control Register [SDMMC_POWER_CTRL.SD_BUS_POWER = 1](#), the system shall supply voltage to the card.

Reset

The SDMMC host controller is reset asynchronously when one of the following occurs:

SD/MMC Interface (SDMMC)

- A hardware reset to the card triggered by the `MMC.RST` pin.
- A software reset occurs. A reset pulse is generated when writing 1 to each bit of the Software Reset Register `SDMMC_SW_RESET`.

Clocks

The clocks connected to SDMMC include:

- `clk_xin`
 - Input clock to the SDMMC controller.
 - This is used to generate `clk_sdcard_out` and `clk_sleep_out`.
 - Frequency of clock is 48 MHz, generated from the System Control Unit (SCU) module.
- `clk_sdcard_out`
 - Clock supplied to the SD/MMC card.
- `clk_sdcard_in`
 - Feedback clock of `clk_sdcard_out` from the pad.
 - Feedback clock is used to reduce the pad delay in the clock line.

13.10.1 Tap delay control

Tap delay is used for conditions where the hold time requirements could not be met due to different routing delays between the SD clock and the SD CMD/data line.

The delay chain will have 16 delay elements with each delay being 200ps. This provides the option of a maximum delay of 3.2ns.

Tap delay can be enabled by setting `SDMMCDEL.TAPEN = 1`. The delay time can be controlled by selecting the number of delay elements and writing the value to `SDMMCDEL.TAPDEL`. For more details, please refer to GCU Registers in the System Control Unit (SCU) chapter.

13.11 Initialisation and System Dependencies

This section provides information on how to initialise and use the SDMMC.

13.11.1 Setup SDMMC Data Transfer

Figure 13-2 shows the flowchart of SDMMC read/ write data transfer.

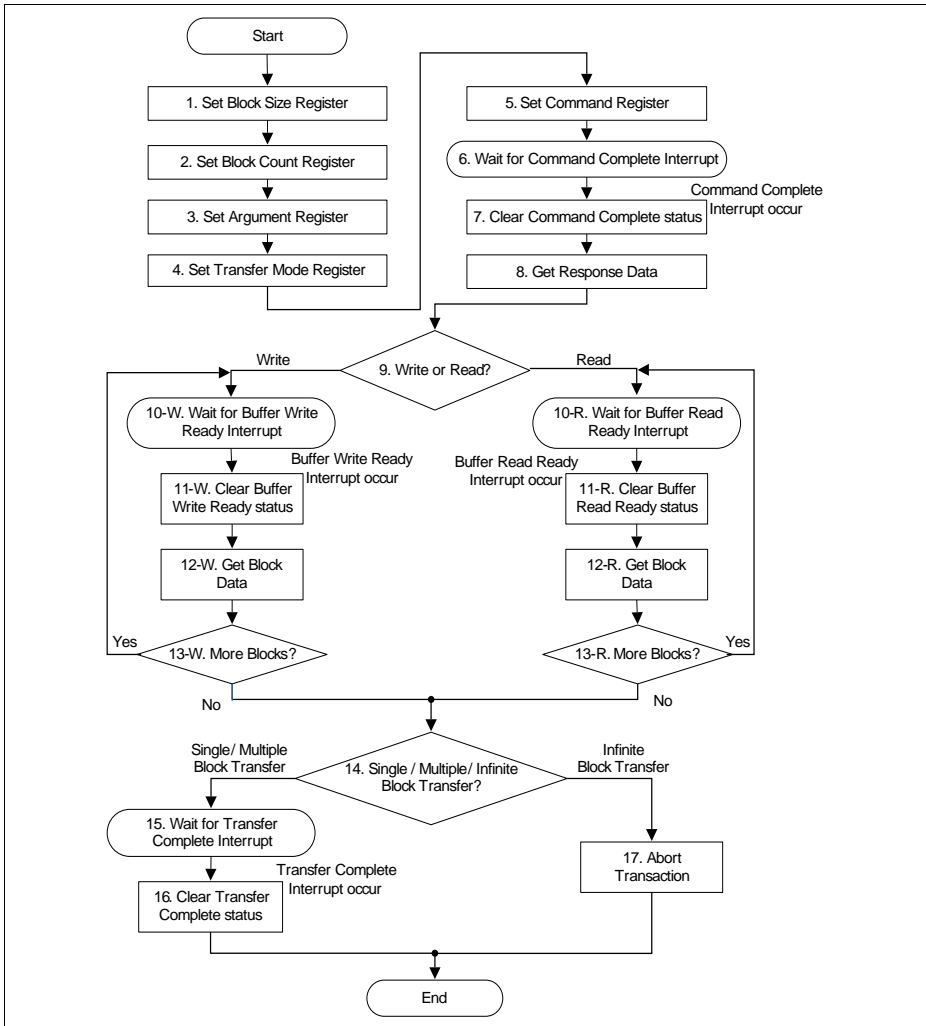


Figure 13-2 Data Transfer sequence

The following describes how to setup read/ write data transfer:

1. Set Block Size Register. Set executed data byte length of one block.

`SDMMC_BLOCK_SIZE.TX_BLOCK_SIZE`

2. Set Block Count Register. Set executed data block count.

`SDMMC_BLOCK_COUNT.BLOCK_COUNT`

3. Set Argument Register. Set value corresponding to issued command.

`SDMMC_ARGUMENT1.ARGUMENT1`

4. Set Transfer Mode Register. Set Multi / single block, block count enable, data transfer direction, Auto CMD12 enable.

`SDMMC_TRANSFER_MODE.MULTI_BLOCK_SELECT.`

`SDMMC_TRANSFER_MODE.BLOCK_COUNT_EN`

`SDMMC_TRANSFER_MODE.TX_DIR_SELECT`

`SDMMC_TRANSFER_MODE.ACMD_EN`

5. Set Command Register. Set value corresponding to the issued command.

Note: When writing the upper byte of Command register, SD command is issued.

SDMMC_COMMAND

6. Wait for Command Complete Interrupt.

`SDMMC_INT_STATUS_NORM.CMD_COMPLETE`

7. Clear Command Complete status

Write `SDMMC_INT_STATUS_NORM.CMD_COMPLETE = 1` to clear bit

8. Read Response Register and get the necessary information in accordance with the issued command.

`SDMMC_RESPONSE`

9. For Read Operation (read from card), go to step (10-R). See **Section 13.11.2**.

For Write Operation (write to card), go to step (10-W). See **Section 13.11.3**.

13.11.2 Read Operation

The following shows the configurations for SDMMC read operation:

10-R. Wait for Buffer Read Ready Interrupt

[SDMMC_INT_STATUS_NORM](#).BUFF_READ_READY

11-R. Clear Buffer Read Ready status

Write [SDMMC_INT_STATUS_NORM](#).BUFF_READ_READY = 1 to clear bit

12-R. Read Block Data (in accordance with the number of bytes specified in step (1))

[SDMMC_DATA_BUFFER](#)

13-R. Repeat until all blocks are received and then go to step (14)

14-R. For Single or Multiple Block Transfer, go to step (15). For Infinite Block Transfer, go to step (17)

15-R. Wait for Transfer Complete Interrupt

[SDMMC_INT_STATUS_NORM](#).TX_COMPLETE

16-R. Get Transfer Complete status and end data transfer

Write [SDMMC_INT_STATUS_NORM](#).TX_COMPLETE = 1 to clear bit

17-R. Perform Abort Transaction. See [Section 13.11.4](#).

13.11.3 Write Operation

The following shows the configurations for SDMMC write operation:

10-W. Wait for Buffer Write Ready Interrupt

[SDMMC_INT_STATUS_NORM](#).BUFF_WRITE_READY

11-W. Clear Buffer Write Ready status

Write [SDMMC_INT_STATUS_NORM](#).BUFF_WRITE_READY = 1 to clear bit

12-W. Write Block Data (in accordance with the number of bytes specified in step (1))

[SDMMC_DATA_BUFFER](#)

13-W. Repeat until all blocks are received and then go to step (14)

14-W. For Single or Multiple Block Transfer, go to step (15). For Infinite Block Transfer, go to step (17)

15-W. Wait for Transfer Complete Interrupt

[SDMMC_INT_STATUS_NORM](#).TX_COMPLETE

16-W. Get Transfer Complete status and end data transfer

Write [SDMMC_INT_STATUS_NORM](#).TX_COMPLETE = 1 to clear bit

17-W. Perform Abort Transaction. See [Section 13.11.4](#).

13.11.4 Abort Transaction

This section describes the sequence for the abort transaction.

Asynchronous Abort

The following shows the asynchronous abort sequence:

1. Check **SDMMC_PRESENT_STATE**. **COMMAND_INHIBIT_CMD** is not set to 1.
2. Issue Abort Command. **SDMMC_COMMAND**. **CMD_TYPE** = 11_B.

Synchronous Abort

The following shows the synchronous abort sequence:

1. Set **SDMMC_BLOCK_GAP_CTRL**. **STOP_AT_BLOCK_GAP** = 1 to stop SD transactions.
2. Wait for Transfer Complete Interrupt. **SDMMC_INT_STATUS_NORM**. **TX_COMPLETE**.
3. Set **SDMMC_INT_STATUS_NORM**. **TX_COMPLETE** = 1 to clear this bit.
4. Issue Abort Command. **SDMMC_COMMAND**. **CMD_TYPE** = 11_B.
5. Set **SDMMC_SW_RESET**. **SW_RST_DAT_LINE** = 1 and **SDMMC_SW_RESET**. **SW_RST_CMD_LINE** = 1 to do software reset.
6. Check **SW_RST_DAT_LINE** and **SW_RST_CMD_LINE**. If both are 0, end data transfer. If either **SW_RST_DAT_LINE** or **SW_RST_CMD_LINE** is 1, repeat step (6).

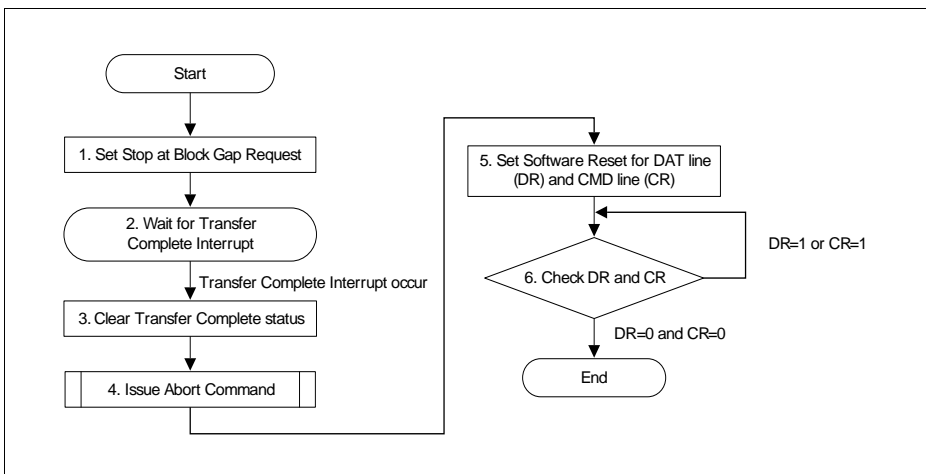


Figure 13-3 Synchronous Abort sequence

13.11.5 Write Protection and Card Detection Software Control

The SD card write protect and card detection pins can be used for other output functions in case the write protection and card detection features are not used.

Write Protection

By default, the input from the SD card write protect pin (P1.1) indicates whether the card is write protected or write enabled. The SDMMC write protection input can be switched to software control via the **SDMMC_CON** register in the System Control Unit (SCU).

The following shows the configurations for SDMMC write protection software control:

1. Set SDMMC write protection input multiplexer control to software bit WPSVAL. **SDMMC_CON.WPSEL** = 1.
2. Set **SDMMC_CON.WPSVAL** = 0 for no write protection, or **SDMMC_CON.WPSVAL** = 1 for write protection active.

Card Detection

By default, the input from the card detection pin (P1.10) indicates whether a card is present or not present. The SDMMC card detection input can be switched to software control via the **SDMMC_CON** register in the System Control Unit (SCU).

The following shows the configurations for SDMMC card detection software control:

1. Set SDMMC card detection control to software bit CDSVAL. **SDMMC_CON.CDSEL** = 1.
2. Set **SDMMC_CON.CDSVAL** = 0 for no card detected, or **SDMMC_CON.CDSVAL** = 1 for card detected.

13.12 Registers

Registers Overview

The absolute register address is calculated by adding:
Module Base Address + Offset Address

Table 13-2 Registers Address Space

Module	Base Address	End Address	Note
SDMMC	4801 C000 _H	4801 FFFF _H	

Table 13-3 Register Overview

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
Reserved	Reserved	0000 _H - 0002 _H	nBE	nBE	

Block Registers

SDMMC_BLOC K_SIZE	Block Size Register	0004 _H	U, PV	U, PV	Page 13-2 1
SDMMC_BLOC K_COUNT	Block Count Register	0006 _H	U, PV	U, PV	Page 13-2 2

Argument1 Register

SDMMC_ARGU MENT1	Argument1 Register	0008 _H	U, PV	U, PV	Page 13-2 3
---------------------	--------------------	-------------------	-------	-------	---------------------------------

Transfer Mode Register

SDMMC_TRAN SFER_MODE	Transfer Mode Register	000C _H	U, PV	U, PV	Page 13-2 4
-------------------------	------------------------	-------------------	-------	-------	---------------------------------

Command Register

SDMMC_COMM AND	Command Register	000E _H	U, PV	U, PV	Page 13-2 7
-------------------	------------------	-------------------	-------	-------	---------------------------------

Response Register

SDMMC_RESP ONSE0	Response 0 Register	0010 _H	U, PV	U, PV	Page 13-3 0
SDMMC_RESP ONSE2	Response 2 Register	0014 _H	U, PV	U, PV	Page 13-3 0

Table 13-3 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
SDMMC_RESP ONSE4	Response 4 Register	0018 _H	U, PV	U, PV	Page 13-30
SDMMC_RESP ONSE6	Response 6 Register	001C _H	U, PV	U, PV	Page 13-30
Buffer Data Port Register					
SDMMC_DATA_ BUFFER	Data Buffer Register	0020 _H	U, PV	U, PV	Page 13-34
Present State Register					
SDMMC_PRES ENT_STATE	Present State Register	0024 _H	U, PV	U, PV	Page 13-35
Control Registers					
SDMMC_HOST _CTRL	Host Control Register	0028 _H	U, PV	U, PV	Page 13-44
SDMMC_POWE R_CTRL	Power Control Register	0029 _H	U, PV	U, PV	Page 13-46
SDMMC_BLOC K_GAP_CTRL	Block Gap Control Register	002A _H	U, PV	U, PV	Page 13-47
SDMMC_WAKE UP_CTRL	Wake-up Control Register	002B _H	U, PV	U, PV	Page 13-50
SDMMC_CLOC K_CTRL	Clock Control Register	002C _H	U, PV	U, PV	Page 13-51
SDMMC_TIMEO UT_CTRL	Timeout Control Register	002E _H	U, PV	U, PV	Page 13-54
SDMMC_SW_R ESET	Software Reset Register	002F _H	U, PV	U, PV	Page 13-55
Interrupt Status Registers					
SDMMC_INT_S TATUS_NORM	Normal Interrupt Status Register	0030 _H	U, PV	U, PV	Page 13-57
SDMMC_INT_S TATUS_ERR	Error Interrupt Status Register	0032 _H	U, PV	U, PV	Page 13-63
SDMMC_EN_IN T_STATUS_NO RM	Normal Interrupt Status Enable Register	0034 _H	U, PV	U, PV	Page 13-68

Table 13-3 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
SDMMC_EN_INTERRUPT_STATUS_ERROR	Error Interrupt Status Enable Register	0036 _H	U, PV	U, PV	Page 13-70
SDMMC_EN_INTERRUPT_SIGNAL_NORMAL	Normal Interrupt Signal Enable Register	0038 _H	U, PV	U, PV	Page 13-72
SDMMC_EN_INTERRUPT_SIGNAL_ERROR	Error Interrupt Signal Enable Register	003A _H	U, PV	U, PV	Page 13-74
SDMMC_AUTO_CMD12_ERROR_STATUS	Auto CMD12 Error Status Register	003C _H	U, PV	U, PV	Page 13-76
Reserved	Reserved	003E _H	nBE	nBE	
Capabilities Registers					
SDMMC_CAPABILITIES	Capabilities Register	0040 _H	U, PV	U, PV	Page 13-79
SDMMC_CAPABILITIES_HI	Capabilities Register High	0044 _H	U, PV	U, PV	Page 13-81
SDMMC_MAXIMUM_CURRENT_CAP	Maximum Current Capabilities Register	0048 _H	U, PV	U, PV	Page 13-83
Reserved	Reserved	004C _H - 004E _H	nBE	nBE	
Error Status Registers					
SDMMC_FORCE_EVENT_AUTO_CMD_ERROR_STATUS	Force Event Register for Auto CMD Error Status	0050 _H	U, PV	U, PV	Page 13-84
SDMMC_FORCE_EVENT_ERROR_INTERRUPT_STATUS	Force Event Register for Error Interrupt Status	0052 _H	U, PV	U, PV	Page 13-86
Reserved	Reserved	0054 _H - 0072 _H	nBE	nBE	
Debug Selection Register					
SDMMC_DEBUG_SELECTION	Debug Selection Register	0074 _H	U, PV	U, PV	Page 13-88

Table 13-3 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
Reserved	Reserved	0075 _H - 00FA _H	nBE	nBE	

Slot Interrupt Status Register

SDMMC_SLOT_INT_STATUS	Slot Interrupt Status Register	00FC _H	U, PV	U, PV	Page 13-89
Reserved	Reserved	00FE _H	nBE	nBE	

GCU Registers (in System Control Unit)

SDMMCDL	SDMMC Delay Control Register	Refer to System Control Unit (SCU) chapter
SDMMC_CON	SDMMC Configuration Register	

Access Restrictions

Note: The SDMMC registers are accessible only through word accesses. Half-word and byte accesses on SDMMC registers will not generate a bus error. Writes to unused address space will not cause an error but will be ignored.

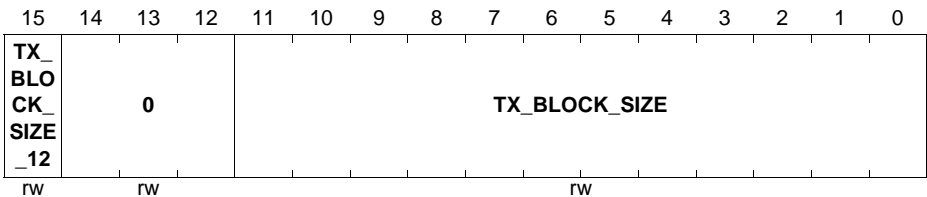
13.12.1 Registers Description

SDMMC_BLOCK_SIZE

This register is used to configure the block size for data transfer.

SDMMC_BLOCK_SIZE

Block Size Register (0004_H) **Reset Value: 0000_H**



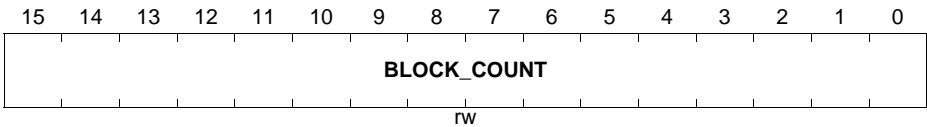
Field	Bits	Type	Description
TX_BLOCK_SIZE	[11:0]	rw	<p>Transfer Block Size</p> <p>This register specifies the block size for block data transfers for CMD17, CMD18, CMD24, CMD25, and CMD53. It can be accessed only if no transaction is executing (i.e after a transaction has stopped). Read operations during transfer return an invalid value and write operations shall be ignored.</p> <p>0000_H No Data Transfer 0001_H 1 Byte 0002_H 2 Bytes 0003_H 3 Bytes 0004_H 4 Bytes ... 01FF_H 511 Bytes 0200_H 512 Bytes (Maximum Block Size)</p> <p><i>Note: Other values are reserved</i></p> <p>This bit can be set and cleared only by software.</p>
0	[14:12]	rw	<p>Reserved</p> <p>Read as 0; must be written with 0.</p>
TX_BLOCK_SIZE_12	15	rw	<p>Transfer Block Size 12th bit.</p> <p>This bit is added to support 4Kb Data block transfer. This bit can be set and cleared only by software.</p>

SDMMC_BLOCK_COUNT

This register is used to configure the block count for current transfer.

SDMMC_BLOCK_COUNT

Block Count Register (0006_H) **Reset Value: 0000_H**



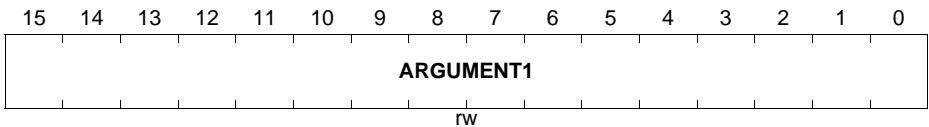
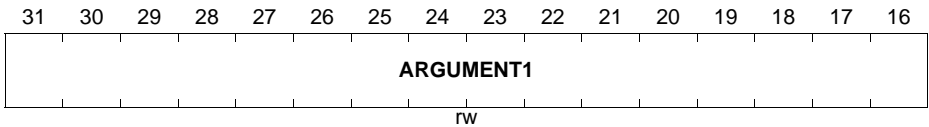
Field	Bits	Type	Description
BLOCK_COUNT	[15:0]	rw	<p>Blocks Count for Current Transfer</p> <p>This register is enabled when Block Count Enable in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. The host controller decrements the block count after each block transfer and stops when the count reaches zero. It can be accessed only if no transaction is executing (i.e after a transaction has stopped). Read operations during transfer return an invalid value and write operations shall be ignored.</p> <p>When saving transfer context as a result of Suspend command, the number of blocks yet to be transferred can be determined by reading this register. When restoring transfer context prior to issuing a Resume command, the host driver shall restore the previously save block count.</p> <p>0000_H Stop Count 0001_H 1 block 0002_H 2 blocks ... FFFF_H 65535 blocks</p> <p>This bit can be set and cleared only by software.</p>

SDMMC_ARGUMENT1

This register is used to configure the SD command argument.

SDMMC_ARGUMENT1

Argument1 Register (0008_H) Reset Value: 00000000_H



Field	Bits	Type	Description
ARGUMENT1	[31:0]	rw	Command Argument The SD Command Argument is specified as bit 39-8 of Command-Format. This bit can be set and cleared only by software.

SDMMC_TRANSFER_MODE

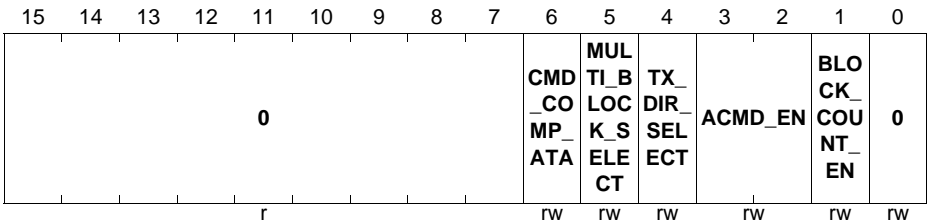
This register is used to configure the data transfer mode.

SDMMC_TRANSFER_MODE

Transfer Mode Register

(000C_H)

Reset Value: 0000_H



Field	Bits	Type	Description
0	0	rw	Reserved Read as 0; must be written with 0.
BLOCK_COUNT_EN	1	rw	Block Count Enable This bit is used to enable the Block count register, which is only relevant for multiple block transfers. When this bit is 0, the Block Count register is disabled, which is useful in executing an infinite transfer. 0 _B Disable 1 _B Enable This bit can be set and cleared only by software.

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
ACMD_EN	[3:2]	rw	<p>Auto CMD Enable</p> <p>This field determines use of auto command functions</p> <p>00_B Auto Command Disabled</p> <p>01_B Auto CMD12 Enable</p> <p><i>Note: Other values are reserved</i></p> <p>To stop Multiple-block read and write operation:</p> <ul style="list-style-type: none"> Auto CMD12 Enable <p>Multiple-block read and write commands for memory require CMD12 to stop the operation. When this field is set to 01_B, the host controller issues CMD12 automatically when last block transfer is completed. Auto CMD12 error is indicated to the Auto CMD Error Status register. The Host Driver shall not set this bit if the command does not require CMD12.</p> <p>This bit can be set and cleared only by software.</p>
TX_DIR_SELECT	4	rw	<p>Data Transfer Direction Select</p> <p>This bit defines the direction of DAT line data transfers.</p> <p>0_B Write (Host to Card)</p> <p>1_B Read (Card to Host)</p> <p>This bit can be set and cleared only by software.</p>
MULTI_BLOCK_SELECT	5	rw	<p>Multi / Single Block Select</p> <p>This bit enables multiple block DAT line data transfers.</p> <p>0_B Single Block</p> <p>1_B Multiple Block</p> <p>This bit can be set and cleared only by software.</p>
CMD_COMP_AT A	6	rw	<p>Command Completion Signal Enable for CE-ATA Device</p> <p>1_B Device will send command completion Signal</p> <p>0_B Device will not send command completion Signal</p> <p>This bit can be set and cleared only by software.</p>
0	[15:7]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Determination of transfer type

Table 13-4 Determination of transfer type

Multi / Single Block Select	Block Count Enable	Block Count	Function
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Not Zero	Multiple Transfer
1	1	Zero	Stop Multiple Transfer

SDMMC_COMMAND

This register is used to configure the SDMMC command.

SDMMC_COMMAND

Command Register

(000E_H)

Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0		CMD_IND						CMD_TY P E		DAT A_P RES ENT _SE LEC T	CMD _IND _CH ECK _EN	CMD _CR C_C HEC K_E N	0		RESP_TY PE_SELE CT	
r		rw						rw		rw	rw	rw	r		rw	

Field	Bits	Type	Description
RESP_TYPE_SELECT	[1:0]	rw	Response Type Select 00 _B No Response 01 _B Response length 136 10 _B Response length 48 11 _B Response length 48 check Busy after response This bit can be set and cleared only by software.
0	2	r	Reserved Read as 0; should be written with 0.
CMD_CRC_CHECK_EN	3	rw	Command CRC Check Enable If this bit is set to 1, the host controller shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. 0 _B Disable 1 _B Enable This bit can be set and cleared only by software.

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
CMD_IND_CHECK_EN	4	rw	<p>Command Index Check Enable</p> <p>If this bit is set to 1, the host controller shall check the index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked.</p> <p>0_B Disable 1_B Enable</p> <p>This bit can be set and cleared only by software.</p>
DATA_PRESENT_SELECT	5	rw	<p>Data Present Select</p> <p>This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line. If is set to 0 for the following:</p> <ol style="list-style-type: none"> 1. Commands using only CMD line (ex. CMD52) 2. Commands with no data transfer but using busy signal on DAT[0] line (R1b or R5b ex. CMD38) 3. Resume Command <p>0_B No Data Present 1_B Data Present</p> <p>This bit can be set and cleared only by software.</p>
CMD_TYPE	[7:6]	rw	<p>Command Type</p> <p>There are three types of special commands. Suspend, Resume and Abort. These bits shall bet set to 00b for all other commands.</p> <p>00_B Normal 01_B Suspend 10_B Resume 11_B Abort</p> <p>This bit can be set and cleared only by software.</p>
CMD_IND	[13:8]	rw	<p>Command Index</p> <p>This bit shall be set to the command number (CMD0-63, ACMD0-63).</p> <p>This bit can be set and cleared only by software.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
0	[15:14]	r	Reserved Read as 0; should be written with 0.

SDMMC_RESPONSE

This register is used to configure the command response. [Table 13-5](#) shows the relation between parameters and the name of response type.

Table 13-5 Relation between parameters and the name of response type

Response Type	Index Check Enable	CRC Check Enable	Name of Response Type
00	0	0	No Response
01	0	1	R2
10	0	0	R3, R4
10	1	1	R1, R6, R5, R7
11	1	1	R1b, R5b

[Table 13-6](#) describes the mapping of command responses from the SD Bus to this register for each response type. In the table, R[] refers to a bit range within the response data as transmitted on the SD Bus, RESPONSE[] refers to a bit range within the Response register.

Table 13-6 Response bit definition for each response type

Kind of Response	Meaning of Response	Response Field	Response Register
R1, R1b (normal response)	Card Status	R[39:8]	RESPONSE 0[31:0]
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R[39:8]	RESPONSE 6[31:0]
R2 (CID, CSD Register)	CID or CSD reg. incl.	R[127:8]	RESPONSE 6[23:0], RESPONSE 4[31:0] RESPONSE 2[31:0], RESPONSE 0[31:0]
R3 (OCR Register)	OCR Register for memory	R[39:8]	RESPONSE 0[31:0]
R4 (OCR Register)	OCR Register for I/O etc.	R[39:8]	RESPONSE 0[31:0]

SD/MMC Interface (SDMMC)

Table 13-6 Response bit definition for each response type (cont'd)

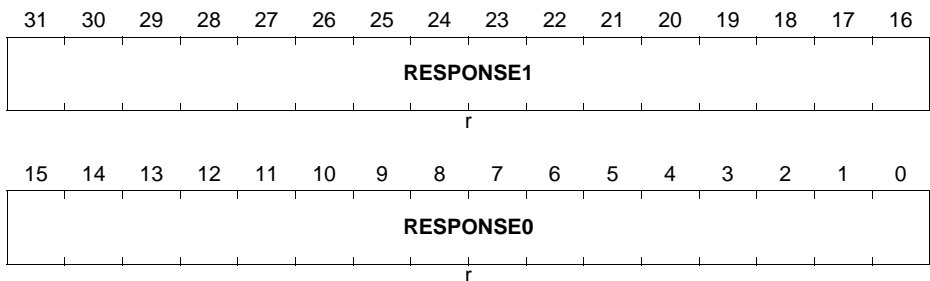
Kind of Response	Meaning of Response	Response Field	Response Register
R5, R5b	SDIO Response	R[39:8]	RESPONSE 0[31:0]
R6 (Published RCA response)	New published RCA[31:16] etc.	R[39:8]	RESPONSE 0[31:0]

SDMMC_RESPONSE0

Response 0 Register

(0010_H)

Reset Value: 00000000_H



Field	Bits	Type	Description
RESPONSE0	[15:0]	r	Response0 This bit is initialized to 0 at reset.
RESPONSE1	[31:16]	r	Response1 This bit is initialized to 0 at reset.

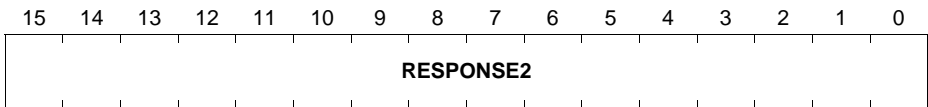
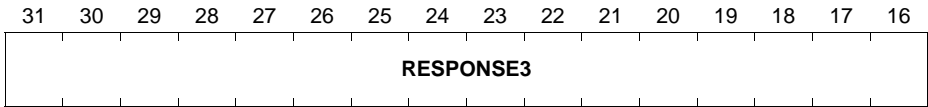
SD/MMC Interface (SDMMC)

SDMMC_RESPONSE2

Response 2 Register

(0014_H)

Reset Value: 00000000_H



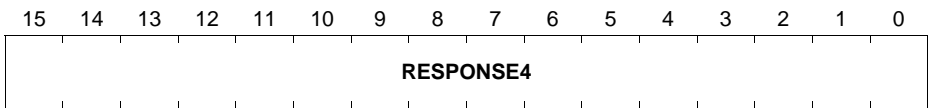
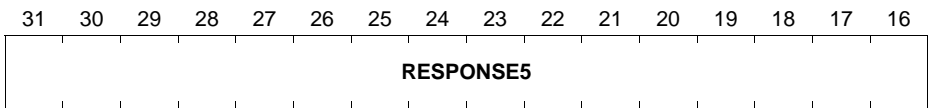
Field	Bits	Type	Description
RESPONSE2	[15:0]	r	Response2 This bit is initialized to 0 at reset.
RESPONSE3	[31:16]	r	Response3 This bit is initialized to 0 at reset.

SDMMC_RESPONSE4

Response 4 Register

(0018_H)

Reset Value: 00000000_H



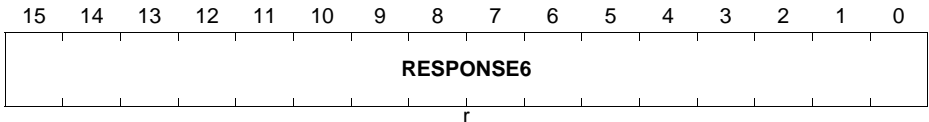
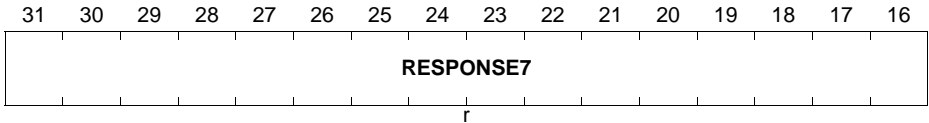
Field	Bits	Type	Description
RESPONSE4	[15:0]	r	Response4 This bit is initialized to 0 at reset.
RESPONSE5	[31:16]	r	Response5 This bit is initialized to 0 at reset.

SDMMC_RESPONSE6

Response 6 Register

(001C_H)

Reset Value: 00000000_H



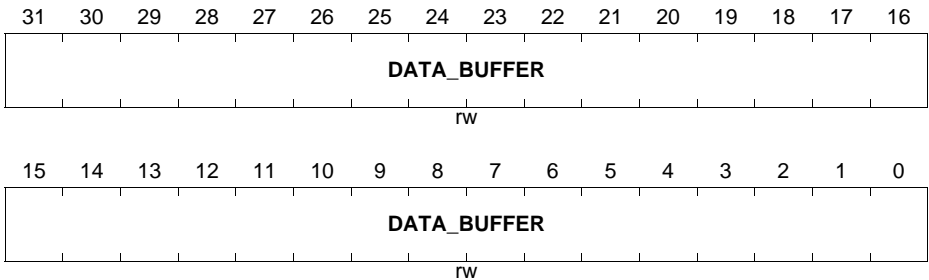
Field	Bits	Type	Description
RESPONSE6	[15:0]	r	Response6 This bit is initialized to 0 at reset.
RESPONSE7	[31:16]	r	Response7 This bit is initialized to 0 at reset.

SDMMC_DATA_BUFFER

This register is used to configure the SDMMC host controller data buffer.

SDMMC_DATA_BUFFER

Data Buffer Register (0020_H) Reset Value: 00000000_H



Field	Bits	Type	Description
DATA_BUFFER	[31:0]	rw	<p>Data Buffer The host controller buffer can be accessed through this 32-bit Data Port Register. Reset: X This bit can be set and cleared only by software.</p>

SD/MMC Interface (SDMMC)

SDMMC_PRESENT_STATE

This register is used to check the present state of the SDMMC host controller.

SDMMC_PRESENT_STATE

Present State Register

(0024_H)

Reset Value: 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			DAT_7_4_PIN_LEVEL				CMD_LINE_LEVEL	DAT_3_0_PIN_LEVEL				WRITE_PROTECT_PIN_LEVEL	CARD_DETECT_PIN_LEVEL	CARD_STATUS_TABLE	CARD_INSERTED
r			r				r	r				r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				BUFFER_ADENABLE	BUFFER_WRITEENABLE	READ_TRAN_SFER_ACTIV E	WRITE_TRAN_SFER_ACTIV E	0				DAT_LINE_ACTIVE	COMMAND_INHIBIT_D AT	COMMAND_INHIBIT_C MD	
r				r	r	r	r	r				r	r	r	

Field	Bits	Type	Description
COMMAND_INHIBIT_CMD	0	r	<p>Command Inhibit (CMD)</p> <p>If this bit is 0, it indicates the CMD line is not in use and the host controller can issue a SD command using the CMD line. This bit is set immediately after the Command register (00Fh) is written. This bit is cleared when the command response is received. Even if the Command Inhibit (DAT) is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a Command complete interrupt in the Normal Interrupt Status register. If the host controller cannot issue the command because of a command conflict error or because of Command Not Issued By Auto CMD12 Error, this bit shall remain 1 and the Command Complete is not set. Status issuing Auto CMD12 is not read from this bit.</p> <p>Auto CMD12 consists of two responses. In this case, this bit is not cleared by the response of CMD12 but cleared by the response of a read/write command. Status issuing Auto CMD12 is not read from this bit. So if a command is issued during Auto CMD12 operation, host controller shall manage to issue two commands: CMD12 and a command set by Command register.</p> <p>This bit is initialized to 0 at reset.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
COMMAND_INHIBIT_DAT	1	r	<p>Command Inhibit (DAT)</p> <p>This status bit is generated if either the DAT Line Active or the Read transfer Active is set to 1. If this bit is 0, it indicates the host controller can issue the next SD command. Commands with busy signal belong to Command Inhibit (DAT) (ex. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Normal interrupt status register.</p> <p><i>Note: The SD Host Driver can save registers in the range of 000_H - 00D_H for a suspend transaction after this bit has changed from 1 to 0.</i></p> <p>0_B Can issue command which uses the DAT line 1_B Cannot issue command which uses the DAT line</p> <p>This bit is initialized to 0 at reset.</p>
DAT_LINE_ACTIVE	2	r	<p>DAT Line Active</p> <p>This bit indicates whether one of the DAT line on SD bus is in use ¹⁾.</p> <p>0_B DAT line inactive 1_B DAT line active</p> <p>This bit is initialized to 0 at reset.</p>
0	[7:3]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
WRITE_TRANSFER_ACTIVE	8	r	<p>Write Transfer Active</p> <p>This status indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the host controller. This bit is set in either of the following cases:</p> <ul style="list-style-type: none"> After the end bit of the write command. When writing a 1 to Continue Request in the Block Gap Control register to restart a write transfer. <p>This bit is cleared in either of the following cases:</p> <ul style="list-style-type: none"> After getting the CRC status of the last data block as specified by the transfer count (Single or Multiple) After getting a CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request. <p>During a write transaction, a Block Gap Event interrupt is generated when this bit is changed to 0, as a result of the Stop At Block Gap Request being set. This status is useful for the host driver in determining when to issue commands during write busy.</p> <p>0_B No valid data 1_B Transferring data</p> <p>This bit is initialized to 0 at reset.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
READ_TRANSFER_ACTIVE	9	r	<p>Read Transfer Active</p> <p>This status is used for detecting completion of a read transfer.</p> <p>This bit is set to 1 for either of the following conditions:</p> <p>After the end bit of the read command</p> <p>When writing a 1 to continue Request in the Block Gap Control register to restart a read transfer</p> <p>This bit is cleared to 0 for either of the following conditions:</p> <p>When the last data block as specified by block length is transferred to the system.</p> <p>When all valid data blocks have been transferred to the system and no current block transfers are being sent as a result of the Stop At Block Gap Request set to 1. A transfer complete interrupt is generated when this bit changes to 0.</p> <p>0_B No valid data 1_B Transferring data</p> <p>This bit is initialized to 0 at reset.</p>
BUFFER_WRITE_ENABLE	10	r	<p>Buffer Write Enable</p> <p>This status is used for non-DMA write transfers. This read only flag indicates if space is available for write data. If this bit is 1, data can be written to the buffer. A change of this bit from 1 to 0 occurs when all the block data is written to the buffer. A change of this bit from 0 to 1 occurs when top of block data can be written to the buffer and generates the Buffer Write Ready Interrupt.</p> <p>0_B Write Disable 1_B Write Enable.</p> <p>This bit is initialized to 0 at reset.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
BUFFER_READ_EN ABLE	11	r	<p>Buffer Read Enable</p> <p>This status is used for non-DMA read transfers. This read only flag indicates that valid data exists in the host side buffer status. If this bit is 1, readable data exists in the buffer. A change of this bit from 1 to 0 occurs when all the block data is read from the buffer. A change of this bit from 0 to 1 occurs when all the block data is ready in the buffer and generates the Buffer Read Ready Interrupt.</p> <p>0_B Read Disable 1_B Read Enable. This bit is initialized to 0 at reset.</p>
0	[15:12]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>
CARD_INSERTED	16	r	<p>Card Inserted</p> <p>This bit indicates whether a card has been inserted. Changing from 0 to 1 generates a Card Insertion interrupt in the Normal Interrupt Status register and changing from 1 to 0 generates a Card Removal Interrupt in the Normal Interrupt Status register. The Software Reset For All in the Software Reset register shall not affect this bit.</p> <p>If a Card is removed while its power is on and its clock is oscillating, the host controller shall clear SD Bus Power in the Power Control register and SD Clock Enable in the Clock control register. In addition the host driver should clear the host controller by the Software Reset For All in Software register. The card detect is active regardless of the SD Bus Power.</p> <p>0_B Reset or Debouncing or No Card 1_B Card Inserted</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
CARD_STATE_STABLE	17	r	<p>Card State Stable</p> <p>This bit is used for testing. If it is 0, the Card Detect Pin Level is not stable. If this bit is set to 1, it means the Card Detect Pin Level is stable. The Software Reset For All in the Software Reset Register shall not affect this bit.</p> <p>0_B Reset of Debouncing 1_B No Card or Inserted Reset: 1_B</p>
CARD_DETECT_PIN_LEVEL	18	r	<p>Card Detect Pin Level</p> <p>This bit reflects the inverse value of the SDCD module input from SCU.</p> <p>0_B No Card present (SDCD = 1) 1_B Card present (SDCD = 0)</p> <p><i>Note: Card detect settings via SDMMC_CON bit may be considered If card detection is switched to software control.</i></p>
WRITE_PROTECT_PIN_LEVEL	19	r	<p>Write Protect Switch Pin Level</p> <p>The Write Protect Switch is supported for memory and combo cards. This bit reflects the SDWP module input from SCU.</p> <p>0_B Write protected (SDWP = 1) 1_B Write enabled (SDWP = 0)</p> <p><i>Note: Write protect settings via SDMMC_CON bit may be considered If write protection is switched to software control.</i></p>
DAT_3_0_PIN_LEVEL	[23:20]	r	<p>Line Signal Level</p> <p>This status is used to check DAT line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from DAT[0].</p> <p>D23 - DAT[3] D22 - DAT[2] D21 - DAT[1] D20 - DAT[0] Reset: F_H</p>

Field	Bits	Type	Description
CMD_LINE_LEVEL	24	r	CMD Line Signal Level This status is used to check CMD line level to recover from errors, and for debugging. Reset: 1 _B
DAT_7_4_PIN_LEVEL	[28:25]	r	Line Signal Level This status is used to check DAT line level to recover from errors, and for debugging. D28 - DAT[7] D27 - DAT[6] D26 - DAT[5] D25 - DAT[4] Reset: F _H
0	[31:29]	r	Reserved Read as 0; should be written with 0.

1) *DAT line active indicates whether one of the DAT line is on SD bus is in use.*

(a) In the case of read transactions

This status indicates whether a read transfer is executing on the SD Bus. Changing this value from 1 to 0 generates a Block Gap Event interrupt in the Normal Interrupt Status register, as the result of the Stop At Block Gap Request being set.

This bit shall be set in either of the following cases:

1. After the end bit of the read command.
2. When writing a 1 to Continue Request in the Block Gap Control register to restart a read transfer.

This bit shall be cleared in either of the following cases:

1. When the end bit of the last data block is sent from the SD Bus to the host controller.
2. When a read transfer is stopped at the block gap initiated by a Stop At BlockGap Request.

The host controller shall stop read operation at the start of the interrupt cycle of the next block gap by driving Read Wait or stopping SD clock. If the Read Wait signal is already driven (due to data buffer cannot receive data), the host controller can continue to stop read operation by driving the Read Wait signal. It is necessary to support Read Wait in order to use suspend / resume function.

(b) In the case of write transactions

This status indicates that a write transfer is executing on the SD Bus. Changing this value from 1 to 0 generate a Transfer Complete interrupt in the Normal Interrupt Status register.

This bit shall be set in either of the following cases:

1. After the end bit of the write command.
2. When writing to 1 to Continue Request in the Block Gap Control register to continue a write transfer.

This bit shall be cleared in either of the following cases:

1. When the SD card releases write busy of the last data block. If SD card does not drive busy signal for 8 SD Clocks, the host controller shall consider the card drive "Not Busy".
2. When the SD card releases write busy prior to waiting for write transfer as a result of a Stop At Block Gap Request.

(c) Command with busy

This status indicates whether a command indicates busy (ex. erase command for memory) is executing on the SD Bus. This bit is set after the end bit of the command with busy and cleared when busy is de-asserted. Changing this bit from 1 to 0 generate a Transfer Complete interrupt in the Normal Interrupt Status register.

Note: The host driver can issue cmd0, cmd12, cmd13 (for memory) and cmd52 (for SDIO) when the DAT lines are busy during data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands shall be issued when Command Inhibit (DAT) is set to zero.

SDMMC_HOST_CTRL

This register is used to configure the modes of the SDMMC host controller.

SDMMC_HOST_CTRL

Host Control Register

(0028_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
CARD_DE T_SIGNAL _DETECT	CARD_DE TECT_TES T_LEVEL	SD_8BIT_ MODE	0		HIGH_SPE ED_EN	DATA_TX _WIDTH	LED_CTR L
rw	rw	rw	rw		rw	rw	rw

Field	Bits	Type	Description
LED_CTRL	0	rw	<p>LED Control</p> <p>This bit is used to caution the user not to remove the card while the SD card is being accessed. If the software is going to issue multiple SD commands, this bit can be set during all transactions. It is not necessary to change for each transaction.</p> <p>0_B LED off 1_B LED on</p> <p>This bit can be set and cleared only by software.</p>
DATA_TX_WIDTH	1	rw	<p>Data Transfer Width (SD1 or SD4)</p> <p>This bit selects the data width of the host controller. The host driver shall select it to match the data width of the SD card.</p> <p>0_B 1 bit mode 1_B 4-bit mode</p> <p>This bit can be set and cleared only by software.</p>
HIGH_SPEED_EN	2	rw	<p>High Speed Enable</p> <p>This bit is optional. If this bit is set to 0 (default), the host controller outputs CMD line and DAT lines at the falling edge of the SD clock (up to 25 MHz / 20 MHz for MMC). If this bit is set to 1, the host controller outputs CMD line and DAT lines at the rising edge of the SD clock (up to 50 MHz for SD / 52 MHz for MMC)</p> <p>0_B Normal Speed Mode 1_B High Speed Mode</p> <p>This bit can be set and cleared only by software.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
0	[4:3]	rw	Reserved Read as 0; must be written with 0.
SD_8BIT_MODE	5	rw	Extended Data Transfer Width This bit controls 8-bit bus width mode for embedded device. If a device supports 8-bit bus mode, this bit may be set to 1. If this bit is 0, bus width is controlled by Data Transfer Width DATA_TX_WIDTH. 0 _B Bus Width is selected by Data Transfer Width 1 _B 8-bit Bus Width This bit can be set and cleared only by software.
CARD_DETECT_TEST_LEVEL	6	rw	Card Detect Test Level This bit is enabled while the Card Detect Signal Selection is set to 1 and it indicates card inserted or not. Generates (card ins or card removal) Interrupt when the normal int sts enable bit is set. 0 _B No Card 1 _B Card Inserted This bit can be set and cleared only by software.
CARD_DET_SIGNAL_DETECT	7	rw	Card detect signal detection This bit selects source for card detection. 0 _B SDCD is selected (for normal use) 1 _B The card detect test level is selected This bit can be set and cleared only by software.

SD/MMC Interface (SDMMC)

SDMMC_POWER_CTRL

This register is used to configure the SD bus power.

SDMMC_POWER_CTRL

Power Control Register

(0029_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
0			HARDWA RE_RESE T	SD_BUS_VOLTAGE_SEL			SD_BUS POWER
r			rw	rw			rw

Field	Bits	Type	Description
SD_BUS_POWER	0	rw	<p>SD Bus Power</p> <p>Before setting this bit, the SD host driver shall set SD Bus Voltage Select. If the host controller detects the No Card State, this bit shall be cleared.</p> <p>0_B Power off 1_B Power on</p> <p>This bit can be set and cleared only by software.</p>
SD_BUS_VOLTAGE_SEL	[3:1]	rw	<p>SD Bus Voltage Select</p> <p>By setting these bits, the host driver selects the voltage level for the SD card. If an unsupported voltage is selected, the Host System shall not supply SD bus voltage</p> <p>111_B 3.3V (Flatop.)</p> <p><i>Note: Other values are reserved</i></p> <p>This bit can be set and cleared only by software.</p>
HARDWARE_RESET	4	rw	<p>Hardware reset</p> <p>Hardware reset signal is generated for eMMC4.4 card when this bit is set</p> <p>This bit can be set and cleared only by software.</p>
0	[7:5]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

SDMMC_BLOCK_GAP_CTRL

This register is used to configure the block gap request.

SDMMC_BLOCK_GAP_CTRL

Block Gap Control Register

(002A_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
0		0		INT_AT_BLOCK_GAP	READ_WAIT_CTRL	CONTINUE_REQ	STOP_AT_BLOCK_GAP
r		rw		rw	rw	rw	rw

Field	Bits	Type	Description
STOP_AT_BLOCK_GAP	0	rw	<p>Stop At Block Gap Request</p> <p>This bit is used to stop executing a transaction at the next block gap for non- DMA transfers. Until the transfer complete is set to 1, indicating a transfer completion the host driver shall leave this bit set to 1. Clearing both the Stop At Block Gap Request and Continue Request shall not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The host controller shall honour Stop At Block Gap Request for write transfers, but for read transfers it requires that the SD card support Read Wait. Therefore the host driver shall not set this bit during read transfers unless the SD card supports Read Wait and has set Read Wait Control to 1. In case of write transfers in which the host driver writes data to the Buffer Data Port register, the host driver shall set this bit after all block data is written. If this bit is set to 1, the host driver shall not write data to Buffer data port register. This bit affects Read Transfer Active, Write Transfer Active, DAT line active and Command Inhibit (DAT) in the Present State register.</p> <p>0_B Transfer 1_B Stop</p> <p>This bit can be set and cleared only by software.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
CONTINUE_REQ	1	rw	<p>Continue Request</p> <p>This bit is used to restart a transaction which was stopped using the Stop At Block Gap Request. To cancel stop at the block gap, set Stop At block Gap Request to 0 and set this bit to restart the transfer.</p> <p>The host controller automatically clears this bit in either of the following cases:</p> <ol style="list-style-type: none"> 1. In the case of a read transaction, the DAT Line Active changes from 0 to 1 as a read transaction restarts. 2. In the case of a write transaction, the Write transfer active changes from 0 to 1 as the write transaction restarts. <p>Therefore it is not necessary for Host driver to set this bit to 0. If Stop At Block Gap Request is set to 1, any write to this bit is ignored.</p> <p>0_B Ignored 1_B Restart</p>
READ_WAIT_CTRL	2	rw	<p>Read Wait Control</p> <p>The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using DAT[2] line. Otherwise the host controller has to stop the SD clock to hold read data, which restricts commands generation. When the host driver detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card. If the card does not support read wait, this bit shall never be set to 1 otherwise DAT line conflict may occur. If this bit is set to 0, Suspend / Resume cannot be supported</p> <p>0_B Disable Read Wait Control 1_B Enable Read Wait Control</p> <p>This bit can be set and cleared only by software.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
INT_AT_BLOCK_GAP	3	rw	Interrupt At Block Gap This bit is valid only in 4-bit mode of the SDIO card and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. If the SD card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0. When the host driver detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card. This bit can be set and cleared only by software.
0	[6:4]	rw	Reserved Read as 0; must be written with 0.
0	7	r	Reserved Read as 0; should be written with 0.

There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether the host controller issues a Suspend command or the SD card accepts the Suspend command.

1. If the host driver does not issue Suspend command, the Continue Request shall be used to restart the transfer.
2. If the host driver issues a Suspend command and the SD card accepts it, a Resume Command shall be used to restart the transfer.
3. If the host driver issues a Suspend command and the SD card does not accept it, the Continue Request shall be used to restart the transfer.

Any time Stop At Block Gap Request stops the data transfer, the host driver shall wait for Transfer Complete (in the Normal Interrupt Status register) before attempting to restart the transfer. When restarting the data transfer by Continue Request, the host driver shall clear Stop At Block Gap Request before or simultaneously.

SD/MMC Interface (SDMMC)

SDMMC_WAKEUP_CTRL

Wakeup functionality depends on the host controller system hardware and software. The host driver shall maintain voltage on the SD Bus, by setting SD Bus power to 1 in the Power Control register, when wakeup event via card interrupt is desired.

SDMMC_WAKEUP_CTRL

Wake-up Control Register

(002B_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
0					WAKEUP_ EVENT_E N_REM	WAKEUP_ EVENT_E N_INS	WAKEUP_ EVENT_E N_INT
					rw	rw	rw

Field	Bits	Type	Description
WAKEUP_EVENT_EN_INT	0	rw	<p>Wakeup Event Enable On Card Interrupt</p> <p>This bit enables wakeup event via Card Interrupt assertion in the Normal Interrupt Status register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1.</p> <p>0_B Disable 1_B Enable</p> <p>This bit can be set and cleared only by software.</p>
WAKEUP_EVENT_EN_INS	1	rw	<p>Wakeup Event Enable On SD Card Insertion</p> <p>This bit enables wakeup event via Card Insertion assertion in the Normal Interrupt Status register. FN_WUS (Wake up Support) in CIS does not affect this bit.</p> <p>0_B Disable 1_B Enable</p> <p>This bit can be set and cleared only by software.</p>
WAKEUP_EVENT_EN_REM	2	rw	<p>Wakeup Event Enable On SD Card Removal</p> <p>This bit enables wakeup event via Card Removal assertion in the Normal Interrupt Status register. FN_WUS (Wake up Support) in CIS does not affect this bit.</p> <p>0_B Disable 1_B Enable</p> <p>This bit can be set and cleared only by software.</p>
0	[7:3]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

SDMMC_CLOCK_CTRL

This register is used to configure the SD Clock.

SDMMC_CLOCK_CTRL

Clock Control Register

(002C_H)

Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDCLK_FREQ_SEL										0	0	SDC LOC K_EN	INTE RNA L_C LOC K_S TAB LE	INTE RNA L_C LOC K_EN	
rw										rw	r	rw	r	rw	

Field	Bits	Type	Description
INTERNAL_CLOCK_EN	0	rw	<p>Internal Clock Enable</p> <p>This bit is set to 0 when the host driver is not using the host controller or the host controller awaits a wakeup event. The host controller should stop its internal clock to go very low power state. Still, registers shall be able to be read and written. Clock starts to oscillate when this bit is set to 1. When clock oscillation is stable, the host controller shall set Internal Clock Stable in this register to 1. This bit shall not affect card detection.</p> <p>0_B Stop 1_B Oscillate</p> <p>This bit can be set and cleared only by software.</p>
INTERNAL_CLOCK_STABLE	1	r	<p>Internal Clock Stable</p> <p>This bit is set to 1 when SD clock is stable after writing to Internal Clock Enable in this register to 1. The SD Host Driver shall wait to set SD Clock Enable until this bit is set to 1.</p> <p><i>Note: This is useful when using PLL for a clock oscillator that requires setup time.</i></p> <p>0_B Not Ready 1_B Ready</p> <p>This bit is initialized to 0 at reset.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
SDCLOCK_EN	2	rw	<p>SD Clock Enable</p> <p>The host controller shall stop SDCLK when writing this bit to 0. SDCLK frequency Select can be changed when this bit is 0. Then, the host controller shall maintain the same clock frequency until SDCLK is stopped (Stop at SDCLK = 0). If the host controller detects the No Card state, this bit shall be cleared.</p> <p>0_B Disable 1_B Enable</p> <p>This bit can be set and cleared only by software.</p>
0	[5:3]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>
0	[7:6]	rw	<p>Reserved</p> <p>Read as 0; must be written with 0.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
SDCLK_FREQ_SEL	[15:8]	rw	<p>SDCLK Frequency Select</p> <p>This register is used to select the frequency of the SDCLK pin. The frequency is not programmed directly; this register holds the divisor of the Base Clock Frequency for SD clock. Only the following settings are allowed.</p> <p>8-bit Divided Clock Mode</p> <p>00_H base clock(10MHz-63MHz) 01_H base clock divided by 2 10_H base clock divided by 32 02_H base clock divided by 4 04_H base clock divided by 8 08_H base clock divided by 16 20_H base clock divided by 64 40_H base clock divided by 128 80_H base clock divided by 256</p> <p>Setting 00_H specifies the highest frequency of the SD Clock. When setting multiple bits, the most significant bit is used as the divisor. But multiple bits should not be set. The two default divider values can be calculated by the Base Clock Frequency for SD Clock (48MHz).</p> <ol style="list-style-type: none"> 25 MHz divider value 400 kHz divider value <p>The frequency of the SDCLK is set by the following formula: Clock Frequency = (Base clock) / divisor. Thus choose the smallest possible divisor which results in a clock frequency that is less than or equal to the target frequency. Maximum Frequency for SD = 48 MHz (base clock) Maximum Frequency for MMC = 48 MHz (base clock) Minimum Frequency = 187.5 kHz (48 MHz / 256), same calculation for MMC</p> <p>This bit can be set and cleared only by software.</p>

SDMMC_TIMEOUT_CTRL

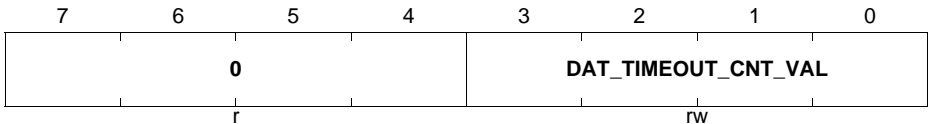
This register is used to configure the interval for data timeout.

SDMMC_TIMEOUT_CTRL

Timeout Control Register

(002E_H)

Reset Value: 00_H



Field	Bits	Type	Description
DAT_TIMEOUT_CNT_VAL	[3:0]	rw	<p>Data Timeout Counter Value</p> <p>This value determines the interval by which DAT line time-outs are detected. Refer to the Data Time-out Error in the Error Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency will be generated by dividing the sdclock TMCLK by this value. When setting this register, prevent inadvertent time-out events by clearing the Data Time-out Error Status Enable (in the Error Interrupt Status Enable register)</p> <p>0000_B TMCLK * 2¹³ 0001_B TMCLK * 2¹⁴ ... 1110_B TMCLK * 2²⁷ 1111_B Reserved</p> <p>This bit can be set and cleared only by software.</p>
0	[7:4]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

SDMMC_SW_RESET

A reset pulse is generated when writing 1 to each bit of this register. After completing the reset, the host controller shall clear each bit. Because it takes some time to complete software reset, the SD Host Driver shall confirm that these bits are 0.

SDMMC_SW_RESET

Software Reset Register

(002F_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
		0			SW_RST_ DAT_LINE	SW_RST_ CMD_LINE	SW_RST_ ALL
		r			rw	rw	rw

Field	Bits	Type	Description
SW_RST_ALL	0	rw	Software Reset for All
SW_RST_CMD_LINE	1	rw	<p>Software Reset for CMD Line Only part of command circuit is reset. The following registers and bits are cleared by this bit:</p> <p>Present State Register Command Inhibit (CMD)</p> <p>Normal Interrupt Status Register Command Complete</p> <p>0_B Work 1_B Reset</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
SW_RST_DAT _LINE	2	rw	<p>Software Reset for DAT Line Only part of data circuit is reset. The following registers and bits are cleared by this bit:</p> <p>Buffer Data Port Register Buffer is cleared and Initialized.</p> <p>Present State register Buffer read Enable Buffer write Enable Read Transfer Active Write Transfer Active DAT Line Active Command Inhibit (DAT)</p> <p>Block Gap Control register Continue Request Stop At Block Gap Request</p> <p>Normal Interrupt Status register Buffer Read Ready Buffer Write Ready Block Gap Event Transfer Complete</p> <p>0_B Work 1_B Reset</p>
0	[7:3]	r	<p>Reserved Read as 0; should be written with 0.</p>

SDMMC_INT_STATUS_NORM

The Normal Interrupt Status Enable affects read of this register, but Normal Interrupt Signal does not affect these reads. An Interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits except Card Interrupt and Error Interrupt, writing 1 to a bit clears it. The Card Interrupt is cleared when the card stops asserting the interrupt: that is when the Card Driver services the Interrupt condition.

SDMMC_INT_STATUS_NORM

Normal Interrupt Status Register

(0030_H)

Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR_INT	0				0		CARD_INTERRUPT	CARD_REMOVAL	CARD_INSERT	BUF_READ_READY	BUF_WRITE_READY	0	BLOCK_GAP_EVENT	TX_COMPLETE	CMD_COMPLETE
r	rw				r		r	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CMD_COMPLETE	0	rw	<p>Command Complete</p> <p>This bit is set when get the end bit of the command response (Except Auto CMD12). Command Time-out Error has higher priority than Command Complete. If both are set to 1, it can be considered that the response was not received correctly.</p> <p>0_B No Command Complete 1_B Command Complete</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
TX_COMPLETE	1	rw	<p>Transfer Complete</p> <p>This bit is set when a read / write transaction is completed.</p> <p>Read Transaction: This bit is set at the falling edge of Read Transfer Active Status. There are two cases in which the Interrupt is generated. The first is when a data transfer is completed as specified by data length (After the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request in the Block Gap Control Register (After valid data has been read to the Host System).</p> <p>Write Transaction: This bit is set at the falling edge of the DAT Line Active Status. There are two cases in which the Interrupt is generated. The first is when the last data is written to the card as specified by data length and Busy signal is released. The second is when data transfers are stopped at the block gap by setting Stop At Block Gap Request in the Block Gap Control Register and data transfers completed. (After valid data is written to the SD card and the busy signal is released). Transfer Complete has higher priority than Data Time-out Error. If both bits are set to 1, the data transfer can be considered complete</p> <p>0_B No Data Transfer Complete 1_B Data Transfer Complete</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
BLOCK_GAP_EVENT	2	rw	<p>Block Gap Event</p> <p>If the Stop At Block Gap Request in the Block Gap Control Register is set, this bit is set.</p> <p>Read Transaction: This bit is set at the falling edge of the DAT Line Active Status (When the transaction is stopped at SD Bus timing. The Read Wait must be supported in order to use this function).</p> <p>Write Transaction: This bit is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing).</p> <p>0_B No Block Gap Event 1_B Transaction stopped at Block Gap</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>
0	3	rw	<p>Reserved</p> <p>Read as 0; must be written with 0.</p>
BUFF_WRITE_READY	4	rw	<p>Buffer Write Ready</p> <p>This status is set if the Buffer Write Enable changes from 0 to 1.</p> <p>0_B Not Ready to Write Buffer. 1_B Ready to Write Buffer.</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>
BUFF_READ_READY	5	rw	<p>Buffer Read Ready</p> <p>This status is set if the Buffer Read Enable changes from 0 to 1.</p> <p>0_B Not Ready to read Buffer. 1_B Ready to read Buffer.</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
CARD_INS	6	rw	<p>Card Insertion</p> <p>This status is set if the Card Inserted in the Present State register changes from 0 to 1. When the host driver writes this bit to 1 to clear this status the status of the Card Inserted in the Present State register should be confirmed. Because the card detect may possibly be changed when the host driver clear this bit an Interrupt event may not be generated.</p> <p>0_B Card State Stable or Debouncing 1_B Card Inserted</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>
CARD_REMOVAL	7	rw	<p>Card Removal</p> <p>This status is set if the Card Inserted in the Present State register changes from 1 to 0. When the host driver writes this bit to 1 to clear this status the status of the Card Inserted in the Present State register should be confirmed. Because the card detect may possibly be changed when the host driver clear this bit an Interrupt event may not be generated.</p> <p>0_B Card State Stable or Debouncing 1_B Card Removed</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
CARD_INT	8	r	<p>Card Interrupt</p> <p>Writing this bit to 1 does not clear this bit. It is cleared by resetting the SD card interrupt factor. In 1-bit mode, the host controller shall detect the Card Interrupt without SD Clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so there are some sample delays between the interrupt signal from the card and the interrupt to the Host system.</p> <p>When this status has been set and the host driver needs to start this interrupt service, Card Interrupt Status Enable in the Normal Interrupt Status register shall be set to 0 in order to clear the card interrupt statuses latched in the host controller and stop driving the Host System. After completion of the card interrupt service (the reset factor in the SD card and the interrupt signal may not be asserted), set Card Interrupt Status Enable to 1 and start sampling the interrupt signal again.</p> <p>Interrupt detected by DAT[1] is supported when there is a card in slot.</p> <p>0_B No Card Interrupt 1_B Generate Card Interrupt</p>
0	[12:9]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>
0	[14:13]	rw	<p>Reserved</p> <p>Read as 0; must be written with 0.</p>
ERR_INT	15	r	<p>Error Interrupt</p> <p>If any of the bits in the Error Interrupt Status Register are set, then this bit is set. Therefore the host driver can test for an error by checking this bit first.</p> <p>0_B No Error. 1_B Error.</p> <p>This bit is initialized to 0 at reset.</p>

Table 13-7 Relation between transfer complete and data timeout error

Transfer Complete	Data Timeout Error	Meaning of the Status
0	0	Interrupted by Another Factor.
0	1	Timeout occur during transfer.
1	Don't Care	Data Transfer Complete

Table 13-8 Relation between command complete and command timeout error

Command Complete	Command Timeout Error	Meaning of the Status
0	0	Interrupted by Another Factor.
Don't Care	1	Response not received within 64 SDCLK cycles.
1	0	Response Received

SDMMC_INT_STATUS_ERR

Status defined in this register can be enabled by the Error Interrupt Status Enable Register, but not by the Error Interrupt Signal Enable Register. The Interrupt is generated when the Error Interrupt Signal Enable is enabled and at least one of the statuses is set to 1. Writing to 1 clears the bit and writing to 0 keeps the bit unchanged. More than one status can be cleared at a single register write.

SDMMC_INT_STATUS_ERR

Error Interrupt Status Register

(0032_H)

Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CEA TA_ ERR	0	0	0			ACM D_E RR	CUR REN T_LI MIT_ ERR	DAT A_E ND_ BIT_ ERR	DAT A_C RC_ ERR	DAT A_TI MEO UT_ ERR	CMD IND _ER R	CMD _EN D_BI T_E RR	CMD _CR _C_E RR	CMD _TIM EOU T_E RR
r	rw	rw	r	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CMD_TIMEOUT_ERR	0	rw	<p>Command Timeout Error</p> <p>Occurs only if the no response is returned within 64 SDCLK cycles from the end bit of the command. If the host controller detects a CMD line conflict, in which case Command CRC Error shall also be set. This bit shall be set without waiting for 64 SDCLK cycles because the command will be aborted by the host controller.</p> <p>0_B No Error 1_B Timeout</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
CMD_CRC_ERR	1	rw	<p>Command CRC Error</p> <p>Command CRC Error is generated in two cases. If a response is returned and the Command Time-out Error is set to 0, this bit is set to 1 when detecting a CRT error in the command response. The host controller detects a CMD line conflict by monitoring the CMD line when a command is issued. If the host controller drives the CMD line to 1 level, but detects 0 level on the CMD line at the next SDCLK edge, then the host controller shall abort the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error shall also be set to 1 to distinguish CMD line conflict.</p> <p>0_B No Error 1_B CRC Error Generated</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>
CMD_END_BIT_ERR	2	rw	<p>Command End Bit Error</p> <p>Occurs when detecting that the end bit of a command response is 0.</p> <p>0_B No Error 1_B End Bit Error Generated</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>
CMD_IND_ERR	3	rw	<p>Command Index Error</p> <p>Occurs if a Command Index error occurs in the Command Response.</p> <p>0_B No Error 1_B Error</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
DATA_TIMEOUT_ERR	4	rw	<p>Data Timeout Error Occurs when detecting one of following timeout conditions. Busy Timeout for R1b, R5b type. Busy Timeout after Write CRC status Write CRC status Timeout Read Data Timeout</p> <p>0_B No Error 1_B Timeout This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>
DATA_CRC_ERR	5	rw	<p>Data CRC Error Occurs when detecting CRC error when transferring read data which uses the DAT line or when detecting the Write CRC Status having a value of other than "010".</p> <p>0_B No Error 1_B Error This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>
DATA_END_BIT_ERR	6	rw	<p>Data End Bit Error Occurs when detecting 0 at the end bit position of read data which uses the DAT line or the end bit position of the CRC status.</p> <p>0_B No Error 1_B Error This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
CURRENT_LIMIT_ERR	7	rw	<p>Current Limit Error</p> <p>By setting the SD Bus Power bit in the Power Control Register, the host controller is requested to supply power for the SD Bus. If the host controller supports the Current Limit Function, it can be protected from an Illegal card by stopping power supply to the card in which case this bit indicates a failure status. Reading 1 means the host controller is not supplying power to SD card due to some failure. Reading 0 means that the host controller is supplying power and no error has occurred. This bit shall always set to be 0, if the host controller does not support this function.</p> <p>0_B No Error 1_B Power Fail</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>
ACMD_ERR	8	rw	<p>Auto CMD Error</p> <p>Auto CMD12 uses this error status. This bit is set when detecting that one of the bits D00-D04 in Auto CMD Error Status register has changed from 0 to 1. In case of Auto CMD12, this bit is set to 1, not only when the errors in Auto CMD12 occur but also when Auto CMD12 is not executed due to the previous command error.</p> <p>0_B No Error 1_B Error</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>
0	[10:9]	rw	<p>Reserved</p> <p>Read as 0; must be written with 0.</p>
0	11	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>
0	12	rw	<p>Reserved</p> <p>Read as 0; must be written with 0.</p>

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
CEATA_ERR	13	rw	<p>Ceata Error Status</p> <p>Occurs when ATA command termination has occurred due to an error condition the device has encountered.</p> <p>0_B no error 1_B error</p> <p>This bit can be cleared by a software write of 1 to the bit. A software write of 0 to the bit has no effect.</p>
0	[15:14]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Table 13-9 Relation between command CRC error and command time-out error

Command CRC Error	Command Time-out Error	Kinds of Error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD Line Conflict

SD/MMC Interface (SDMMC)

SDMMC_EN_INT_STATUS_NORM

Interrupt status can be enabled by writing 1 to the bit in this register. The host controller may sample the card Interrupt signal during interrupt period and may hold its value in the flip-flop. If the Card Interrupt Status Enable is set to 0, the host controller shall clear all internal signals regarding Card Interrupt.

SDMMC_EN_INT_STATUS_NORM

Normal Interrupt Status Enable Register(0034_H)

Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIXE D_T O_0			0				CAR D_IN T_E N	CAR D_R E_M O_V A_L _E N	CAR D_IN S_E N	BUF F_R E_A D_R E _E N	BUF F_W R_I T_E _R E _E N	0	BLO C_K _G A P _E V E N _E N	TX_ C_O M_P L_E T_E _E N	CMD _C O M_P L E T E _E N
r			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CMD_COMPLETE_EN	0	rw	Command Complete Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
TX_COMPLETE_EN	1	rw	Transfer Complete Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
BLOCK_GAP_EVENT_EN	2	rw	Block Gap Event Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
0	3	rw	Reserved Read as 0; must be written with 0.

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
BUFF_WRITE_READY_EN	4	rw	Buffer Write Ready Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
BUFF_READ_READY_EN	5	rw	Buffer Read Ready Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CARD_INS_EN	6	rw	Card Insertion Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CARD_REMOVAL_EN	7	rw	Card Removal Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CARD_INT_EN	8	rw	Card Interrupt Status Enable If this bit is set to 0, the host controller shall clear Interrupt request to the System. The Card Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The host driver may clear the Card Interrupt Status Enable before servicing the Card Interrupt and may set this bit again after all Interrupt requests from the card are cleared to prevent inadvertent Interrupts. 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
0	[14:9]	rw	Reserved Read as 0; must be written with 0.
FIXED_TO_0	15	r	Fixed to 0 The host controller shall control error Interrupts using the Error Interrupt Status Enable register.

SD/MMC Interface (SDMMC)

SDMMC_EN_INT_STATUS_ERR

Interrupt status can be enabled by writing 1 to the bit in this register. To Detect CMD Line conflict, the host driver must set both Command Time-out Error Status Enable and Command CRC Error Status Enable to 1.

SDMMC_EN_INT_STATUS_ERR

Error Interrupt Status Enable Register(0036_H)

Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CEA TA_ ERR_ EN	TAR GET_ RE_ SP_ ERR_ EN	0	0	0	0	ACM REN D_E RR_ EN	CUR REN T_LI MIT_ ERR_ EN	DAT A_E ND_ BIT_ ERR_ EN	DAT A_C RC_ ERR_ EN	DAT A_TI MEO UT_ ERR_ EN	CMD _IND _ER R_E N	CMD _EN D_BI T_E RR_ EN	CMD _CR C_E RR_ EN	CMD _TIM EOU T_E RR_ EN
r	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CMD_TIMEOUT_ ERR_EN	0	rw	Command Timeout Error Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CMD_CRC_ERR_ EN	1	rw	Command CRC Error Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CMD_END_BIT_ ERR_EN	2	rw	Command End Bit Error Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CMD_IND_ERR_ EN	3	rw	Command Index Error Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
DATA_TIMEOUT _ERR_EN	4	rw	Data Timeout Error Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
DATA_CRC_ERR_EN	5	rw	Data CRC Error Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
DATA_END_BIT_ERR_EN	6	rw	Data End Bit Error Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CURRENT_LIMIT_ERR_EN	7	rw	Current Limit Error Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
ACMD_ERR_EN	8	rw	Auto CMD12 Error Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
0	[10:9]	rw	Reserved Read as 0; must be written with 0.
0	11	r	Reserved Read as 0; should be written with 0.
TARGET_RESP_ERR_EN	12	rw	Target Response Error Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CEATA_ERR_EN	13	rw	Ceata Error Status Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
0	[15:14]	r	Reserved Read as 0; should be written with 0.

SD/MMC Interface (SDMMC)

SDMMC_EN_INT_SIGNAL_NORM

This register is used to select which interrupt status is indicated to the Host System as the Interrupt. The interrupt line is shared by all the status bits. Interrupt generation can be enabled by writing 1 to any of these bits.

SDMMC_EN_INT_SIGNAL_NORM

Normal Interrupt Signal Enable Register(0038_H)

Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIXE D_T O_0			0				CAR D_IN T_E N	CAR D_R E_M O V A L _E N	CAR D_IN S_E N	BUF F_R E_A D _R E A D Y _E N	BUF F_W R_I T E _R E A D Y _E N	0	BLO C K _G A P _E V E N T _E N	TX _C O M P L E T E _E N	CMD _C O M P L E T E _E N
r			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CMD_COMPLETE_EN	0	rw	Command Complete Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
TX_COMPLETE_EN	1	rw	Transfer Complete Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
BLOCK_GAP_EVENT_EN	2	rw	Block Gap Event Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
0	3	rw	Reserved Read as 0; must be written with 0.
BUFF_WRITE_READY_EN	4	rw	Buffer Write Ready Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
BUFF_READ_READY_EN	5	rw	Buffer Read Ready Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CARD_INS_EN	6	rw	Card Insertion Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CARD_REMOVAL_EN	7	rw	Card Removal Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CARD_INT_EN	8	rw	Card Interrupt Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
0	[14:9]	rw	Reserved Read as 0; must be written with 0.
FIXED_TO_0	15	r	Fixed to 0 The host driver shall control error Interrupts using the Error Interrupt Signal Enable register.

SDMMC_EN_INT_SIGNAL_ERR

This register is used to select which interrupt status is notified to the Host System as the Interrupt. The interrupt line is shared by all the status bits. Interrupt generation can be enabled by writing 1 to any of these bits.

SDMMC_EN_INT_SIGNAL_ERR

Error Interrupt Signal Enable Register(003A_H)

Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CEA TA_ ERR _EN	TAR GET _RE SP_ ERR _EN	0	0	0	0	ACM REN D_E RR_ ERR _EN	CUR REN T_LI MIT_ ERR _EN	DAT A_E ND_ BIT_ ERR _EN	DAT A_C RC_ ERR _EN	DAT A_TI MEO UT_ ERR _EN	CMD _IND _ER R_E N	CMD _EN D_BI T_E RR_ EN	CMD _CR C_E RR_ EN	CMD _TIM EOU T_E RR_ EN
r	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CMD_TIMEOUT_ERR_EN	0	rw	Command Timeout Error Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CMD_CRC_ERR_EN	1	rw	Command CRC Error Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CMD_END_BIT_ERR_EN	2	rw	Command End Bit Error Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CMD_IND_ERR_EN	3	rw	Command Index Error Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
DATA_TIMEOUT_ERR_EN	4	rw	Data Timeout Error Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
DATA_CRC_ERR_EN	5	rw	Data CRC Error Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
DATA_END_BIT_ERR_EN	6	rw	Data End Bit Error Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CURRENT_LIMIT_ERR_EN	7	rw	Current Limit Error Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
ACMD_ERR_EN	8	rw	Auto CMD12 Error Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
0	[10:9]	rw	Reserved Read as 0; must be written with 0.
0	11	r	Reserved Read as 0; should be written with 0.
TARGET_RESP_ERR_EN	12	rw	Target Response Error Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
CEATA_ERR_EN	13	rw	Ceata Error Signal Enable 0 _B Masked 1 _B Enabled This bit can be set and cleared only by software.
0	[15:14]	r	Reserved Read as 0; should be written with 0.

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
ACMD_CRC_ERR	2	r	Auto CMD CRC Error Occurs when detecting a CRC error in the command response. 0 _B No Error 1 _B CRC Error Generated This bit is initialized to 0 at reset.
ACMD_END_BIT_ERR	3	r	Auto CMD End Bit Error Occurs when detecting that the end bit of command response is 0. 0 _B No Error 1 _B End Bit Error Generated This bit is initialized to 0 at reset.
ACMD_IND_ERR	4	r	Auto CMD Index Error Occurs if the Command Index error occurs in response to a command. 0 _B No Error 1 _B Error This bit is initialized to 0 at reset.
0	[6:5]	r	Reserved Read as 0; should be written with 0.
CMD_NOT_ISSUED_BY_ACMD12_ERR	7	r	Command Not Issued By Auto CMD12 Error Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 error (D04 - D01) in this register. 0 _B No Error 1 _B Not Issued This bit is initialized to 0 at reset.
0	[15:8]	r	Reserved Read as 0; should be written with 0.

Table 13-10 Relation between Auto CMD12 CRC error and Auto CMD12 timeout error

Auto Cmd12 CRC Error	Auto CMD12 Timeout Error	Kinds of Error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD Line Conflict

The timing of changing Auto CMD12 Error Status can be classified in three scenarios:

1. When the host controller is going to issue Auto CMD12.
 - a) Set D00 to 1 if Auto CMD12 cannot be issued due to an error in the previous command.
 - b) Set D00 to 0 if Auto CMD12 is issued.
2. At the end bit of Auto CMD12 response.
 - a) Check received responses by checking the error bits D01, D02, D03, D04.
 - b) Set to 1 if Error is Detected.
 - c) Set to 0 if Error is Not Detected.
3. Before reading the Auto CMD12 Error Status bit D07
 - a) Set D07 to 1 if there is a command cannot be issued.
 - b) Set D07 to 0 if there is no command to issue.

Timing of generating the Auto CMD12 Error and writing to the Command register are Asynchronous. Then D07 shall be sampled when driver never writing to the Command register. So just before reading the Auto CMD12 Error Status register is good timing to set the D07 status bit.

SD/MMC Interface (SDMMC)

SDMMC_CAPABILITIES

This register provides the host controller implementation information.

SDMMC_CAPABILITIES

Capabilities Register

(0040_H)

Reset Value: 01A030B0_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOT_TY PE	ASY NC_I NT_ SUP POR T	SYS BUS _64_ SUP POR T	0	VOL TAG E_S UPP ORT _1_8 V	VOL TAG E_S UPP ORT _3V	VOL TAG E_S UPP ORT _3_3 V	SUS PEN D_R ESU ME_ SUP POR T	SDM A_S UPP ORT	HIG H_S PEE D_S UPP ORT	0	ADM A2_ SUP POR T	EXT _ME DIA_ BUS_ SU PPO RT	MAX_BLO CK_LENG TH		
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE_SD_CLOCK_FREQ							TIME OUT _CL OCK _UNI T	0	TIMEOUT_CLOCK_FREQ						
							r	r							

Field	Bits	Type	Description
TIMEOUT_CLOCK_FREQ	[5:0]	r	Timeout Clock Frequency Base clock frequency used to detect Data Timeout Error 110000 _B 48 MHz
0	6	r	Reserved Read as 0.
TIMEOUT_CLOCK_UNIT	7	r	Timeout Clock Unit Unit of base clock frequency used to detect Data Timeout Error 1 _B MHz
BASE_SD_CLOCK_FREQ	[15:8]	r	Base Clock Frequency for SD Clock 6-bit Base Clock Frequency <i>Note: Upper 2-bit is always 0.</i> 30 _H 48 MHz

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
MAX_BLOCK_LEN GTH	[17:16]	r	Max Block Length Maximum block size that the host driver can read and write to the buffer in the host controller. The buffer shall transfer this block size without wait cycles. 00 _B 512 byte
EXT_MEDIA_BUS_SUPPORT	18	r	Extended Media Bus Support 0 _B Extended Media Bus not supported
ADMA2_SUPPORT	19	r	ADMA2 Support 0 _B ADMA not supported
0	20	r	Reserved Read as 0.
HIGH_SPEED_SUPPORT	21	r	High Speed Support 1 _B High Speed supported
SDMA_SUPPORT	22	r	SDMA Support 0 _B SDMA not supported
SUSPEND_RESUME_SUPPORT	23	r	Suspend / Resume Support 1 _B Supported
VOLTAGE_SUPPORT_3_3V	24	r	Voltage Support 3.3V 1 _B 3.3V supported
VOLTAGE_SUPPORT_3_0V	25	r	Voltage Support 3.0V 0 _B 3.0V not supported
VOLTAGE_SUPPORT_1_8V	26	r	Voltage Support 1.8V 0 _B 1.8V not supported
0	27	r	Reserved Read as 0.
SYSBUS_64_SUPPORT	28	r	64-bit System Bus Support 0 _B Does not support 64-bit system address
ASYNC_INTERRUPT_SUPPORT	29	r	Asynchronous Interrupt Support 0 _B Asynchronous Interrupt not supported
SLOT_TYPE	[31:30]	r	Slot Type 00 _B Removable Card Slot

SDMMC_CAPABILITIES_HI

This register provides the host controller implementation information.

SDMMC_CAPABILITIES_HI

Capabilities Register High

(0044_H)

Reset Value: 03000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			0			1	1								
			r			r	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RE_TUNING_MODES	USE_TUNING_SDR5		0						DRV_D_SUPPORT	DRV_C_SUPPORT	DRV_A_SUPPORT	0	DDR50_SUPPORT	SDR104_SUPPORT	SDR50_SUPPORT
r	r	r	r					r	r	r	r	r	r	r	r

Field	Bits	Type	Description
SDR50_SUPPORT	0	r	SDR50 Support 0 _B SDR50 is not supported
SDR104_SUPPORT	1	r	SDR104 Support 0 _B SDR104 is not supported
DDR50_SUPPORT	2	r	DDR50 Support 0 _B DDR50 is not supported
0	3	r	Reserved Read as 0.
DRV_A_SUPPORT	4	r	Driver Type A Support 0 _B Driver Type A is not supported
DRV_C_SUPPORT	5	r	Driver Type C Support 0 _B Driver Type C is not supported
DRV_D_SUPPORT	6	r	Driver Type D Support 0 _B Driver Type D is not supported
0	7	r	Reserved Read as 0.
TIM_CNT_RETUNE	[11:8]	r	Timer count for Re-Tuning Retuning is not used. 0 _H Get information via other source

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
0	12	r	Reserved Read as 0.
USE_TUNING_SDR50	13	r	Use Tuning for SDR50 Tuning is not required. 0 _B SDR50 does not require tuning
RE_TUNING_MODES	[15:14]	r	Re-tuning modes Retuning if not used. 00 _B Mode 1
CLK_MULT	[23:16]	r	Clock Multiplier 00 _H Clock Multiplier not supported
1	24, 25	r	Reserved Read as 1.
0	[31:26]	r	Reserved Read as 0.

SD/MMC Interface (SDMMC)

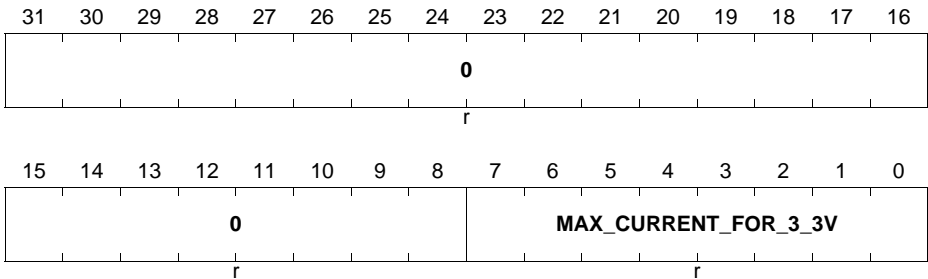
SDMMC_MAX_CURRENT_CAP

This register indicates the maximum current capability for the supported voltage.

SDMMC_MAX_CURRENT_CAP

Maximum Current Capabilities Register(0048_H)

Reset Value: 00000001_H



Field	Bits	Type	Description
MAX_CURRENT_FOR_3_3V	[7:0]	r	Maximum Current for 3.3V
0	[31:8]	r	Reserved Read as 0.

Table 13-11 Maximum current value definition

Register Value	Current Value
0	Get information via another method
1	4 mA
2	8 mA
3	12 mA
...	...
255	1020 mA

SD/MMC Interface (SDMMC)

SDMMC_FORCE_EVENT_ACMD_ERR_STATUS

The Force Event Register is an address at which the Auto CMD12 Error Status Register can be written.

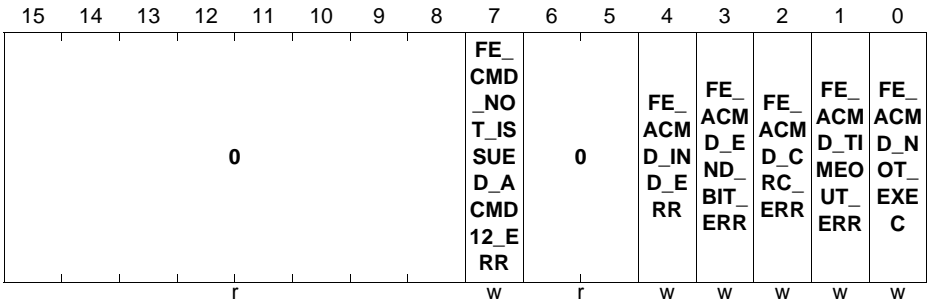
Writing 1 : set each bit of the Auto CMD12 Error Status Register

Writing 0 : no effect.

SDMMC_FORCE_EVENT_ACMD_ERR_STATUS

Force Event Register for Auto CMD Error Status(0050_H)

Reset Value: 0000_H



Field	Bits	Type	Description
FE_ACMD_NOT_EXEC	0	w	Force Event for Auto CMD12 NOT Executed 0 _B No interrupt 1 _B Interrupt is generated
FE_ACMD_TIMEOUT_ERR	1	w	Force Event for Auto CMD timeout Error 0 _B No interrupt 1 _B Interrupt is generated
FE_ACMD_CRC_ERR	2	w	Force Event for Auto CMD CRC Error 0 _B No interrupt 1 _B Interrupt is generated
FE_ACMD_END_BIT_ERR	3	w	Force Event for Auto CMD End bit Error 0 _B No interrupt 1 _B Interrupt is generated
FE_ACMD_IND_ERR	4	w	Force Event for Auto CMD Index Error 0 _B No interrupt 1 _B Interrupt is generated
0	[6:5]	r	Reserved Read as 0; should be written with 0.

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
FE_CMD_NOT_ISSUE D_ACMD12_ERR	7	w	Force Event for CMD not issued by Auto CMD12 Error 0 _B No interrupt 1 _B Interrupt is generated
0	[15:8]	r	Reserved Read as 0; should be written with 0.

SDMMC_FORCE_EVENT_ERR_STATUS

The Force Event Register is an address at which the Error Interrupt Status register can be written. The effect of a write to this address will be reflected in the Error Interrupt Status Register if the corresponding bit of the Error Interrupt Status Enable Register is set.

Writing 1 : set each bit of the Error Interrupt Status Register

Writing 0 : no effect

SDMMC_FORCE_EVENT_ERR_STATUS

Force Event Register for Error Interrupt Status(0052_H)

Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FE_CEA TA_ERR	FE_T ARG ET_ RES PON SE_ ERR	0	0	0	FE_ ACM D12 ERR	FE_ CUR REN T_LI MIT_ ERR	FE_ DAT A_E ND_ BIT_ ERR	FE_ DAT A_C RC_ ERR	FE_ DAT A_TI MEO UT_ ERR	FE_ CMD _IND _ER R	FE_ CMD _EN D_BI T_E RR	FE_ CMD _CR C_E RR	FE_ CMD _TIM EOU T_E RR	
W	W	W	r	r	W	W	W	W	W	W	W	W	W	W	W

Field	Bits	Type	Description
FE_CMD_TIMEOUT_ERR	0	w	Force Event for Command Timeout Error 0 _B No interrupt 1 _B Interrupt is generated
FE_CMD_CRC_ERR	1	w	Force Event for Command CRC Error 0 _B No interrupt 1 _B Interrupt is generated
FE_CMD_END_BIT_ERR	2	w	Force Event for Command End Bit Error 0 _B No interrupt 1 _B Interrupt is generated
FE_CMD_IND_ERR	3	w	Force Event for Command Index Error 0 _B No interrupt 1 _B Interrupt is generated
FE_DATA_TIMEOUT_ERR	4	w	Force Event for Data Timeout Error 0 _B No interrupt 1 _B Interrupt is generated
FE_DATA_CRC_ERR	5	w	Force Event for Data CRC Error 0 _B No interrupt 1 _B Interrupt is generated

SD/MMC Interface (SDMMC)

Field	Bits	Type	Description
FE_DATA_END_BIT_ERR	6	w	Force Event for Data End Bit Error 0 _B No interrupt 1 _B Interrupt is generated
FE_CURRENT_LIMIT_ERR	7	w	Force Event for Current Limit Error 0 _B No interrupt 1 _B Interrupt is generated
FE_ACMD12_ERR	8	w	Force Event for Auto CMD Error 0 _B No interrupt 1 _B Interrupt is generated
0	9	w	Reserved Must be written with 0.
0	[11:10]	r	Reserved Read as 0; should be written with 0.
FE_TARGET_RESPONSE_ERR	12	w	Force event for Target Response Error 0 _B No interrupt 1 _B Interrupt is generated
FE_CEATA_ERR	13	w	Force Event for Ceata Error 0 _B No interrupt 1 _B Interrupt is generated
0	[15:14]	w	Reserved Must be written with 0.

SDMMC_DEBUG_SEL

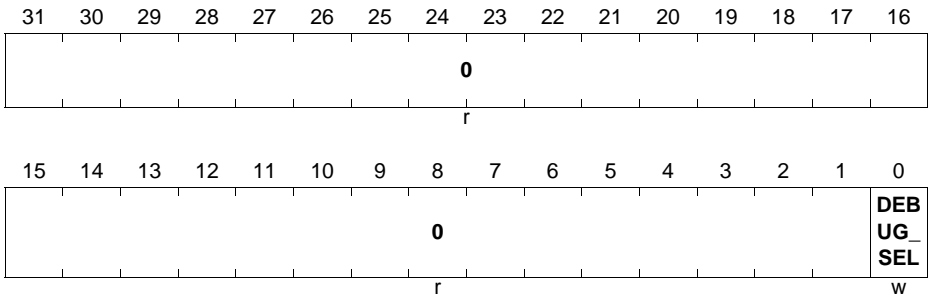
This register is used to select the debug mode.

SDMMC_DEBUG_SEL

Debug Selection Register

(0074_H)

Reset Value: 00000000_H



Field	Bits	Type	Description
DEBUG_SEL	0	w	Debug_sel 0 _B receiver module and fifo_ctrl module signals are probed out 1 _B cmd register, Interrupt status, transmitter module and clk sdcard signals are probed out.
0	[31:1]	r	Reserved Read as 0; should be written with 0.

SDMMC_SLOT_INT_STATUS

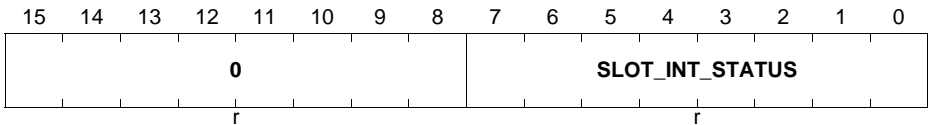
This register is used to configure the interrupt signal for card slot.

SDMMC_SLOT_INT_STATUS

Slot Interrupt Status Register

(00FC_H)

Reset Value: 0000_H



Field	Bits	Type	Description
SLOT_INT_STATUS	[7:0]	r	<p>Interrupt Signal for Card Slot</p> <p>These status bit indicate the Interrupt signal and Wakeup signal for the card slot. By a power on reset or by Software Reset For All, the Interrupt signal shall be de asserted and this status shall read 00_H. 00_H Slot 1</p> <p><i>Note: Other values are reserved</i></p> <p>This bit is initialized to 0 at reset.</p>
0	[15:8]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

13.13 Interconnects

The interface signals of the SDMMC Host Controller are described below.

Table 13-12 SDMMC Pin Connections

Input/ Output	I/O	Connected to	Description
clk_xin	I	SCU.EXTCLK	Input clock to SDMMC controller
SDMMC.CLK_IN	I	P3.6	Feedback clock of clk_sdcard_out from the pads
SDMMC.DATA7_IN	I	P1.13	MMC8 mode: Data7 Input
SDMMC.DATA6_IN	I	P1.12	MMC8 mode: Data6 Input
SDMMC.DATA5_IN	I	P1.9	MMC8 mode: Data5 Input
SDMMC.DATA4_IN	I	P1.8	MMC8 mode: Data4 Input
SDMMC.DATA3_IN	I	P4.1	SD4/MMC8 mode: Data3 Input
SDMMC.DATA2_IN	I	P1.7	SD4/MMC8 : Data2 input
SDMMC.DATA1_IN	I	P1.6	SD1 mode: Interrupt SD4 mode: Data1 Input or Interrupt (optional) MMC8 mode: Data1 Input
SDMMC.DATA0_IN	I	P4.0	SD1/SD4/MMC8 : Data0 Input
SDMMC.SDCD	I	SCU	Active low. Card Detection
SDMMC.SDWC	I	SCU	Active High. SD Card Write Protect
SDMMC.CMD_IN	I	P3.5	SD1/SD4/MMC8 : Command Input
SDMMC.CLK_OUT	O	P3.6	Clock supplied to SD/MMC card
SDMMC.DATA7_OUT	O	P1.13	MMC8 mode: Data7 Output
SDMMC.DATA6_OUT	O	P1.12	MMC8 mode: Data6 Output
SDMMC.DATA5_OUT	O	P1.9	MMC8 mode: Data5 Output
SDMMC.DATA4_OUT	O	P1.8	MMC8 mode: Data4 Output
SDMMC.DATA3_OUT	O	P4.1	SD4/MMC8 mode: Data3 Output
SDMMC.DATA2_OUT	O	P1.7	SD1 mode: Read Wait(optional) SD4 mode: Data2 Output or Read Wait (optional) MMC8 mode: Data2 Output
SDMMC.DATA1_OUT	O	P1.6	SD4/MMC8 : Data1 Output
SDMMC.DATA0_OUT	O	P4.0	SD1/SD4/MMC8 : Data0 Output

Table 13-12 SDMMC Pin Connections (cont'd)

Input/ Output	I/O	Connected to	Description
SDMMC.CMD_OUT	O	P3.5	SD1/SD4/MMC8 : Command Output
SDMMC.BUS_POWER	O	P3.4	Control Card Power Supply
SDMMC.LED	O	P3.3	LED indication
SDMMC.RST	O	P0.11	Hardware reset to card
SDMMC.SR0	O	NVIC	Service request line

SD/MMC Interface (SDMMC)

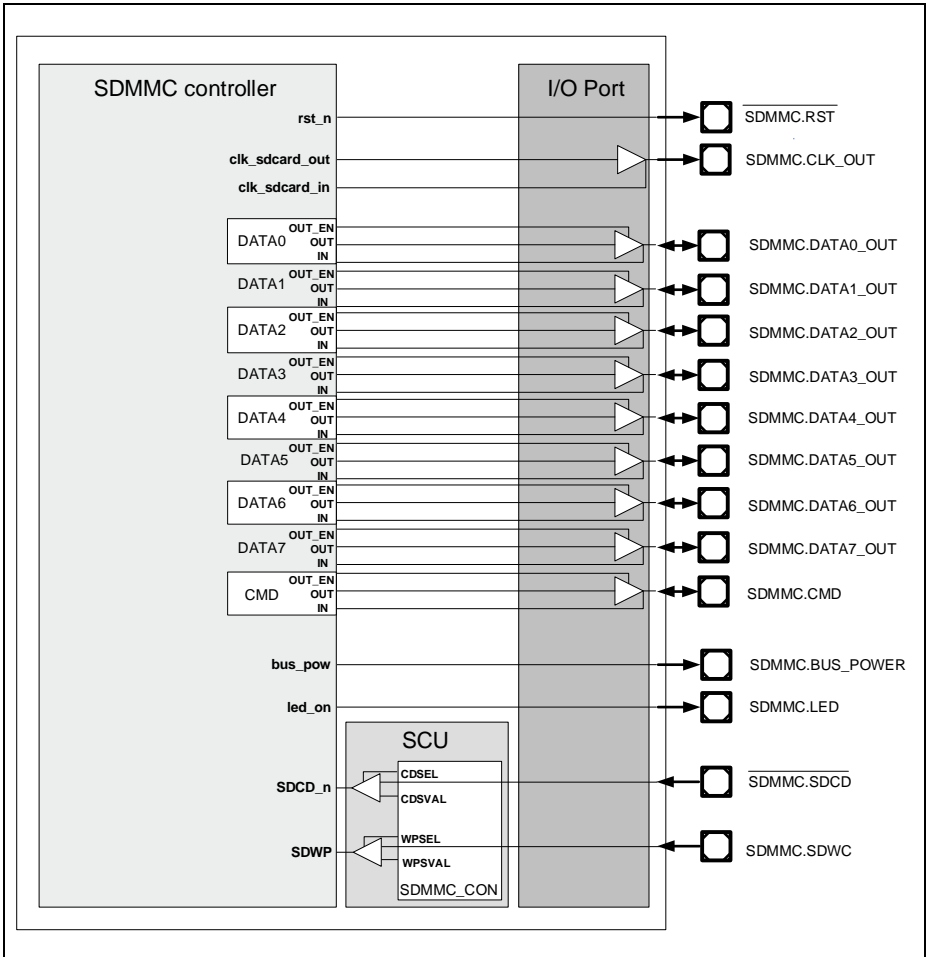


Figure 13-4 External Pin Connections of SDMMC

14 Ethernet MAC (ETH)

The Ethernet MAC (ETH) is a major communication peripheral that supports 10/100 MBit/s data transfer rates in compliance with the IEEE 802.3-2002 standard. The ETH may be used to implement Internet connected applications using IPv4 and IPv6. The ETH also includes support for IEEE1588 time synchronisation to allow implementation of Real Time Ethernet protocols.

Table 14-1 Abbreviations

ETH	Ethernet MAC Peripheral
MTL	MAC Transaction Layer
PHY	Physical Layer Interface
MMC	MAC Management Counters
SMI	Station Management Interface
COE	Checksum Offload Engine
PMT	Power Management
MII	Media Independent Interface
RMII	Reduced media Independent interface

The following document is reprinted with permission of Synopsys Inc. No disclosure of Databook to Synopsys' Competitors without Synopsys consent. List of Competitors is available from Infineon Technologies AG or Synopsys Inc.

14.1 Overview

The ETH peripheral is comprised of five major functional units. The ETH-Core takes user provided data frames and formats them for transmission to an external PHY via an MII or RMII interface. The ETH MAC Transaction Layer (MTL) acts as a bridge between the application and the ETH Core. The MTL provides two 2K byte FIFO's to buffer the transmit and receive frames. The application may write data frames directly to the MTL (cut through mode) or more normally will use the dedicated ETH DMA unit. The ETH DMA allows the application to define a region of RAM to be used as transmit and receive buffers. DMA transfers are initiated by DMA descriptors which are also held in RAM. The ETH also includes a system time module which allows timestamping of transmit and receive frames. The ETH also includes an extensive set of MAC Management counters which provide detailed bus statistics.

The ETH includes the following features, listed by category.

14.1.1 ETH Core Features

- Supports 10/100-Mbit/s data transfer rates with the following PHY interfaces
 - IEEE 802.3-compliant RMII/MII (default) interface to communicate with an external Fast Ethernet PHY
- Supports both full-duplex and half-duplex operation
 - Supports CSMA/CD Protocol for half-duplex operation
 - Supports IEEE 802.3x flow control for full-duplex operation
 - Optional forwarding of received pause control frames to the user application in full-duplex operation
 - Back-pressure support for half-duplex operation
 - Automatic transmission of zero-quantum pause frame on deassertion of flow control input in full-duplex operation
- Preamble and start-of-frame data (SFD) insertion in Transmit, and deletion in Receive paths
- Automatic CRC and pad generation controllable on a per-frame basis
- Options for Automatic Pad/CRC Stripping on receive frames
- Programmable frame length to support Standard or Jumbo Ethernet frames with sizes up to 16 KB
- Programmable InterFrameGap (40-96 bit times in steps of 8)
- Supports a variety of flexible address filtering modes:
 - Up to 3 additional 48-bit perfect (DA) address filters with masks for each byte
 - Up to 3 48-bit SA address comparison check with masks for each byte
 - 64-bit Hash filter for multicast and uni-cast (DA) addresses
 - Option to pass all multicast addressed frames
 - Promiscuous mode support to pass all frames without any filtering for network monitoring
 - Passes all incoming packets (as per filter) with a status report
- Separate 32-bit status returned for transmission and reception packets
- Supports IEEE 802.1Q VLAN tag detection for reception frames
- Separate transmission, reception, and control interfaces to the Application
- Supports 32-bit data transfer interface on the system-side
- Complete network statistics with RMON/MIB Counters (RFC1757/RFC2819 / RFC2665). It is completely under control of higher protocol level (SW) to make use of these counters.
- MDIO Master interface for PHY device configuration and management, e.g. for switching the PHY in external loopback mode.
- Detection of LAN wake-up frames and AMD Magic Packet frames
- Enhanced Receive module for checking IPv4 header checksum and TCP, UDP, or ICMP checksum encapsulated in IPv4 or IPv6 datagrams.
- Module to support Ethernet frame time stamping as described in IEEE 1588-2008. Sixty-four-bit time stamps are given in each frame's transmit or receive status.

14.1.2 DMA Block Features

The DMA block exchanges data between the MTL block and the XMC4300 memory. A set of registers (DMA CSR) to control DMA operation is accessible by the XMC4300.

DMA features include:

- 32-bit data transfers
- Single-channel Transmit and Receive engines
- Fully synchronous design operating on a single system clock (except for CSR module, when a separate CSR clock is configured)
- Optimization for packet-oriented DMA transfers with frame delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) or linked-list (chained) descriptor chaining
- Descriptor architecture, allowing large blocks of data transfer with minimum CPU intervention; each descriptor can transfer up to 8 KB of data
- Comprehensive status reporting for normal operation and transfers with errors
- Individual programmable burst size for Transmit and Receive DMA Engines for optimal bus utilization
- Programmable interrupt options for different operational conditions
- Per-frame Transmit/Receive complete interrupt control
- Round-robin or fixed-priority arbitration between Receive and Transmit engines
- Start/Stop modes
- Separate ports for CPU CSR access and data interface

14.1.3 Transaction Layer (MTL) Features

The MTL block consists of two sets of FIFOs: a Transmit FIFO with programmable threshold capability, and a Receive FIFO with a configurable threshold (default of 64 bytes).

MTL features include:

- 32-bit Transaction Layer block providing a bridge between the application and the -CORE
- Single-channel Transmit and Receive engines
- Data transfers executed using simple FIFO-protocol
- Synchronization for all clocks in the design (Transmit, Receive and system clocks)
- Optimization for packet-oriented transfers with frame delimiters
- Four Separate ports for system-side and -CORE-side transmission and reception
- Two RAM-based asynchronous FIFOs of 2K Bytes depth with synchronous/asynchronous Read and Write operation with respect to the Read and Write clocks (one for transmission and one for reception)
- Receive Status vectors inserted into the Receive FIFO after the EOF transfer enables multiple-frame storage in the Receive FIFO without requiring another FIFO to store those frames' Receive Status.

Ethernet MAC (ETH)

- Configurable Receive FIFO threshold (default fixed at 64 bytes) in Cut-Through mode
- Option to filter all error frames on reception and not forward them to the application in Store-and-Forward mode
- Option to forward under-sized good frames
- Supports statistics by generating pulses for frames dropped or corrupted (due to overflow) in the Receive FIFO
- Supports Store and Forward mechanism for transmission to the core
- Supports threshold control for transmit buffer management
- Supports configurable number of frames to be stored in FIFO at any time. The default is up to 8 frames in -MTL configuration.
- Automatic generation of PAUSE frame control or backpressure signal to the -core based on Receive FIFO-fill (threshold configurable) level.
- Handles automatic retransmission of Collision frames for transmission
- Discards frames on late collision, excessive collisions, excessive deferral and underrun conditions
- Software control to flush Tx FIFO
- Data FIFO RAM chip-select disabled when inactive, to reduce power consumption
- module to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum in frames transmitted in Store-and-Forward mode.

14.1.4 Monitoring, Test, and Debugging Support Features

- Supports internal loopback on the MII for debugging
- External loopback is supported via the integrated MDIO controlling the PHY
- DMA states (Tx and Rx) given as status bits
- Debug status register that gives status of FSMs in Transmit and Receive data-paths and FIFO fill-levels.
- Application Abort status bits
- MMC (RMON) module in the core
- Current Tx/Rx Buffer pointer as status registers
- Current Tx/Rx Descriptor pointer as status registers

14.1.5 Block Diagram

A block diagram of the ETH's major system configurations is provided in [Figure 14-1](#).

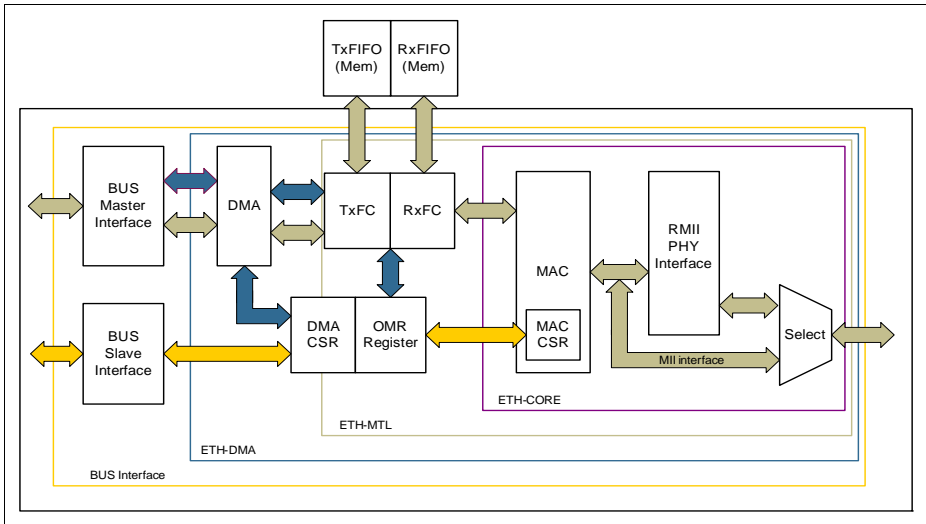


Figure 14-1 ETH Block Diagram

14.2 Functional Description

This chapter describes the structure and programming requirements of the features within the ETH subsystem. Each significant programming feature is discussed in a separate section.

14.2.1 ETH Core

The ETH core supports two interfaces towards the PHY chip, MII and RMII. The PHY interface can be selected only once after reset. The ETH core communicates with the application side with the MAC Transmit Interface (MTI), MAC Receive Interface (MRI) and the MAC Control Interface (MCI).

14.2.1.1 Transmission

Transmission is initiated when the MTL Application pushes in data with the SOF . When the SOF signal is detected, the ETH accepts the data and begins transmitting to the MII. The time required to transmit the frame data to the RMII/MII after the Application initiates transmission is variable, depending on delay factors like IFG delay, time to transmit preamble/SFD, and any back-off delays for Half-Duplex mode. Until then, the ETH does not accept the data received from MTL.

After the EOF is transferred to the ETH Core, the core complete normal transmission and then gives the Status of Transmission back to the MTL. If a normal collision (in Half-duplex mode) occurs during transmission, the ETH core makes valid the Transmit Status to the MTL. It then accepts and drops all further data until the next SOF is received. The MTL block should retransmit the same frame from SOF on observing a Retry request (in the Status) from the ETH.

The ETH issues an underflow status if the MTL is not able to provide the data continuously during the transmission. During the normal transfer of a frame from MTL, if the ETH receives a SOF without getting an EOF for the previous frame, then it (the SOF) is ignored and the new frame is considered as continuation of the previous frame.

The following six modules constitute the transmission function of the ETH:

- Transmit Bus Interface Module (TBU)
- Transmit Frame Controller Module (TFC)
- Transmit Protocol Engine Module (TPE)
- Transmit Scheduler Module (STX)
- Transmit CRC Generator Module (CTX)
- Transmit Flow Control Module (FTX)

Transmit Bus Interface Module

This module interfaces the transmit path of the ETH core with the external frame with a FIFO interface.

This module also outputs the (32-bit) Transmit Status to the application at the end of normal transmission or collision.

Additionally, this module outputs the Transmit Snapshot register value.

Transmit Frame Controller Module

The Transmit Frame Controller (TFC) consists of two registers to hold data, byte enables, and the last data control received from the TBU. The register provides a buffer between the Application and the TPE to regulate data flow as well as converts the input data into an 8-bit bus towards the TPE.

When the number of bytes received from the Application falls below 60 (DA+SA+LT+DATA), the state machine that interfaces with the TBU automatically appends zeros to the transmitting frame to make the data length exactly 46 bytes to meet the minimum data field requirement of IEEE 802.3. The ETH can be programmed not to append any padding.

The cyclic redundancy check (CRC) for the Frame Check Sequence (FCS) field is calculated before transmission to the TPE module. This value is computed by CTX module. The TFC module receives the computed CRC and appends it to the data being transmitted to the TPE module. When the ETH is programmed to not append the CRC value to the end of Ethernet frames, the TFC module ignores the computed CRC and transmits only the data received from the TBU module to the TPE module. An exception to this rule is that when the ETH is programmed to append pads for frames (DA+SA+LT+DATA) less than 60 bytes sent by the TBU module, the TFC module will append the CRC at the end of padded frame.

The TFC converts the data received from the TBU into 8-bit data for the TPE module.

Transmit Protocol Engine Module

The Transmit Protocol Engine (TPE) module consists of a transmit state machine that controls the operation of Ethernet frame transmission. The module's transmit state machine performs the following functions to meet the IEEE 802.3 specifications.

- Generates preamble and SFD
- Generates jam pattern in Half-Duplex mode
- Jabber timeout
- Flow control for Half-Duplex mode (back pressure)
- Generates transmit frame status
- Contains time stamp snapshot logic for IEEE 1588 support

When a new frame transmission from the TFC is requested, the transmit state machine sends out the preamble and SFD, followed by the data received. The preamble is defined as 7 bytes of 10101010_B pattern, and the SFD is defined as 1 byte of 10101011_B pattern.

The collision window is defined as 1 slot time (512 bit times for 10/100 Mbit/s Ethernet). The jam pattern generation is applicable only to Half-Duplex mode, not to Full-Duplex mode. In Full-Duplex mode, the transmit state machine ignores the collision signal from the PHY.

In MII mode, if a collision occurs any time from the beginning of the frame to the end of the CRC field, the transmit state machine sends a 32-bit jam pattern of 55555555_H on the MII to inform all other stations that a collision has occurred. If the collision is seen during the preamble transmission phase, the transmit state machine completes the transmission of preamble and SFD and then sends the jam pattern.

If the collision occurs after the collision window and before the end of the FCS field (or the end of Burst if the Frame Burst mode is enabled), the transmit state machine sends a 32-bit jam pattern and sets the late collision bit in the transmit frame status.

The TPE module maintains a jabber timer to cut off the transmission of Ethernet frames if the TFC module transfers more than 2048 (default) bytes. The time-out is changed to 10240 bytes when the Jumbo frame is enabled.

The Transmit state machine uses the deferral mechanism for the flow control (Back Pressure) in Half-Duplex mode. When the Application requests to stop receiving frames, the Transmit state machine sends a JAM pattern of 32 bytes whenever it senses a reception of a frame, provided the transmit flow control is enabled. This will result in a collision and the remote station will back off. The Application requests the flow control by setting **ETH0_FLOW_CONTROL.FCA_BPA** bit. If the application requests a frame to be transmitted, then it will be scheduled and transmitted even when the backpressure is activated. Note that if the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur) then the remote stations will abort their transmissions due to excessive collisions.

If IEEE 1588 time stamping is enabled for the transmit frame, this block takes a snapshot of the system time when the SFD is put onto the transmit MII bus. The system time source is either an external input or internally generated, according to the configuration selected.

Transmit Scheduler Module

The Transmit Scheduler (STX) module is responsible for scheduling the frame transmission on the MII. The two major functions of this module are to maintain the inter-frame gap between two transmitted frames and to follow the Truncated Binary Exponential Back-off algorithm for Half-Duplex mode. This module provides an enable signal to the TPE module after satisfying the IFG and Back-off delays.

The STX module maintains an idle period of the configured inter-frame gap (**ETH0_MAC_CONFIGURATION.IFG** bits) between any two transmitted frames. If frames from the TFC arrive at the TPE module sooner than the configured IFG time, the TPE module waits for the enable signal from the STX module before starting the transmission on the MII. The STX module starts its IFG counter as soon as the carrier signal of the MII goes inactive. At the end of programmed IFG value, the module issues an enable signal to the TPE module in Full-Duplex mode. In Half-Duplex mode and when IFG is configured for 96 bit times, the STX module follows the rule of deference specified in Section 4.2.3.2.1 of the IEEE 802.3 specification. The module resets its IFG counter

if a carrier is detected during the first two-thirds (64-bit times for all IFG values) of the IFG interval. If the carrier is detected during the final one third of the IFG interval, the STX module continues the IFG count and enables the transmitter after the IFG interval.

The STX module implements the Truncated Binary Exponential Back-off algorithm when it operates in Half-Duplex mode.

Transmit CRC Generator Module

The Transmit CRC Generator (CTX) module interfaces with the TFC module to generate CRC for the FCS field of the Ethernet frame. The TFC module sends the frame data and any necessary padding to the CTX module through an 8-bit interface.

This module calculates the 32-bit CRC for the FCS field of the Ethernet frame. The encoding is defined by the following generating polynomial.

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The module gets the Ethernet frame's byte data from the TFC module (DA + SA + LT + DATA + PAD) qualified with a Data Valid signal. The TFC also indicates to the CTX when to reset the previously calculated CRC and to start the new CRC calculation for the coming frame. The TFC module issues the start command before sending the new frame data for calculation. The calculated CRC is valid on the next clock after the data is received.

Transmit Flow Control Module

The Transmit Flow Control (FTX) module generates Pause frames and transmit them to the TFC module as necessary, in Full-Duplex mode. The TFC module receives the Pause frame from the FTX module, appends the calculated CRC, and sends the frame to the TPE module. Pause frame generation can be initiated in two ways. The Application can request the FTX module to send a Pause frame either by setting the **ETH0_FLOW_CONTROL.FCB** bit or in response to the receive FIFO full conditions (packet buffer).

If the Application has requested the flow control by setting the **FLOW_CONTROL.FCB** bit of, the FTX module will generate and transmit a single Pause frame to the TFC module. The value of the Pause Time in the generated frame contains the programmed Pause Time value in the **FLOW_CONTROL** Register. To extend the pause or end the pause prior to the time specified in the previously transmitted Pause frame, the application must request another Pause frame transmission after programming the Pause Time register with appropriate value.

If the Application has requested the flow control by asserting the **mti_flowctrl_i** signal, the FTX module will generate and transmit a Pause frame to the TFC module. The value of the Pause Time in the generated frame contains the programmed Pause Time value in the **FLOW_CONTROL** Register. The FTX module monitors the **MTI Flow Control Signal**. If it remains asserted at a configurable number of slot-times (**FLOW_CONTROL.PLT**

bits) before this Pause-time runs-out, a second Pause frame will be transmitted to the TFC module. The process will be repeated as long as the MTI flow control signal remains asserted.

If the MTI flow control signal goes inactive prior to the sampling time, the FTX module will transmit a Pause frame with zero Pause Time to indicate to the remote end that the receive buffer is ready to receive new data frames.

14.2.1.2 MAC Transmit Interface Protocol

The MAC Transmit Interface (MTI) connects the application with the MTL in the ETH to provide the Ethernet data for transmission.

The application initiates the Ethernet frame transmission by writing the first data of the frame to the ETH, provided the ETH is ready to accept data . The Application can push-in data as long as the ETH core is ready to accept it.

If the frame transmission is not successful (due to underflow, collision, jabber timeout, excessive deferral events), the ETH core will assert the transmit status even before the EOF is received. The Application will have to take the appropriate action as per the status. The ETH will drop all further data input to it until the next SOF.

14.2.1.3 Reception

A receive operation is initiated when the ETH detects an SFD on the MII. The core strips the preamble and SFD before proceeding to process the frame. The header fields are checked for the filtering and the FCS field used to verify the CRC for the frame. The received frame is stored in a shallow buffer until the address filtering is performed. The frame is dropped in the core if it fails the address filter.

The following are the functional blocks in the Receive path of the ETH core.

- Receive Protocol Engine Module (RPE)
- Receive CRC Module (CRX)
- Receive Frame Controller Module (RFC)
- Receive Flow Control Module (FRX)
- Receive IP Checksum checker (IPC)
- Receive Bus Interface Unit Module (RBU)
- Address Filtering Module (AFM)

Receive Protocol Engine Module

The RPE consists of the receive state machine which strips the preamble SFD. Once the external PHY detects ethernet traffic, the RPE's receive state machine begins hunting for the SFD field from the receive modifier logic. Until then, the state machine drops the receiving preambles. Once the SFD is detected, the state machine begins sending the data of the Ethernet frame to the RFC module, beginning with the first byte following the SFD (destination address).

If IEEE 1588 time stamping is enabled, the RPE takes a snapshot of the system time when any frame's SFD is detected on the MII. Unless the MAC filters out and drops the frame, this time stamp is passed on to the application.

In MII mode, the RPE converts the received nibble data into bytes, then forwards the valid frame data to the RFC module

The receive state machine of the RPE module decodes the Length/Type field of the receiving Ethernet frame. If the Length/Type field is less than 600 (hex) and if the MAC is programmed for the auto crc/pad stripping option, the state machine sends the data of the frame up to the count specified in the Length/Type field, then starts dropping bytes (including the FCS field). The state machine of the RPE module decodes the Length/Type field and checks for the Length interpretation.

If the Length/Type field is greater than or equal to 600 (hex), the RPE module will send all received Ethernet frame data to the RFC module, irrespective of the value on the programmed auto-CRC strip option.

As a default, the ETH is programmed for watchdog timer to be enabled, that is, frames above 2.048 (10.240 if Jumbo Frame is enabled) bytes (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module. This feature can be disabled by programming the [ETH0_MAC_CONFIGURATION.WD](#) bit. However even if the watchdog timer is disabled, frames greater than 16 KB in size are cut off and a watchdog time-out status is given.

The ETH supports loopback of transmitted frames onto its receiver. As a default, the ETH loopback function is disabled, but this feature can be enabled by programming the ETH Configuration register, Loopback bit. The transmit and receive clocks can have an asynchronous timing relationship, so an asynchronous FIFO is used to make the loopback path of the PHY transmit path connected onto the receive path. The asynchronous FIFO is 6 bits wide to accommodate the PHY transmit, receive and enable signals. The FIFO is nine words deep and free-running to write on the write clock and read on every read clock.

The write and read pointers gets re-initialized to have an offset of 4 at the start of each frame read out of the FIFO. This helps to avoid overflow/underflow during the transfer of a frame, and ensures that the overflow/underflow occurs only during the IFG period between the frames. Please note that the FIFO depth of nine is sufficient to prevent data corruption for frame sizes up to 9.022 bytes with a difference of 200 ppm between the MII Transmit and Receive clock frequencies. Hence, bigger frames should not be looped back, as they may get corrupted in this loopback FIFO.

At the end of every received frame, the RPE module generates received frame status and sends it to the RFC module. Control, missed frame, and filter fail status are added to the receive status in the RFC module.

Receive CRC Module

The Receive CRC (CRX) interfaces to the RPE module to check for any CRC error in the receiving frame.

This module calculates the 32-bit CRC for the received frame that includes the Destination address field through the FCS field. The encoding is defined by the following generating polynomial.

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The module gets the data from the RPE module (DA+SA+LT+DATA+PAD+FCS). The RPE module also sends a control signal that indicates the validity of the data. Irrespective of the auto pad/CRC strip, the CRX module receives the entire frame to compute the CRC check for received frame. As a note on the auto pad/CRC strip settings, the entire frame is not transferred between the RPE and RFC 8-bit interface.

Receive Checksum Offload Engine

The Receive Checksum Offload engine can detect both IPv4 and IPv6 frames in the received ethernet packets. Once an IP frame is detected it is processed for data integrity.

The Receive Checksum Offload engine is enabled by setting the **ETH0_MAC_CONFIGURATION**.IPC bit. The ETH receiver identifies IPv4 or IPv6 frames by checking for value 0800_H or 86DD_H, respectively, in the received Ethernet frames' Type field. This identification applies to VLAN-tagged frames as well.

The Receive Checksum Offload engine calculates IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received frame does not have enough bytes, as indicated by the IPv4 header's Length field (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

This engine also identifies a TCP, UDP or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP/UDP/ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not tally to the expected payload length given in the IP header.

As mentioned in **TCP/UDP/ICMP Checksum Engine**, this engine bypasses the payload of fragmented IP datagrams, IP datagrams with security features, IPv6 routing headers, and payloads other than TCP, UDP or ICMP.

In this configuration, the core does not append any payload checksum bytes to the received Ethernet frames.

Receive Frame Controller Module

The Receive Frame Controller (RFC) receives the Ethernet frame data and status from the RPE module. The RFC module consists of a FIFO of parameterized depth (default set to 4 deep and 37 bits wide) and two state machines for writing and reading the FIFO. The FIFO holds the received Ethernet frame data and byte enables, along with a control bit to indicate the last data. The state machines manage the FIFO and provide a frame buffering for the receiving Ethernet frame from the RPE module. The main functions of the RFC module are:

- Data path conversion, which converts the 8-bit data to 32-bit data to the RBU module.
- Frame filtering
- Attaching the calculated IP Checksum input from IPC.
- Update the Receive Status and forward to RBU.

If the **ETH0_MAC_FRAME_FILTER.RA** bit is set, the RFC module initiates the data transfer to the RBU module as soon as 4 bytes of Ethernet data are received from the RPE module. At the end of the data transfer, the RFC module sends out the received frame status that includes the frame filter bits (RDES0.SAF SA Filterfail and RDES0.AFM DA Filterfail) and status from the RFC module. These bits are generated based on the filter-fail signals from the AFM module. This status bit indicates to the Application whether the received frame has passed the filter controls (both address filter and Frame Filter controls from CSR). The RFC module will not drop any frame on its own in this mode.

If the **MAC_FRAME_FILTER.RA** bit is reset, the RFC module performs frame filtering based on the destination/source address (the Application still needs to perform another level of filtering if it decides not to receive any bad frames like runt, CRC error frames, etc. The RFC module waits to receive the first 14 bytes of received data (type field) from the RPE module. Until then, the module will not initiate any transfers to the RBU module. After receiving the destination/source address bytes, the RFC checks the filter-fail signal from the AFM module for an address match. On detecting a filter-fail from AFB, the frame is dropped at the RFC module and not transferred to the Application.

On a delayed filter response from the AFM (this can only occur if you change the AFM logic), the RFC module waits until the FIFO is full, and then proceeds with the frame transfer to the RBU module. However, it will still take the delayed response from the AFM module and if it is a (DA/SA) filter failure, then it will drop the rest of the frame and send the Rx Status Word (with zero frame-length, CRC Error and Runt Error bits set) immediately indicating the filter-fail. If there is no response from the AFM until the end of frame is transmitted, the filter fail status in the Rx Status Word is updated accordingly.

When the PMT module is configured for power-down mode, all received frames are dropped by this block, and are not forwarded to the application.

Receive Flow Control Module

The Receive Flow Controller (FRX) detects the receiving Pause frame and pauses the frame transmission for the delay specified within the received Pause frame. The FRX module is enabled only in Full-Duplex mode. The Pause frame detection function can be enabled or disabled with the **ETH0_FLOW_CONTROL**.RFE bit.

Once the receive flow control is enabled, the FRX module begins monitoring the received frame destination address for any match with the multicast address of the control frame (0180C200001_H). If a match is detected, the FRX module indicates to the RFC module, that the destination address of the received frame matches the reserved control frame destination address. The RFC module then decides whether or not to transfer the received control frame to the Application, based on the **ETH0_MAC_FRAME_FILTER**.PCF bit .

The FRX module also decodes the Type, Op-code, and Pause Timer field of the receiving control frame. At the end of received frame, the FRX module gets the received frame status from RPE. If the byte count of the status indicates 64 bytes, and if there is no CRC error, the FRX module requests the MAC transmitter to pause the transmission of any data frame for the duration of the decoded Pause Time value, multiplied by the slot time (64 byte times). Meanwhile, if another Pause frame is detected with a zero Pause Time value, the FRX module resets the Pause Time and gives another pause request to the Transmitter. If the received control frame matches neither the Type field (8808_H), Opcode (00001_H), nor byte length (64 bytes), or if there is a CRC error, the FRX module does not generate a Pause request to Transmitter.

In the case of a pause frame with a multicast destination address, the RFC filters the frame based on the address match from the FRX module. For a pause frame with a unicast destination address, the filtering in the FRX module depends on whether the DA matched the contents of the MAC Address Register 0 and the **ETH0_FLOW_CONTROL**.UP bit is set (detecting a pause frame even with a unicast destination address). The **MAC_FRAME_FILTER**.PCF register bits control the filtering for control frames in addition to the Address filter module.

Receive Bus Interface Unit Module

The Receive Bus Interface Unit (RBU) converts the 32-bit data received from the RFC module into a 32-bit FIFO protocol on the Application side. The RBU module interfaces with the Application through the MAC receive interface (MRI).

If IEEE 1588 time stamping is enabled, the RBU also outputs the time stamp captured from the received frame.

Address Filtering Module

The Address Filtering (AFM) module performs the destination and source address checking function on all received frames and reports the address filtering status to the

RFC module. The address checking is based on different parameters (Frame Filter register) chosen by the Application. These parameters are inputs to the AFM module as control signals, and the AFM module reports the status of the address filtering based on the combination of these inputs. The AFM module does not filter the receive frames by itself, but reports the status of the address filtering (whether to drop the frame or not) to the RFC module. The AFM module also reports whether the receiving frame is a multicast frame or a broadcast frame, as well as the address filter status.

The AFM module probes the 8-bit receive data path between the RPE module and the RFC module and checks the destination and source address field of each incoming packet. In MII mode the module takes 14/26 clocks (from the start of frame) to compare the destination/ source address of the receiving frame. The AFM module gets the station's physical (MAC) address and the Multicast Hash table from CSR module for address checking. The CSR module provides the Frame Filter register parameters to AFM.

Unicast Destination Address Filter

The AFM supports up to 4 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of Frame Filter register is reset), the AFM compares all 48 bits of the received unicast address with the programmed MAC address for any match. Default MacAddr0 is always enabled, other addresses MacAddr1–MacAddr3 are selected with an individual enable bit. Each byte of these other addresses (MacAddr1–MacAddr3) can be masked during comparison with the corresponding received DA byte by setting the corresponding Mask Byte Control bit in the register. This helps group address filtering for the DA.

In Hash filtering mode (When HUC bit is set), the AFM performs imperfect filtering for unicast addresses using a 64-bit Hash table. For hash filtering, the AFM uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 000000 selects Bit 0 of the selected register, and a value of 111111 selects Bit 63 of the Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast frame is said to have passed the Hash filter; otherwise, the frame has failed the Hash filter.

Multicast Destination Address Filter

The ETH can be programmed to pass all multicast frames by setting the **ETH0_MAC_FRAME_FILTER.PM** bit. If the **MAC_FRAME_FILTER.PM** bit is reset, the AFM performs the filtering for multicast addresses based on the **MAC_FRAME_FILTER.HMC** bit. In Perfect Filtering mode, the multicast address is compared with the programmed MAC Destination Address registers (1–31). Group address filtering is also supported.

In Hash filtering mode, the AFM performs imperfect filtering using a 64-bit Hash table. For hash filtering, the AFM uses the upper 6 bits CRC of the received multicast address

to index the content of the Hash table. A value of 000000_B selects Bit 0 of the selected register and a value of 111111_B selects Bit 63 of the Hash Table register.

If the corresponding bit is set to 1, then the multicast frame is said to have passed the Hash filter; otherwise, the frame has failed the Hash filter.

Hash or Perfect Address Filter

The DA filter can be configured to pass a frame when its DA matches either the Hash filter or the Perfect filter by setting the MAC_FRAME_FILTER.HPF bit and setting the corresponding [ETH0_MAC_FRAME_FILTER.HUC](#) or [MAC_FRAME_FILTER.HMC](#) bits. This configuration applies to both unicast and multicast frames. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to the received frame.

Broadcast Address Filter

The AFM doesn't filter any broadcast frames in the default mode. However, if the ETH is programmed to reject all broadcast frames by setting the MAC_FRAME_FILTER.DBF bit, the DAF module asserts the Filter fail signal to RFC, whenever a broadcast frame is received. This will tell the RFC module to drop the frame.

Unicast Source Address Filter

The ETH can also perform a perfect filtering based on the source address field of the received frames. By default, the AFM compares the SA field with the values programmed in the SA registers. The MAC Address registers [1:3] can be configured to contain SA instead of DA for comparison, by setting Bit 30 of the corresponding Register. Group filtering with SA is also supported. The frames that fail the SA Filter are dropped by the ETH if the MAC_FRAME_FILTER.SAF bit of Frame Filter register is set.

When MAC_FRAME_FILTER.SAF bit is set, the result of SA Filter and DA filter is AND'ed to decide whether the frame needs to be forwarded. This means that either of the filter fail result will drop the frame and both filters have to pass in-order to forward the frame to the application.

Inverse Filtering Operation

For both Destination and Source address filtering, there is an option to invert the filter-match result at the final output. These are controlled by the DAIF and SAIF bits of the Frame Filter register respectively. The MAC_FRAME_FILTER.DAIF bit is applicable for both Unicast and Multicast DA frames. The result of the unicast/multicast destination address filter is inverted in this mode. Similarly, when the MAC_FRAME_FILTER.SAIF bit is set, the result of unicast SA filter is reversed.

[Table 14-2](#) and [Table 14-3](#) summarize the Destination and Source Address filtering based on the type of frames received.

Table 14-2 Destination Address Filtering Table

Frame Type	PR	HPF	HUC	DAIF	HMC	PM	DB	DA Filter Operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all frames.
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match.
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match.
	0	0	1	0	X	X	X	Pass on Hash filter match.
	0	0	1	1	X	X	X	Fail on Hash filter match.
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match.
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match.
Multicast	1	X	X	X	X	X	X	Pass all frames.
	X	X	X	X	X	1	X	Pass all frames.
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.
	0	0	X	0	1	0	X	Pass on Hash filter match and drop PAUSE control frames if PCF = 0x.
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.
	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.

Table 14-2 Destination Address Filtering Table (cont'd)

Frame Type	PR	HPF	HUC	DAIF	HMC	PM	DB	DA Filter Operation
	0	0	X	1	1	0	X	Fail on Hash filter match and drop PAUSE control frames if PCF = 0x.
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.

Table 14-3 Source Address Filtering Table

Frame Type	PR	SAIF	SAF	SA Filter Operation
Unicast	1	X	X	Pass all frames.
	0	0	0	Pass status on Perfect/Group filter match but do not drop frames that fail.
	0	1	0	Fail status on Perfect/Group filter match but do not drop frame.
	0	0	1	Pass on Perfect/Group filter match and drop frames that fail.
	0	1	1	Fail on Perfect/Group filter match and drop frames that fail.

14.2.2 MAC Transaction Layer (MTL)

The MAC Transaction Layer provides FIFO memory to buffer and regulate the frames between the application system memory and the ETH core. It also enables the data to be transferred between the application clock domain and the ETH clock domains. The MTL layer has 2 data paths, namely the Transmit path and the Receive Path. The data path for both directions is 32-bit wide and operates with a simple FIFO protocol.

The ETH-MTL communicates with the application side with the Application Transmit Interface (ATI), Application Receive Interface (ARI), and the MAC Control Interface (MCI).

14.2.2.1 Transmit Path

DMA controls all transactions for the transmit path through the ATI. Ethernet frames read from the system memory is pushed into the FIFO by the DMA. The frame is then popped

out and transferred to the ETH core when triggered. When the end-of-frame is transferred, the status of the transmission is taken from the ETH core and transferred back to the DMA.

The Transmit FIFO has a depth of 2K bytes. A 2 FIFO-fill level is indicated to the DMA so that it can initiate a data fetch in required bursts from the system memory, using the Bus interface. The data from the Bus Master interface is pushed into the FIFO with the appropriate byte lanes qualified by the DMA. The DMA also indicates the start-of-frame (SOF) and end-of-frame (EOF) transfers along with a few signals controlling the pad-insertion/CRC generation for that frame in the ETH core.

Per-frame control bits, such as Automatic Pad/CRC Stripping disable, time stamp capture, and so forth are taken as control inputs on the ATI, stored in a separate register FIFO, and passed on to the core transmitter when the corresponding frame data is read from the Transmit FIFO.

There are two modes of operation for popping data towards the ETH core. In Threshold mode, as soon as the number of bytes in the FIFO crosses the configured threshold level (or when the end-of-frame is written before the threshold is crossed), the data is ready to be popped out and forwarded to the ETH core. The threshold level is configured using the TTC bits of DMA **ETH0_BUS_MODE** Register. In store-and-forward mode, the MTL pops the frame towards the ETH core only when one or more of the following conditions are true:

- When a complete frame is stored in the FIFO
- When the TX FIFO becomes almost full
- When the ATI watermark becomes low. The watermark becomes low when the requested FIFO does not have space to accommodate the requested burst-length on the ATI.

Therefore, the MTL never stops in the store-and-forward mode even if the Ethernet frame length is bigger than the Tx FIFO depth.

The application can flush the Transmit FIFO of all contents by setting the **ETH0_OPERATION_MODE**.FTF bit. This bit is self-clearing and initializes the FIFO pointers to the default state. If the FTF bit is set during a frame transfer from the MTL to the ETH core, then the MTL stops further transfer as the FIFO is considered to be empty. Hence an underflow event occurs at the ETH transmitter and the corresponding Status word is forwarded to the DMA.

Initialization through **Transmit Status Word** detail initialization and transmit operations for the MTL Layer.

Initialization

Upon reset, the MTL is ready to manage the flow of data to and from the DMA and the ETH .

There are no requirements for enabling the MTL. However, the ETH block and the DMA controller must be enabled individually through their respective CSRs.

Single-Packet Transmit Operation

During a transmit operation, the MTL block is slaved to the DMA controller. The general sequence of events for a transmit operation is as follows.

1. If the system has data to be transferred, the DMA controller, if enabled, fetches data from the XMC4300 RAM through the Bus Master interface and starts forwarding it to the MTL. The MTL pushes the data received from the DMA into the FIFO. It continues to receive the data until the end-of frame of the frame is transferred.
2. The data is taken out of the FIFO and sent to the MAC by the FIFO controller engine. When the threshold level is crossed or a full packet of data is received into the FIFO, the MTL pops out the frame data and drives them to the ETH core. The engine continues to transfer data from the FIFO until a complete packet has been transferred to the MAC. Upon completion of the frame, the MTL receives the Status from the ETH and then notifies the DMA controller

Transmit Operation—Two Packets in the Buffer

1. Because the DMA must update the descriptor status before releasing it to the CPU, there can be at the most two frames inside a transmit FIFO. The second frame will be fetched by the DMA and put into the FIFO only if the OSF (Operate on Second Frame bit is set). If this bit is not set, the next frame will be fetched from the memory only after the MAC has completely processed the frame and the DMA has released the descriptors.
2. If the OSF bit is set, the DMA starts fetching the second frame immediately after completing the transfer of the first frame to the FIFO. It does not wait for the status to be updated. The MTL, in the meantime, receives the second frame into the FIFO while transmitting the first frame. As soon as the first frame has been transferred and the status is received from the MAC, the MTL pushes it to the DMA. If the DMA has already completed sending the second packet to the MTL, it must wait for the status of the first packet before proceeding to the next frame.

Transmit Operation—Multiple Packets in Buffer

In ETH -MTL configuration, the transmit FIFO can be configured to accept more than 2 packets at a time. This option limits the number of status words that can be stored in the MTL before it is transferred to the DMA/CPU. By default, this number is limited to 2 but can be configured for 4 or 8 as well. Once the MTL FIFO accepts the number of frames equal to the status FIFO depth, it will stop accepting further frames unless the transmit Status that is given out and accepted by the CPU/DMA thus freeing up the space in this small FIFO.

Retransmission During Collision

While a frame is being transferred from the MTL to the ETH, a collision event occurs on the ETH line interface in Half-Duplex mode. The ETH then indicates a retry attempt to the MTL by giving the status even before the end-of-frame is transferred from MTL. Then the MTL will enable the retransmission by popping out the frame again from the FIFO.

After more than 96 bytes are popped towards the ETH core, the FIFO controller frees up that space and makes it available to the DMA to push in more data. This means that the retransmission is not possible after this threshold is crossed or when the ETH core indicates a late-collision event.

Transmit FIFO Flush Operation

The ETH provides a control signal to the software to flush the Transmit FIFO in the MTL layer through the use of the **ETH0_OPERATION_MODE.FTF** bit. The Flush operation is immediate and the MTL clears the Tx FIFO and the corresponding pointers to the initial state even if it is in the middle of transferring a frame to the ETH Core. The data which is already accepted by the MAC transmitter will not be flushed. It will be scheduled for transmission and will result in underflow as Tx FIFO does not complete the transfer of rest of the frame. As in all underflow conditions, a runt frame will be transmitted and observed on the line. The status of such a frame will be marked with both Underflow and Frame Flush events (TDES0_{RAM} bits 13 and 1).

The MTL layer also stops accepting any data from the application (DMA) during the Flush operation. It will generate and transfer Transmit Status Words to the application for the number of frames that is flushed inside the MTL (including partial frames). Frames that are completely flushed in the MTL will have the Frame Flush Status bit (TDES0 13_{RAM}) set. The MTL completes the Flush operation when the application (DMA) accepts all of the Status Words for the frames that were flushed, and then clears the Transmit FIFO Flush control register bit. At this point, the MTL starts accepting new frames from the application (DMA).

Transmit Status Word

At the end of transfer of the Ethernet frame to the ETH core and after the core completes the transmission of the frame, the MTL outputs the transmit status to the application. The detailed description of the Transmit Status is the same as for bits [23:0] of TDES0_{RAM} given in [Table 14-9](#).

If IEEE 1588 time stamping is enabled, the MTL returns specific frame's 64-bit time stamp, along with the ATI's transmit status.

Transmit Checksum Offload Engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. Because the most widespread

use of Ethernet is to encapsulate TCP and UDP over IP datagrams, the ETH has an Checksum Offload Engine (COE) to support checksum calculation and insertion in the transmit path, and error detection in the receive path. This section explains the operation of the Checksum Offload Engine for transmitted frames.

*Note: The checksum for TCP, UDP, or ICMP is calculated over a complete frame, then inserted into its corresponding header field. Due to this requirement, this function is enabled only when the Transmit FIFO is configured for Store-and-Forward mode (that is, when the **ETH0_OPERATION_MODE.TSF** bit is set .). If the core is configured for Threshold (cut-through) mode, the Transmit COE is bypassed.*

Note: You must make sure that the Transmit FIFO is deep enough to store a complete frame before that frame is transferred to the ETH Core transmitter. The reason being that when space is not available to accept the programmed burst length of the data, then the MTL TxFIFO starts reading to avoid dead-lock. Once reading starts, then checksum insertion engine fails and consequently all succeeding frames may get corrupted due to improper recovery. Therefore, you must enable the checksum insertion only in the frames that are less than the following number of bytes in size (even in the store-and-forward mode):

FIFO Depth – PBL – 3 FIFO Locations

*The **ETH0_BUS_MODE.PBL** is the programmed burst-length.*

This checksum engine can be controlled for each frame by setting the CIC bits (Bits 28:27 of TDES1_{RAM}, described in **Transmit Descriptor 1**).

Note: See IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.

IP Header Checksum Engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Ethernet frame's Type field has the value 0800_H and the IP datagram's Version field has the value 4_H. The input frame's checksum field is ignored during calculation and replaced with the calculated value.

IPv6 headers do not have a checksum field; thus, the COE does not modify IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status bit in the Transmit status (Bit 16 in **Table 14-9**). This status bit is set whenever the values of the Ethernet Type field and the IP header's Version field are not consistent, or when the Ethernet frame does not have enough data, as indicated by the IP header Length field.

In other words, this bit is set when an IP header error is asserted under the following circumstances:

For IPv4 datagrams

- The received Ethernet type is 0800_H , but the IP header's Version field does not equal 4_H
- The IPv4 Header Length field indicates a value less than 5_H (20 bytes)
- The total frame length is less than the value given in the IPv4 Header Length field

For IPv6 datagrams

- The Ethernet type is $86DD_H$ but the IP header Version field does not equal 6_H
- The frame ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

Even when the COE detects such an IP header error, it inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

TCP/UDP/ICMP Checksum Engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP.

Note: For non-TCP, -UDP, or -ICMP/ICMPv6 payloads, this checksum engine is bypassed and nothing further is modified in the frame.

Note: Fragmented IP frames (IPv4 or IPv6), IP frames with security features (such as an authentication header or encapsulated security payload), and IPv6 frames with routing headers are not processed by this engine, and therefore must be bypassed. In other words, payload checksum insertion must not be enabled for such frames.

The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. This engine can work in the following two modes:

- In the first mode, the TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the input frame's Checksum field. This engine includes the Checksum field in the checksum calculation, then replaces the Checksum field with the final calculated checksum.
- In the second mode, the engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

Note: For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be $16'h0000$ in both modes, because pseudo-headers are not defined for such packets. If it does not equal $16'h0000$, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (Bit 12 in [Table 14-9](#)). This engine sets the Payload Checksum

Error status bit when it detects that the frame has been forwarded to the MAC Transmitter engine in Store-and-Forward mode without the end-of-frame being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

14.2.2.2 Receive Path

This module receives the frames given out by the ETH core and pushes them into the Rx FIFO. The status (fill level) of this FIFO is indicated to the DMA once it crosses the configured Receive threshold (**ETH0_OPERATION_MODE.RTC** bits). The MTL also indicates the FIFO fill level so that the DMA can initiate pre-configured burst transfers towards the Bus interface.

Receive Operation through **Receive Status Word** detail receive operations for the MTL Layer.

Receive Operation

During an Rx operation, the MTL is slaved to the ETH. The general sequence of Receive operation events is as follows:

1. When the ETH receives a frame, it pushes in data along with byte enables. The ETH also indicates the SOF and EOF. The MTL accepts the data and pushes it into the Rx FIFO. After the EOF is transferred, the ETH drives the status word, which is also pushed into the same Rx FIFO by the MTL.
2. When IEEE 1588 time stamping is enabled and the 64-bit time stamp is available along with the receive status, it is appended to the frame received from the ETH and is pushed into the Rx FIFO before the corresponding receive status word is written. Thus, two additional locations per frame are taken for storing the time stamp in the Rx FIFO.
3. The MTL_RX engine takes the data out of the FIFO and sends it to the DMA. In the default Cut-Through mode, when 64 bytes (configured with the **ETH0_OPERATION_MODE.RTC** bits or a full packet of data are received into the FIFO, the MTL_RX engine pops out the data and indicates its availability to the DMA. Once the DMA initiates the transfer to the Bus interface, the MTL_RX engine continues to transfer data from the FIFO until a complete packet has been transferred. Upon completion of the EOF frame transfer, the MTL pops out the status word and sends it to the DMA controller.
4. In Rx FIFO Store-and-Forward mode (configured by the Operation Mode.RSF bit), a frame is read out only after being written completely into the Receive FIFO. In this mode, all error frames are dropped (if the core is configured to do so) such that only

valid frames are read out and forwarded to the application. In Cut-Through mode, some error frames are not dropped, because the error status is received at the end-of-frame, by which time the start of that frame has already been read out of the FIFO.

Note: The time-stamp transfer takes two clock cycles and the lower 32-bit of the time-stamp is given out first. The status also may be extended to two cycles when Advanced Time-stamp feature is enabled.

Receive Operation Multiframe Handling

Since the status is available immediately following the data, the MTL is capable of storing any number of frames into the FIFO, as long as it is not full.

Error Handling

If the MTL Rx FIFO is full before it receives the EOF data from the ETH, an overflow is declared, the whole frame (including the status word) is dropped, and the overflow counter in the DMA ([ETH0_MISSED_FRAME_AND_BUFFER_OVERFLOW_COUNTER](#) Register) is incremented. This is true even if the Forward Error Frame ([ETH0_OPERATION_MODE.FEF](#) bit) is set. If the start address of such a frame has already been transferred to the Read Controller, the rest of the frame is dropped and a dummy EOF is written to the FIFO along with the status word. The status will indicate a partial frame due to overflow. In such frames, the Frame Length field is invalid.

The MTL Rx Control logic can filter error and undersized frames, if enabled (using the Operation Mode.FEF and Operation Mode.FUF bits). If the start address of such a frame has already been transferred to the Rx FIFO Read Controller, that frame is not filtered. The start address of the frame is transferred to the Read Controller after the frame crosses the receive threshold (set by the Operation Mode.RTC bits).

If the MTL Receive FIFO is configured to operate in Store-and-Forward mode, all error frames can be filtered and dropped.

Receive Status Word

At the end of the transfer of the Ethernet frame to the XMC4300 RAM, the MTL outputs the receive status to the Application. The detailed description of the receive status is the same as for Bits[31:0] of RDES_{0RAM}, given in [Table 14-4](#), except that Bits 31, 14, 9, and 8 are reserved and have a reset of 0 by default. When the status of a partial frame due to overflow is given out, the Frame Length field in the status word is not valid.

Note: When Advanced Time Stamp feature is enabled, the status is composed of two parts - normal (default [31:0]), and extended. The extended status[63:32] gives the information about the received ethernet payload when it is carrying PTP packets or TCP/UDP/ICMP over IP packets. These are transferred over two clock cycles. The detailed description of the receive status is the same as described in RDES0

and RDES4 in **Receive Descriptor**, except that bits 31, 14, 9, and 8 of normal status is reserved and have a reset value of 0_B. When the status of a partial frame due to overflow is given out, the Frame Length field in the status word is not valid.

14.2.3 DMA Controller

The DMA has independent Transmit and Receive engines, and a CSR space. The Transmit Engine transfers data from system memory to the device port (MTL), while the Receive Engine transfers data from the device port to system memory. The controller utilizes descriptors to efficiently move data from source to destination with minimal CPU intervention. The DMA is designed for packet-oriented data transfers such as frames in Ethernet. The controller can be programmed to interrupt the CPU for situations such as Frame Transmit and Receive transfer completion, and other normal/error conditions.

The DMA and the CPU driver communicate through two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

Control and Status registers are described in detail in [Chapter 14.6](#). Descriptors are described in detail in [DMA Descriptors](#).

The DMA descriptors are held in ram. To avoid confusion with the ETH registers the DMA descriptors use the subscript RAM for example RDES0[0]_{RAM}.

The DMA transfers data frames received by the core to the Receive Buffer in the XMC4300 memory, and Transmit data frames from the Transmit Buffer in the XMC4300 memory. Descriptors that reside in the XMC4300 memory act as pointers to these buffers.

There are two descriptor lists; one for reception, and one for transmission. The base address of each list is written into DMA RECEIVE_DESCRIPTOR_LIST_ADDRESS Register and TRANSMIT_DESCRIPTOR_LIST_ADDRESS Register, respectively. A descriptor list is forward linked (either implicitly or explicitly). The last descriptor may point back to the first entry to create a ring structure. Explicit chaining of descriptors is accomplished by setting the second address chained in both Receive and Transmit descriptors (RDES1[24]_{RAM} and TDES1[24]_{RAM}). The descriptor lists resides in the XMC4300 physical memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the XMC4300 physical memory space, and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers contain only data, buffer status is maintained in the descriptor. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA will skip to the next frame buffer when end-of-frame is detected. Data chaining can be enabled or disabled.

The descriptor ring and chain structure is shown in [Figure 14-2](#).

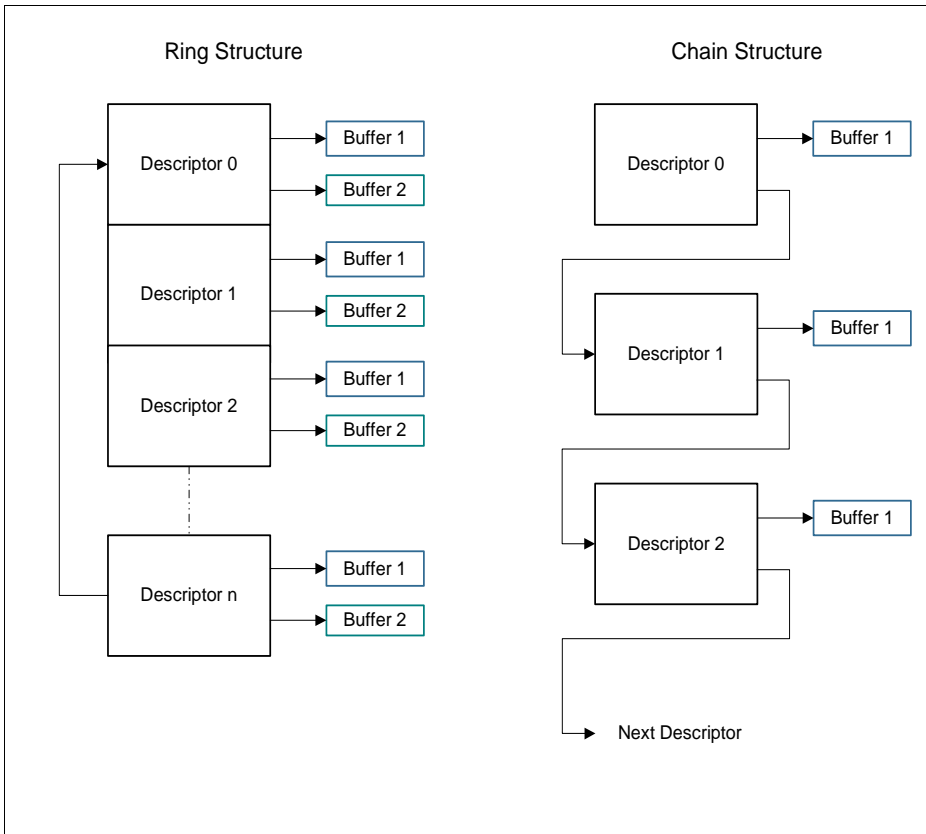


Figure 14-2 Descriptor Ring and Chain Structure

14.2.3.1 Initialization

Initialization for the ETH is as follows.

1. Write to **ETH0_BUS_MODE** Register to set XMC4300 bus access parameters.
2. Write to **ETH0_INTERRUPT_ENABLE** Register to mask unnecessary interrupt causes.
3. The software driver creates the Transmit and Receive descriptor lists. Then it writes to both DMA **ETH0_RECEIVE_DESCRIPTOR_LIST_LIST_ADDRESS** Register and DMA **ETH0_TRANSMIT_DESCRIPTOR_LIST_ADDRESS** Register, providing the DMA with the starting address of each list.

4. Write to ETH Registers **ETH0_TRANSMIT_POLL_DEMAND**, **ETH0_RECEIVE_POLL_DEMAND**, and Receive Descriptor List Address for desired filtering options.
5. Write to **ETH0_MAC_CONFIGURATION** Register to configure and enable the Transmit and Receive operating modes. The **ETH0_MAC_CONFIGURATION.DM** bit is set based on the auto-negotiation result (read from the PHY).
6. Write to **ETH0_OPERATION_MODE.ST** and **ETH0_OPERATION_MODE.SR** bits start transmission and reception.
7. The Transmit and Receive engines enter the Running state and attempt to acquire descriptors from the respective descriptor lists. The Receive and Transmit engines then begin processing Receive and Transmit operations. The Transmit and Receive processes are independent of each other and can be started or stopped separately.

XMC4300 Bus Burst Access

The DMA will attempt to execute fixed-length Burst transfers on the Bus Master interface if configured to do so (**ETH0_BUS_MODE.FB** Register). The maximum Burst length is indicated and limited by the PBL field (Bus Mode.PBL Register). The Receive and Transmit descriptors are always accessed in the maximum possible (limited by PBL or $(16 * 8)/32$) burst-size for the 16-bytes to be read.

The Transmit DMA will initiate a data transfer only when sufficient space to accommodate the configured burst is available in MTL Transmit FIFO or the number of bytes till the end of frame (when it is less than the configured burst-length). The DMA will indicate the start address and the number of transfers required to the Bus Master Interface. When the Bus Interface is configured for fixed-length burst, then it will transfer data using the best combination of INCR4/8 and SINGLE transactions.

The Receive DMA will initiate a data transfer only when sufficient data to accommodate the configured burst is available in MTL Receive FIFO or when the end of frame (when it is less than the configured burst-length) is detected in the Receive FIFO. The DMA will indicate the start address and the number of transfers required to the Bus Master Interface. When the Bus Interface is configured for fixed-length burst, then it will transfer data using the best combination of INCR4/8 and SINGLE transactions. If the end-of frame is reached before the fixed-burst ends on the Bus interface, then dummy transfers are performed in-order to complete the fixed-burst.

When the Bus interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer the Bus initiates is less than or equal to the size of the configured PBL. Thus, all subsequent beats start at an address that is aligned to the configured PBL. The DMA can only align the address for beats up to size (for $PBL >$), because the Bus interface does not support more than INCR8/16.

XMC4300 Data Buffer Alignment

The Transmit and Receive data buffers do not have any restrictions on start address alignment. For example, the start address for the buffers can be aligned to any of the four bytes. However, the DMA always initiates transfers with address aligned to the bus width with dummy data for the byte lanes not required. This typically happens during the transfer of the beginning or end of an Ethernet frame.

Example - Buffer Read

If the Transmit buffer address is $00000FF2_H$, and 15 bytes need to be transferred, then the DMA will read five full words from address $00000FF0_H$, but when transferring data to the MTL Transmit FIFO, the extra bytes (the first two bytes) will be dropped or ignored. Similarly, the last 3 bytes of the last transfer will also be ignored. The DMA always ensures it transfers a full 32-bit data to the MTL Transmit FIFO, unless it is the end-of-frame.

Buffer Size Calculations

The DMA does not update the size fields in the Transmit and Receive descriptors. The DMA updates only the status fields ($RDES_{RAM}$ and $TDES_{RAM}$) of the descriptors. The driver has to perform the size calculations.

The transmit DMA transfers to the ETH the exact number of bytes (indicated by buffer size field of $TDES1_{RAM}$) towards the ETH core. If a descriptor is marked as first (FS bit of $TDES1_{RAM}$ is set), then the DMA marks the first transfer from the buffer as the start of frame. If a descriptor is marked as last (LS bit of $TDES1_{RAM}$), then the DMA marks the last transfer from that data buffer as the end-of frame to the MTL.

The Receive DMA transfers data to a buffer until the buffer is full or the end-of frame is received from the MTL. If a descriptor is not marked as last (LS bit of $RDES0_{RAM}$), then the descriptor's corresponding buffer(s) are full and the amount of valid data in a buffer is accurately indicated by its buffer size field minus the data buffer pointer offset when the FS bit of that descriptor is set. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, then the buffer may not be full (as indicated by the buffer size in $RDES1_{RAM}$). To compute the amount of valid data in this final buffer, the driver must read the frame length (FL bits of $RDES0[29:16]_{RAM}$) and subtract the sum of the buffer sizes of the preceding buffers in this frame. The Receive DMA always transfers the start of next frame with a new descriptor.

Note: Even when the start address of a receive buffer is not aligned to a word boundary, the system should allocate a receive buffer aligned to a word boundary. For example, if the system allocates a 1024-byte (1 KB) receive buffer starting from address 1000_H , the software can program the buffer start address in the Receive descriptor to have a 1002_H offset. The Receive DMA writes the frame to this buffer with dummy data in the first two locations (1000_H and 1001_H). The actual frame is written from location 1002_H . Thus, the actual useful space in this buffer is 1022

bytes, even though the buffer size is programmed as 1024 bytes, due to the start address offset.

DMA Arbiter

The arbiter inside the DMA module performs the arbitration between the Transmit and Receive channel accesses to the Bus Master interface. Two types of arbitrations are possible: round-robin, and fixed-priority.

When round-robin arbitration is selected (**ETH0_BUS_MODE**.DA bit is reset), the arbiter allocates the data bus in the ratio set by the Bus Mode.PR Bits, when both Transmit and Receive DMAs are requesting for access simultaneously. When the DA bit is set, the Receive DMA always gets priority over the Transmit DMA for data access.

14.2.3.2 Transmission

The Transmit DMA engine has two operating modes, default and Operate Second Frame (OSF). Both these modes are described below.

TxDMA Operation: Default (Non-OSF) Mode

The Transmit DMA engine in default mode proceeds as follows:

1. The CPU sets up the transmit descriptor (TDES0_{RAM} - TDES3_{RAM}) and sets the Own bit (TDES0[31]_{RAM}) after setting up the corresponding data buffer(s) with Ethernet Frame data.
2. Once the **ETH0_OPERATION_MODE**.ST bit is set, the DMA enters the Run state.
3. While in the Run state, the DMA polls the Transmit Descriptor list for frames requiring transmission. After polling starts, it continues in either sequential descriptor ring order or chained order. If the DMA detects a descriptor flagged as owned by the CPU, or if an error condition occurs, transmission is suspended and both the Transmit Buffer Unavailable (**ETH0_STATUS**.TU) and Normal Interrupt Summary (STATUS.NIS Register) bits are set. The Transmit Engine proceeds to Step 8.
4. If the acquired descriptor is flagged as owned by DMA (TDES0[31]_{RAM} = 1_B), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
5. The DMA fetches the Transmit data from the XMC4300 memory and transfers the data to the MTL for transmission.
6. If an Ethernet frame is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps Step 2, Step 3 and Step 4 are repeated until the end-of-Ethernet-frame data is transferred to the MTL.
7. When frame transmission is complete, if IEEE 1588 time stamping was enabled for the frame (as indicated in the transmit status) the time-stamp value obtained from MTL is written to the transmit descriptor (TDES2_{RAM} and TDES3_{RAM}) that contains the end-of-frame buffer. The status information is then written to this transmit descriptor (TDES0_{RAM}). Because the Own bit is cleared during this step, the CPU now owns this

descriptor. If time stamping was not enabled for this frame, the DMA does not alter the contents of $TDES2_{RAM}$ and $TDES3_{RAM}$.

8. Transmit Interrupt (**ETH0_STATUS.TI**) is set after completing transmission of a frame that has Interrupt on Completion ($TDES1[31]_{RAM}$) set in its Last Descriptor. The DMA engine then returns to Step 2.
9. In the Suspend state, the DMA tries to re-acquire the descriptor (and thereby return to Step 2) when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared.

The TxDMA transmission flow in default mode is shown in **Figure 14-3**.

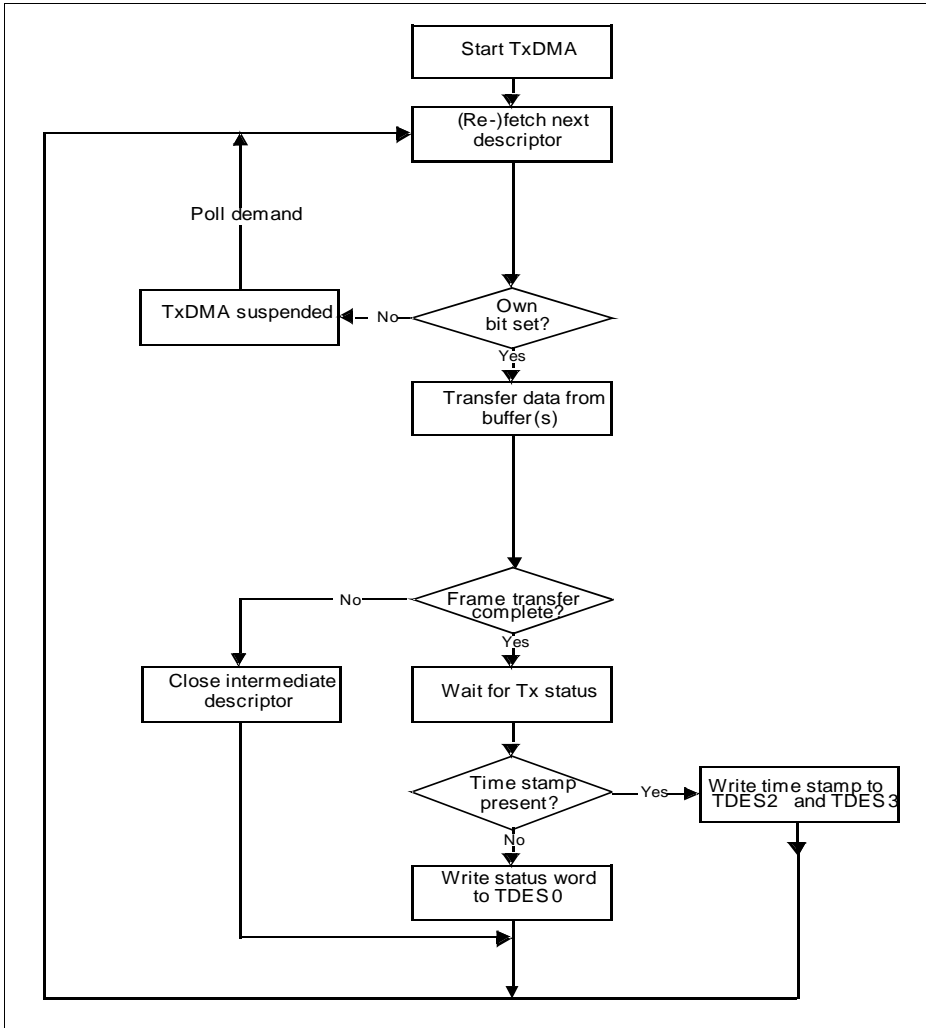


Figure 14-3 TxDMA Operation in Default Mode

TxDMA Operation: OSF Mode

While in the Run state, the transmit process can simultaneously acquire two frames without closing the Status descriptor of the first if **ETH0_OPERATION_MODE.OSF** is set. As the transmit process finishes transferring the first frame, it immediately polls the

Transmit Descriptor list for the second frame. If the second frame is valid, the transmit process transfers this frame before writing the first frame's status information.

In OSF mode, the Run state Transmit DMA operates in the following sequence:

1. The DMA operates as described in Step 1–Step 5 of the TxDMA (default mode).
2. Without closing the previous frame's last descriptor, the DMA fetches the next descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and skips to Step 6.
4. The DMA fetches the Transmit frame from the XMC4300 memory and transfers the frame to the MTL until the End-of-Frame data is transferred, closing the intermediate descriptors if this frame is split across multiple descriptors.
5. The DMA waits for the previous frame's frame transmission status and time stamp. Once the status is available, the DMA writes the time stamp to $TDES2_{RAM}$ and $TDES3_{RAM}$, if such time stamp was captured (as indicated by a status bit). The DMA then writes the status, with a cleared Own bit, to the corresponding $TDES0_{RAM}$, thus closing the descriptor. If time stamping was not enabled for the previous frame, the DMA does not alter the contents of $TDES2_{RAM}$ and $TDES3_{RAM}$.
6. If enabled, the Transmit interrupt is set, the DMA fetches the next descriptor, then proceeds to Step 2 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (Step 6).
7. In Suspend mode, if a pending status and time stamp are received from the MTL, the DMA writes the time stamp (if enabled for the current frame) to $TDES2_{RAM}$ and $TDES3_{RAM}$, then writes the status to the corresponding $TDES0_{RAM}$. It then sets relevant interrupts and returns to Suspend mode.
8. The DMA can exit Suspend mode and enter the Run state (go to Step 1 or Step 2 depending on pending status) only after receiving a Transmit Poll demand (**ETH0_TRANSMIT_POLL_DEMAND** Register).

Note: As the DMA fetches the next descriptor in advance before closing the current descriptor, the descriptor chain should have more than 2 different descriptors for correct and proper operation.

The basic flow is charted in **Figure 14-4**.

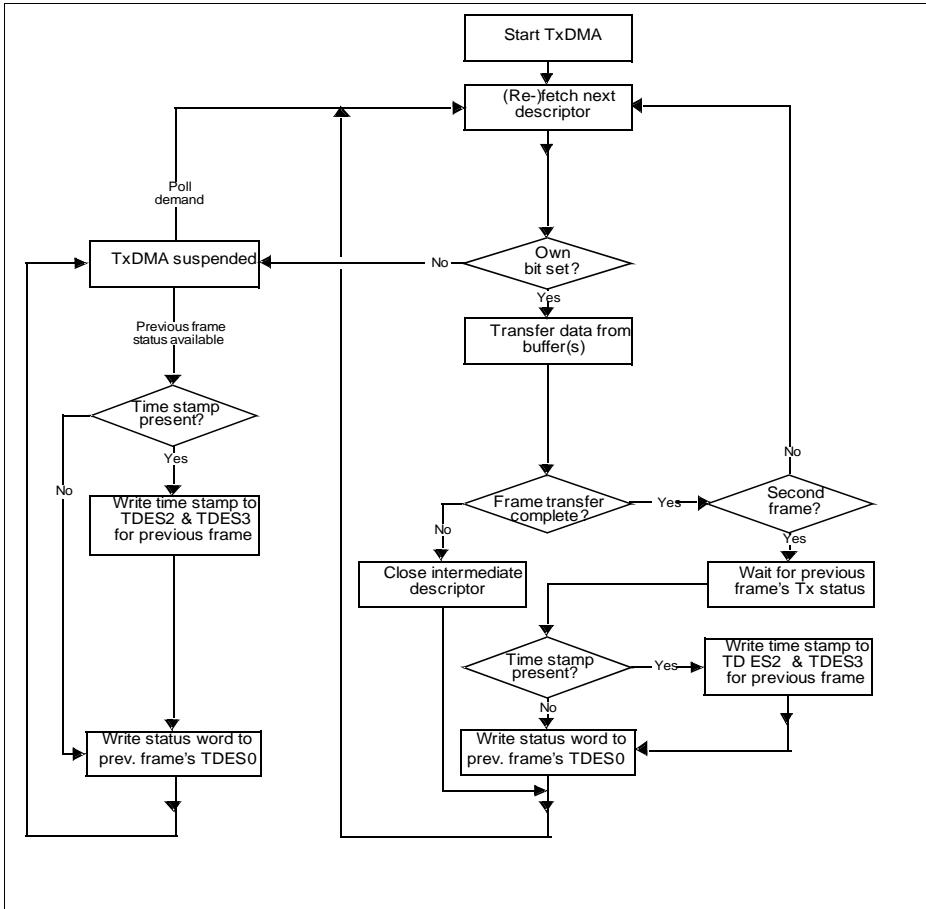


Figure 14-4 TxDMA Operation in OSF Mode

Transmit Frame Processing

The Transmit DMA expects that the data buffers contain complete Ethernet frames, excluding preamble, pad bytes, and FCS fields. The DA, SA, and Type/Len fields must contain valid data. If the Transmit Descriptor indicates that the MAC core must disable CRC or PAD insertion, the buffer must have complete Ethernet frames (excluding preamble), including the CRC bytes.

Frames can be data-chained and can span several buffers. Frames must be delimited by the First Descriptor (TDES1[29]_{RAM}) and the Last Descriptor (TDES1[30]_{RAM}), respectively.

As transmission starts, the First Descriptor must have (TDES1[29]_{RAM}) set. When this occurs, frame data transfers from the XMC4300 RAM buffer to the MTL Transmit FIFO. Concurrently, if the current frame has the Last Descriptor (TDES1[30]_{RAM}) clear, the Transmit Process attempts to acquire the Next Descriptor. The Transmit Process expects this descriptor to have TDES1[29]_{RAM} clear. If TDES1[30]_{RAM} is clear, it indicates an intermediary buffer. If TDES1[30]_{RAM} is set, it indicates the last buffer of the frame.

After the last buffer of the frame has been transmitted, the DMA writes back the final status information to the Transmit Descriptor 0 (TDES0_{RAM}) word of the descriptor that has the last segment set in Transmit Descriptor 1 (TDES1[30]_{RAM}). At this time, if Interrupt on Completion (TDES1[31]_{RAM}) was set, Transmit Interrupt (ETH0_STATUS.TI) is set, the Next Descriptor is fetched, and the process repeats.

Actual frame transmission begins after the MTL Transmit FIFO has reached either a programmable transmit threshold (ETH0_OPERATION_MODE.TTC), or a full frame is contained in the FIFO. There is also an option for Store and Forward Mode (Operation Mode.TSF). Descriptors are released (Own bit TDES0[31]_{RAM} clears) when the DMA finishes transferring the frame.

Transmit Polling Suspended

Transmit polling can be suspended by either of the following conditions:

- The DMA detects a descriptor owned by the CPU (TDES0[31]_{RAM} = 0). To resume, the driver must give descriptor ownership to the DMA and then issue a Poll Demand command.
- A frame transmission is aborted when a transmit error due to underflow is detected. The appropriate Transmit Descriptor 0 (TDES0_{RAM}) bit is set.

If the second condition occurs, both Abnormal Interrupt Summary (STATUS.AIS) and Transmit Underflow bits (STATUS.TU) are set, and the information is written to Transmit Descriptor 0, causing the suspension. If the DMA goes into SUSPEND state due to the first condition, then both Normal Interrupt Summary (STATUS.NIS) and Transmit Buffer Unavailable (STATUS.TU) are set.

In both cases, the position in the Transmit List is retained. The retained position is that of the descriptor following the Last Descriptor closed by the DMA.

The driver must explicitly issue a Transmit Poll Demand command after rectifying the suspension cause.

14.2.3.3 Reception

The Receive DMA engine's reception sequence is depicted in [Figure 14-5](#) and proceeds as follows:

1. The CPU sets up Receive descriptors (RDES0_{RAM} -RDES3_{RAM}) and sets the Own bit (RDES0[31_{RAM}]).
2. Once the **ETH0_OPERATION_MODE**.SR bit is set, the DMA enters the Run state. While in the Run state, the DMA polls the Receive Descriptor list, attempting to acquire free descriptors. If the fetched descriptor is not free (is owned by the CPU), the DMA enters the Suspend state and jumps to Step 8.
3. The DMA decodes the receive data buffer address from the acquired descriptors.
4. Incoming frames are processed and placed in the acquired descriptor's data buffers.
5. When the buffer is full or the frame transfer is complete, the Receive engine fetches the next descriptor.
6. If the current frame transfer is complete, the DMA proceeds to Step 6. If the DMA does not own the next fetched descriptor and the frame transfer is not complete (EOF is not yet transferred), the DMA sets the Descriptor Error bit in the RDES0 (unless flushing is disabled). The DMA closes the current descriptor (clears the Own bit) and marks it as intermediate by clearing the Last Segment (LS) bit in the RDES0 value (marks it as Last Descriptor if flushing is not disabled), then proceeds to Step 7. If the DMA does own the next descriptor but the current frame transfer is not complete, the DMA closes the current descriptor as intermediate and reverts to Step 3.
7. If IEEE 1588 time stamping is enabled, the DMA writes the time stamp (if available) to the current descriptor's RDES2_{RAM} and RDES3_{RAM}. It then takes the receive frame's status from the MTL and writes the status word to the current descriptor's RDES0_{RAM}, with the Own bit cleared and the Last Segment bit set.
8. The Receive engine checks the latest descriptor's Own bit. If the CPU owns the descriptor (Own bit is 1'b0) the Receive Buffer Unavailable bit (**ETH0_STATUS**.RU) is set and the DMA Receive engine enters the Suspended state (Step 8). If the DMA owns the descriptor, the engine returns to Step 3 and awaits the next frame.
9. Before the Receive engine enters the Suspend state, partial frames are flushed from the Receive FIFO (You can control flushing using Operation Mode.DFF).
10. The Receive DMA exits the Suspend state when a Receive Poll demand is given or the start of next frame is available from the MTL's Receive FIFO. The engine proceeds to Step 1 and refetches the next descriptor.

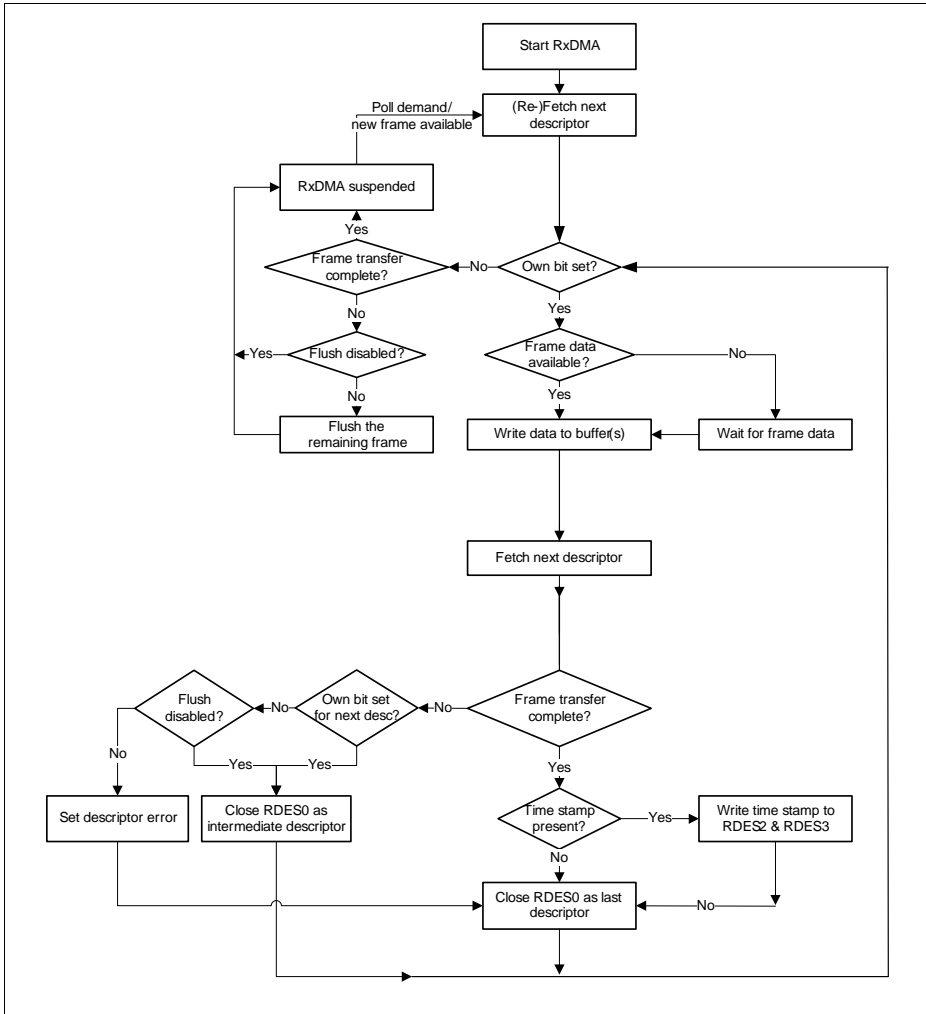


Figure 14-5 Receive DMA Operation

The DMA does not acknowledge accepting the status from the MTL until it has completed the time stamp write-back and is ready to perform status write-back to the descriptor.

If software has enabled time stamping through CSR, when a valid time stamp value is not available for the frame (for example, because the receive FIFO was full before the time stamp could be written to it), the DMA writes all-ones to RDES2_{RAM} and RDES3_{RAM}.

Otherwise (that is, if time stamping is not enabled), the RDES2_{RAM} and RDES3_{RAM} remain unchanged.

Receive Descriptor Acquisition

The Receive Engine always attempts to acquire an extra descriptor in anticipation of an incoming frame. Descriptor acquisition is attempted if any of the following conditions is satisfied:

- The receive Start/Stop bit (**ETH0_OPERATION_MODE.SR**) has been set immediately after being placed in the Run state.
- The data buffer of current descriptor is full before the frame ends for the current transfer.
- The controller has completed frame reception, but the current Receive Descriptor is not yet closed.
- The receive process has been suspended because of a CPU-owned buffer (RDES0[31]_{RAM} = 0) and a new frame is received.
- A Receive poll demand has been issued.

Receive Frame Processing

The ETH transfers the received frames to the XMC4300 memory only when the frame passes the address filter and frame size is greater than or equal to configurable threshold bytes set for the Receive FIFO of the MTL, or when the complete frame is written to the FIFO in Store-and-Forward mode.

If the frame fails the address filtering, it is dropped in the ETH block itself (unless Receive All **ETH0_MAC_FRAME_FILTER.RA** is set). Frames that are shorter than 64 bytes, because of collision or premature termination, can be purged from the MTL Receive FIFO.

After 64 (configurable threshold) bytes have been received, the MTL block requests the DMA block to begin transferring the frame data to the Receive Buffer pointed to by the current descriptor. The DMA sets the First Descriptor bit (RDES0[9]_{RAM}) after the DMA CPU Interface becomes ready to receive a data transfer (if DMA is not fetching transmit data from the XMC4300 RAM), to delimit the frame. The descriptors are released when the Own (RDES[31]_{RAM}) bit is reset to 1'b0, either as the Data buffer fills up or as the last segment of the frame is transferred to the Receive buffer. If the frame is contained in a single descriptor, both Last Descriptor bit(RDES[8]_{RAM}) and First Descriptor bit (RDES[9]_{RAM}) are set.

The DMA fetches the next descriptor, sets the Last Descriptor (RDES[8]_{RAM}) bit, and releases the RDES0_{RAM} status bits in the previous frame descriptor. Then the DMA sets Receive Interrupt flag (**ETH0_STATUS.RI**). The same process repeats unless the DMA encounters a descriptor flagged as being owned by the CPU. If this occurs, the Receive Process sets Receive Buffer Unavailable (STATUS.RU) and then enters the Suspend state. The position in the receive list is retained.

Receive Process Suspended

If a new Receive frame arrives while the Receive Process is in Suspend state, the DMA refetches the current descriptor in the XMC4300 memory. If the descriptor is now owned by the DMA, the Receive Process re-enters the Run state and starts frame reception. If the descriptor is still owned by the CPU, by default, the DMA discards the current frame at the top of the MTL Rx FIFO and increments the missed frame counter. If more than one frame is stored in the MTL Rx FIFO, the process repeats.

The discarding or flushing of the frame at the top of the MTL Rx FIFO can be avoided by setting **ETH0_OPERATION_MODE.DFF** bit. In such conditions, the receive process sets the Receive Buffer Unavailable status and returns to the Suspend state.

14.2.3.4 Interrupts

Interrupts can be generated as a result of various events. The **ETH0_STATUS** Register contains all the bits that might cause an interrupt. The **ETH0_INTERRUPT_ENABLE** Register contains an enable bit for each of the events that can cause an interrupt.

There are two groups of interrupts, Normal and Abnormal, as described in the STATUS Register. Interrupts are cleared by writing 1_B to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both the summary bits are cleared, the interrupt signal to the NVIC is deasserted. If the ETH core is the cause for assertion of the interrupt, then any of the ELI, EMI, or EPI bits of DMA STATUS Register will be set high.

Note: The interrupt signal to the NVIC will be asserted due to any event in the DMA STATUS register only if the corresponding interrupt enable bit is set in DMA Interrupt Enable Register.

Interrupts are not queued and if the interrupt event occurs before the driver has responded to it, no additional interrupts are generated. For example, Receive Interrupt (STATUS.RI) indicates that one or more frames was transferred to the XMC4300 RAM buffer. The driver must scan all descriptors, from the last recorded position to the first one owned by the DMA.

An interrupt is generated only once for simultaneous, multiple events. The driver must scan the STATUS Register for the cause of the interrupt. The interrupt is not generated again unless a new interrupting event occurs, after the driver has cleared the appropriate bit in the STATUS Register. For example, the controller generates a Receive interrupt (DMA STATUS.RI) and the driver begins reading the STATUS Register. Next, Receive Buffer Unavailable (STATUS Register) occurs. The driver clears the Receive interrupt. Even then, the DMA interrupt signal to the NVIC is not deasserted, due to the active or pending Receive Buffer Unavailable interrupt.

An interrupt timer (**ETH0_RECEIVE_INTERRUPT_WATCHDOG_TIMER**) is given for flexible control of Receive Interrupt (STATUS.RI). When this Interrupt timer is programmed with a non-zero value, it will get activated as soon as the RxDMA

completes a transfer of a received frame to system memory without asserting the Receive Interrupt because it is not enabled in the corresponding Receive Descriptor (RDES1[31]_{RAM} in Table 7-3). When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding **ETH0_INTERRUPT_ENABLE**.RI bit is enabled. This timer gets disabled before it runs out, when a frame is transferred to memory and the **ETH0_STATUS**.RI is set because it is enabled for that descriptor.

14.2.4 DMA Descriptors

This chapter describes the descriptor format used by the ETH DMA. The ETH DMA descriptors are held in RAM. To avoid confusion with the ETH registers the DMA descriptors use the subscript $]_{RAM}$ for example $RDES0[0]_{RAM}$.

14.2.4.1 Descriptor Formats

The DMA in the Ethernet subsystem transfers data based on a linked list of descriptors, as explained in **DMA Controller**. The default descriptor formats (common for both Receive and Transmit Descriptors) are shown in **Figure 14-6**, and field descriptions are provided in **“Receive Descriptor” on Page 14-41** to **“Transmit Descriptor” on Page 14-47**.

Note:

7. *Changes to the default descriptor format when IEEE1588 time stamping is enabled are described in **Chapter “Descriptor Format With IEEE 1588 Time Stamping Enabled” on Page 14-53**“.*

Each descriptor contains two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory management schemes.

The descriptor addresses must be aligned to 32 bit word boundaries .

Figure 14-6 show the descriptor format .

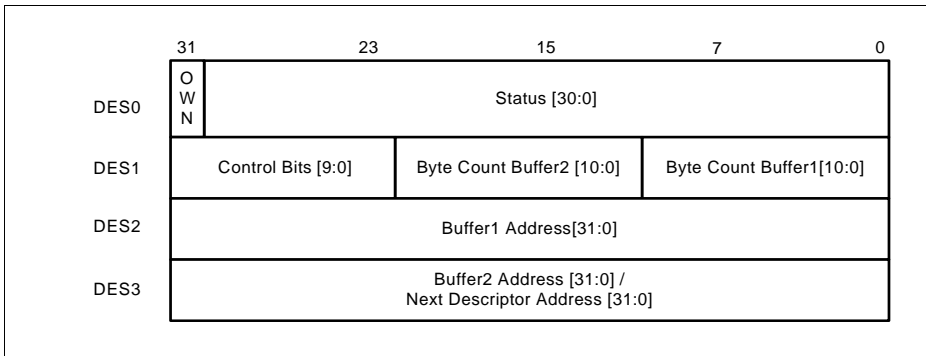


Figure 14-6 Rx/Tx Descriptors

Receive Descriptor

The ETH Subsystem requires at least two descriptors when receiving a frame. The Receive state machine of the DMA (in the ETH Subsystem) always attempts to acquire an extra descriptor in anticipation of an incoming frame. (The size of the incoming frame

is unknown). Before the RxDMA closes a descriptor, it will attempt to acquire the next descriptor even if no frames are received.

In a single descriptor (receive) system, the subsystem will generate a descriptor error if the receive buffer is unable to accommodate the incoming frame and the next descriptor is not owned by the DMA. Thus, the CPU is forced to increase either its descriptor pool or the buffer size. Otherwise, the subsystem starts dropping all incoming frames.

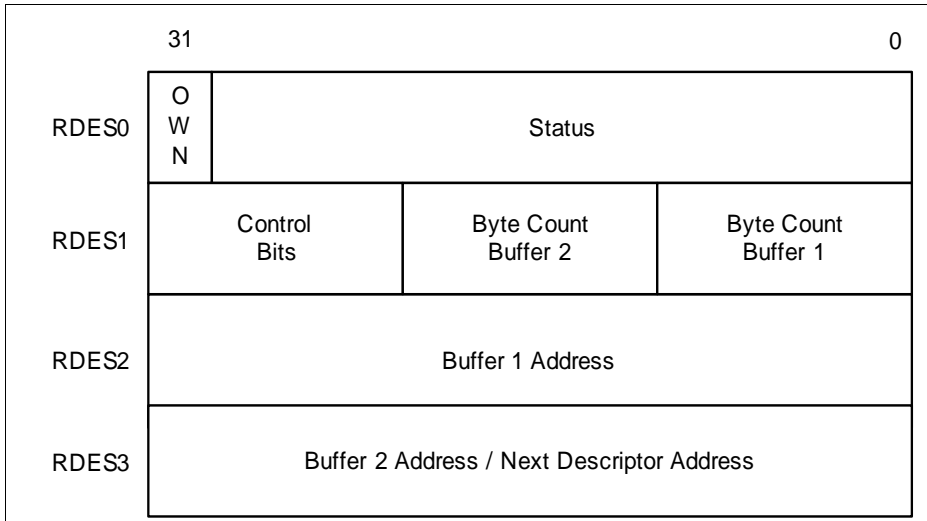


Figure 14-7 Receive Descriptor Format

Receive Descriptor 0 (RDES0_{RAM})

RDES0_{RAM} contains the received frame status, the frame length, and the descriptor ownership information. The format of the descriptor is given in tables [Table 14-4](#) through [Table 14-12](#).

Table 14-4 Receive Descriptor 0

Bit	Description
31	<p>OWN: Own Bit</p> <p>When set, this bit indicates that the descriptor is owned by the DMA of the ETH Subsystem. When this bit is reset, this bit indicates that the descriptor is owned by the CPU. The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full.</p>
30	<p>AFM: Destination Address Filter Fail</p> <p>When set, this bit indicates a frame that failed in the DA Filter in the ETH Core.</p>

Table 14-4 Receive Descriptor 0 (cont'd)

Bit	Description
29:1	FL: Frame Length
6	<p>These bits indicate the byte length of the received frame that was transferred to XMC4300 memory (including CRC). This field is valid when Last Descriptor (RDES0[8]_{RAM}) is set and either the Descriptor Error (RDES0[14]_{RAM}) or Overflow Error bits are reset. The frame length also includes the two bytes appended to the Ethernet frame when IP checksum calculation (Type 1) is enabled and the received frame is not a MAC control frame.</p> <p>This field is valid when Last Descriptor (RDES0[8]_{RAM}) is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current frame.</p>
15	<p>ES: Error Summary</p> <p>Indicates the logical OR of the following bits:</p> <ul style="list-style-type: none"> • RDES0[0]_{RAM}: Payload Checksum Error • RDES0[1]_{RAM}: CRC Error • RDES0[3]_{RAM}: Receive Error • RDES0[4]_{RAM}: Watchdog Timeout • RDES0[6]_{RAM}: Late Collision • RDES0[7]_{RAM}: IPC Checksum (Type 2) / Giant Frame • RDES0[11]_{RAM}: Overflow Error • RDES0[14]_{RAM}: Descriptor Error <p>This field is valid only when the Last Descriptor (RDES0[8]_{RAM}) is set.</p>
14	<p>DE: Descriptor Error</p> <p>When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next Descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]_{RAM}) is set.</p>
13	<p>SAF: Source Address Filter Fail</p> <p>When set, this bit indicates that the SA field of frame failed the SA Filter in the ETH Core.</p>
12	<p>LE: Length Error</p> <p>When set, this bit indicates that the actual length of the frame received and that the Length/ Type field does not match. This bit is valid only when the Frame Type (RDES0[5]_{RAM}) bit is reset. Length error status is not valid when CRC error is present.</p>
11	<p>OE: Overflow Error</p> <p>When set, this bit indicates that the received frame was damaged due to buffer overflow in MTL.</p>

Table 14-4 Receive Descriptor 0 (cont'd)

Bit	Description
10	VLAN: VLAN Tag When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the ETH Core.
9	FS: First Descriptor When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame.
8	LS: Last Descriptor When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame
7	IPC Checksum Error/Giant Frame This bit indicates an error in the IPv4 or IPv6 header. This error can be due to inconsistent Ethernet Type field and IP header Version field values, a header checksum mismatch in IPv4, or an Ethernet frame lacking the expected number of IP header bytes. Refer to Table 14-5 for more details.
6	LC: Late Collision When set, this bit indicates that a late collision has occurred while receiving the frame in Half-Duplex mode.
5	FT: Frame Type When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 0600 _H). When this bit is reset, it indicates that the received frame is an IEEE802.3 frame. This bit is not valid for Runt frames less than 14 bytes. Refer to Table 14-5 for more details.
4	RWT: Receive Watchdog Timeout When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout.
3	RE: Receive Error When set, this bit indicates that the MII Receive Error signal is asserted while Carrier Sense signal is asserted during frame reception. This error also includes carrier extension error in MII and Half-duplex mode. Error can be of less/no extension, or error (rxd ≠ 0f) during extension.
2	DE: Dribble Bit Error When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in MII Mode.

Table 14-4 Receive Descriptor 0 (cont'd)

Bit	Description
1	CE: CRC Error When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8] _{RAM}) is set.
0	Payload Checksum Error When set, indicates the TCP, UDP, or ICMP checksum the core calculated does not match the received encapsulated TCP, UDP, or ICMP segment's Checksum field. This bit is also set when the received number of payload bytes does not match the value indicated in the Length field of the encapsulated IPv4 or IPv6 datagram in the received Ethernet frame. Refer to Table 14-5 for more details.

The permutations of bits 5, 7, and 0 reflect the conditions discussed in [Table 14-5](#).

Table 14-5 Receive Descriptor 0 When COE (Type 2) Is Enabled

Bit 5: Frame Type	Bit 7: IPC Checksum Error	Bit 0: Payload Checksum Error	Frame Status
0	0	0	IEEE 802.3 Type frame (Length field value is less than 0600 _H)
1	0	0	IPv4/IPv6 Type frame, no checksum error detected
1	0	1	IPv4/IPv6 Type frame with a payload checksum error (as described for PCE) detected
1	1	0	IPv4/IPv6 Type frame with an IP header checksum error (as described for IPC CE) detected
1	1	1	IPv4/IPv6 Type frame with both IP header and payload checksum errors detected
0	0	1	IPv4/IPv6 Type frame with no IP header checksum error and the payload check bypassed, due to an unsupported payload
0	1	1	A Type frame that is neither IPv4 or IPv6 (the Checksum Offload engine bypasses checksum completely.)
0	1	0	Reserved

Receive Descriptor 1 (RDES1_{RAM})

RDES1_{RAM} contains the buffer sizes and other bits that control the descriptor chain/ring.

Note: See [Buffer Size Calculations](#) for further detail on calculating buffer sizes.

Table 14-6 Receive Descriptor 1

Bit	Description
31	<p>Disable Interrupt on Completion</p> <p>When set, this bit will prevent the setting of the ETH0_STATUS.RI bit of the Status Register for the received frame that ends in the buffer pointed to by this descriptor. This, in turn, will disable the assertion of the interrupt to the CPU due to RI for that frame.</p>
30:26	Reserved
25	<p>RER: Receive End of Ring</p> <p>When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a Descriptor Ring.</p>
24	<p>RCH: Second Address Chained</p> <p>When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When RDES1[24]_{RAM} is set, RBS2_{RAM} (RDES1[21-11]_{RAM}) is a “don't care” value. RDES1[25]_{RAM} takes precedence over RDES1[24]_{RAM}.</p>
23:22	Reserved
21:11	<p>RBS2: Receive Buffer 2 Size</p> <p>These bits indicate the second data buffer size in bytes. The buffer size must be a multiple of 4, even if the value of RDES3_{RAM} (buffer2 address pointer) is not aligned to bus width. In the case where the buffer size is not a multiple of 4, the resulting behavior is undefined. This field is not valid if RDES1[24]_{RAM} is set.</p>
10:0	<p>RBS1: Receive Buffer 1 Size</p> <p>Indicates the first data buffer size in bytes. The buffer size must be a multiple of 4, even if the value of RDES2_{RAM} (buffer1 address pointer) is not aligned. In the case where the buffer size is not a multiple of 4/8/16, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (Bit 24).</p>

Receive Descriptor 2 (RDES2_{RAM})

RDES2_{RAM} contains the address pointer to the first data buffer in the descriptor.

Note: See [XMC4300 Data Buffer Alignment](#) for further detail on buffer address alignment.

Table 14-7 Receive Descriptor 2 (Default Operation)

Bit	Description
31:0	<p>Buffer 1 Address Pointer</p> <p>These bits indicate the physical address of Buffer 1. There are no limitations on the buffer address alignment except for the following condition: The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. Note that the DMA performs a write operation with the RDES2[3/2/1:0]_{RAM} bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[3/2/1:0]_{RAM} if the address pointer is to a buffer where the middle or last part of the frame is stored.</p>

Receive Descriptor 3 (RDES3_{RAM})

RDES3_{RAM} contains the address pointer either to the second data buffer in the descriptor or to the next descriptor.

Table 14-8 Receive Descriptor 3

Bit	Description
31:0	<p>Buffer 2 Address Pointer (Next Descriptor Address)</p> <p>These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[24]_{RAM}) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present.</p> <p>If RDES1[24]_{RAM} is set, the buffer (Next Descriptor) address pointer must be bus width-aligned (RDES3[3, 2, or 1:0]_{RAM} = 0, corresponding to a bus width of 128, 64, or 32. LSBs are ignored internally.) However, when RDES1[24]_{RAM} is reset, there are no limitations on the RDES3_{RAM} value, except for the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3[3, 2, or 1:0]_{RAM} if the address pointer is to a buffer where the middle or last part of the frame is stored.</p>

Transmit Descriptor

The descriptor addresses must be aligned to the 32 bit word boundary . **Figure 14-8** shows the transmit descriptor format.

Each descriptor is provided with two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory-management schemes.

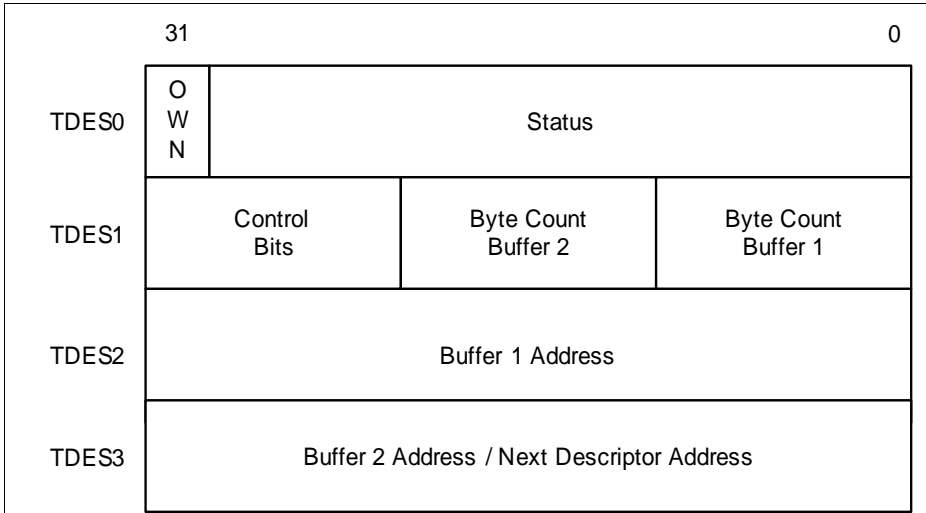


Figure 14-8 Transmit Descriptor Format

Transmit Descriptor 0 (TDES0_{RAM})

TDES0_{RAM} contains the transmitted frame status and the descriptor ownership information.

Table 14-9 Transmit Descriptor 0

Bit	Description
31	<p>OWN: Own Bit</p> <p>When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, this bit indicates that the descriptor is owned by the CPU. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are empty. The ownership bit of the First Descriptor of the frame should be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.</p>
30:18	Reserved

Table 14-9 Transmit Descriptor 0 (cont'd)

Bit	Description
17	<p>TTSS: Tx Time Stamp Status</p> <p>This status bit indicates that a time stamp has been captured for the corresponding transmit frame. When this bit is set, TDES2_{RAM} and TDES3_{RAM} have time stamp values that were captured for the transmit frame. This field is valid only when the Last Segment control bit (TDES1[30]_{RAM}) in a descriptor is set. This bit is valid only when IEEE1588 time stamping feature is enabled; otherwise, it is reserved.</p>
16	<p>IHE: IP Header Error</p> <p>When set, this bit indicates that the Checksum Offload engine detected an IP header error and consequently did not modify the transmitted frame for any checksum insertion.</p>
15	<p>ES: Error Summary</p> <p>Indicates the logical OR of the following bits:</p> <ul style="list-style-type: none"> • TDES0[14]_{RAM}: Jabber Timeout • TDES0[13]_{RAM}: Frame Flush • TDES0[11]_{RAM}: Loss of Carrier • TDES0[10]_{RAM}: No Carrier • TDES0[9]_{RAM}: Late Collision • TDES0[8]_{RAM}: Excessive Collision • TDES0[2]_{RAM}: Excessive Deferral • TDES0[1]_{RAM}: Underflow Error
14	<p>JT: Jabber Timeout</p> <p>When set, this bit indicates the ETH transmitter has experienced a jabber timeout. This bit is only set when the ETH configuration register's JD bit is not set.</p>
13	<p>FF: Frame Flushed</p> <p>When set, this bit indicates that the DMA/MTL flushed the frame due to a SW flush command given by the CPU.</p>
12	<p>PCE: Payload Checksum Error</p> <p>This bit, when set, indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either due to insufficient bytes, as indicated by the IP Header's Payload Length field, or the MTL starting to forward the frame to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet frame being transmitted: to avoid deadlock, the MTL starts forwarding the frame when the FIFO is full, even in Store-and-Forward mode.</p>

Table 14-9 Transmit Descriptor 0 (cont'd)

Bit	Description
11	<p>LC: Loss of Carrier</p> <p>When set, this bit indicates that Loss of Carrier occurred during frame transmission . This is valid only for the frames transmitted without collision and when the ETH operates in Half-Duplex Mode.</p>
10	<p>NC: No Carrier</p> <p>When set, this bit indicates that the carrier sense signal form the PHY was not asserted during transmission.</p>
9	<p>LC: Late Collision</p> <p>When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte times including Preamble in MII Mode). Not valid if Underflow Error is set.</p>
8	<p>EC: Excessive Collision</p> <p>When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the DR (Disable Retry) bit in the ETH Configuration Register is set, this bit is set after the first collision and the transmission of the frame is aborted.</p>
7	<p>VF: VLAN Frame</p> <p>When set, this bit indicates that the transmitted frame was a VLAN-type frame.</p>
6:3	<p>CC: Collision Count</p> <p>This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the Excessive Collisions bit (TDES0[8]_{RAM}) is set.</p>
2	<p>ED: Excessive Deferral</p> <p>When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000 Mbit/s mode, or in Jumbo Frame enabled mode) if the Deferral Check (DC) bit is set high in the ETH Control Register.</p>
1	<p>UF: Underflow Error</p> <p>When set, this bit indicates that the ETH aborted the frame because data arrived late from the XMC4300 memory. Underflow Error indicates that the DMA encountered an empty Transmit Buffer while transmitting the frame. The transmission process enters the suspended state and sets both STATUS.TU and STATUS.TI.</p>
0	<p>DB: Deferred Bit</p> <p>When set, this bit indicates that the ETH defers before transmission because of the presence of carrier. This bit is valid only in Half-Duplex mode.</p>

Transmit Descriptor 1 (TDES1_{RAM})

TDES1_{RAM} contains the buffer sizes and other bits which control the descriptor chain/ring and the frame being transferred.

Note: See [Buffer Size Calculations](#) for further detail on calculating buffer sizes.

Table 14-10 Transmit Descriptor 1

Bit	Description
31	<p>IC: Interrupt on Completion</p> <p>When set, this bit sets Transmit Interrupt, STATUS.TI bit after the present frame has been transmitted.</p>
30	<p>LS: Last Segment</p> <p>When set, this bit indicates that the buffer contains the last segment of the frame.</p>
29	<p>FS: First Segment</p> <p>When set, this bit indicates that the buffer contains the first segment of a frame.</p>
28:27	<p>CIC: Checksum Insertion Control</p> <p>These bits control the insertion of checksums in Ethernet frames that encapsulate TCP, UDP, or ICMP over IPv4 or IPv6 as described below.</p> <ul style="list-style-type: none"> • 00: Do nothing. Checksum Engine is bypassed • 01: Insert IPv4 header checksum. Use this value to insert IPv4 header checksum when the frame encapsulates an IPv4 datagram. • 10: Insert TCP/UDP/ICMP checksum. The checksum is calculated over the TCP, UDP, or ICMP segment only and the TCP, UDP, or ICMP pseudo-header checksum is assumed to be present in the corresponding input frame's Checksum field. An IPv4 header checksum is also inserted if the encapsulated datagram conforms to IPv4. • 11: Insert a TCP/UDP/ICMP checksum that is fully calculated in this engine. In other words, the TCP, UDP, or ICMP pseudo-header is included in the checksum calculation, and the input frame's corresponding Checksum field has an all-zero value. An IPv4 Header checksum is also inserted if the encapsulated datagram conforms to IPv4. <p>The Checksum engine detects whether the TCP, UDP, or ICMP segment is encapsulated in IPv4 or IPv6 and processes its data accordingly.</p>
26	<p>DC: Disable CRC</p> <p>When set, the ETH does not append the Cyclic Redundancy Check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES1[29]_{RAM}) bit is set.</p>
25	<p>TER: Transmit End of Ring</p> <p>When set, this bit indicates that the descriptor list reached its final descriptor. The returns to the base address of the list, creating a descriptor ring.</p>

Table 14-10 Transmit Descriptor 1 (cont'd)

Bit	Description
24	TCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES1[24] _{RAM} is set, TBS2 (TDES1[21–11] _{RAM}) are “don't care” values. TDES1[25] _{RAM} takes precedence over TDES1[24] _{RAM} .
23	DP: Disable Padding When set, the ETH does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes and the CRC field is added despite the state of the DC (TDES1[26] _{RAM}) bit. This is valid only when the first segment (TDES1[29] _{RAM}) is set.
22	TTSE: Transmit Time Stamp Enable When set, this bit enables IEEE1588 hardware time stamping for the transmit frame referenced by the descriptor. This field is valid only when the First Segment control bit (TDES1[29] _{RAM}) is set.
21:11	TBS2: Transmit Buffer 2 Size These bits indicate the Second Data Buffer in bytes. This field is not valid if TDES1[24] _{RAM} is set.
10:0	TBS1: Transmit Buffer 1 Size These bits indicate the First Data Buffer byte size. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of TCH (Bit 24).

Transmit Descriptor 2 (TDES2)

TDES2 contains the address pointer to the first buffer of the descriptor.

Table 14-11 Transmit Descriptor 2

Bit	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment. See XMC4300 Data Buffer Alignment for further detail on buffer address alignment.

Transmit Descriptor 3 (TDES3_{RAM})

TDES3_{RAM} contains the address pointer either to the second buffer of the descriptor or the next descriptor.

Table 14-12 Transmit Descriptor 3

Bit	Description
31:0	Buffer 2 Address Pointer (Next Descriptor Address) Indicates the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (TDES1[24] _{RAM}) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES1[24] _{RAM} is set. (LSBs are ignored internally.)

Descriptor Format With IEEE 1588 Time Stamping Enabled

The default descriptor format (as described in [“Receive Descriptor” on Page 14-55](#) and [“Transmit Descriptor” on Page 14-47](#)), and field descriptions remain unchanged when created by software (Own bit is set in DES0_{RAM}). However, if the software has enabled IEEE 1588 functionality, the DES2_{RAM} and DES3_{RAM} descriptor fields (see [Figure 14-9](#)) take on a different meaning when the DMA closes the descriptor (own bit in DES0_{RAM} is cleared).

The DMA updates the DES2_{RAM} and DES3_{RAM} with the time stamp value before clearing the Own bit in DES0_{RAM}.

DES2_{RAM} is updated with the lower 32 time stamp bits (the Sub-Second field, called TSL in subsequent sections) and DES3_{RAM} is updated with the upper 32 time stamp bits (the Seconds field, called TSH in subsequent sections).

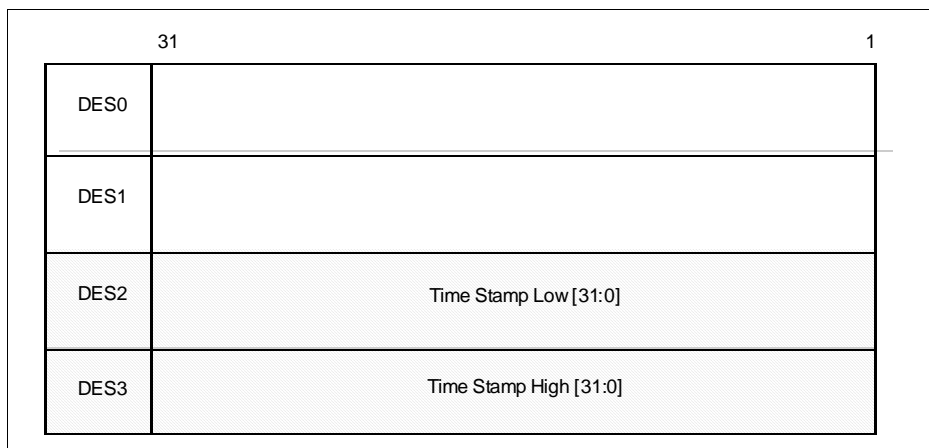


Figure 14-9 Receive Descriptor Fields When DMA Clears the Own Bit

The following sections describe the details specific to receive and transmit descriptors in this mode.

Receive Descriptor

Receive Time Stamp

The tables below describe the fields that have different meaning for $RDES2_{RAM}$ and $RDES3_{RAM}$ when the receive descriptor is closed and time stamping is enabled.

*Note: When software disables the time stamping feature (the **ETH0_TIMESTAMP_CONTROL.TSENA** bit is low), the DMA does not update the descriptor's $RDES2_{RAM}/RDES3_{RAM}$ fields before closing the $RDES0_{RAM}$.*

Table 14-13 Receive Descriptor Fields (RDES2)

Bit	Description
31:0	<p>RTSL: Receive Frame Time Stamp Low</p> <p>The DMA updates this field with the least significant 32 bits of the time stamp captured for the corresponding receive frame. The DMA updates this field only for the last descriptor of the receive frame indicated by Last Descriptor status bit ($RDES0[8]_{RAM}$). When this field and the RTSH field in $RDES3_{RAM}$ show an all-ones value, the time stamp must be treated as corrupt.</p>

Table 14-14 Receive Descriptor Fields (RDES3)

Bit	Description
31:0	<p>RTSH: Receive Frame Time Stamp High</p> <p>The DMA updates this field with the most significant 32 bits of the time stamp captured for the corresponding receive frame. The DMA updates this field only for the last descriptor of the receive frame indicated by Last Descriptor status bit ($RDES0[8]_{RAM}$).</p> <p>When this field and $RDES2_{RAM}$'s RTSL field show all-ones values, the time stamp must be treated as corrupt.</p>

Transmit Descriptor

In addition to the changes described in **“Descriptor Format With IEEE 1588 Time Stamping Enabled” on Page 14-53**, the Transmit descriptor has additional control and status bits (TTSE and TTSS, respectively) for time stamping, as shown in **Figure 14-10**. Software sets the TTSE bit (when the Own bit is set), instructing the core to generate a time stamp for the corresponding Ethernet frame being transmitted. The DMA sets the TTSS bit if the time stamp has been updated in the $TDES2_{RAM}$ and $TDES3_{RAM}$ fields when the descriptor is closed (Own bit is cleared).

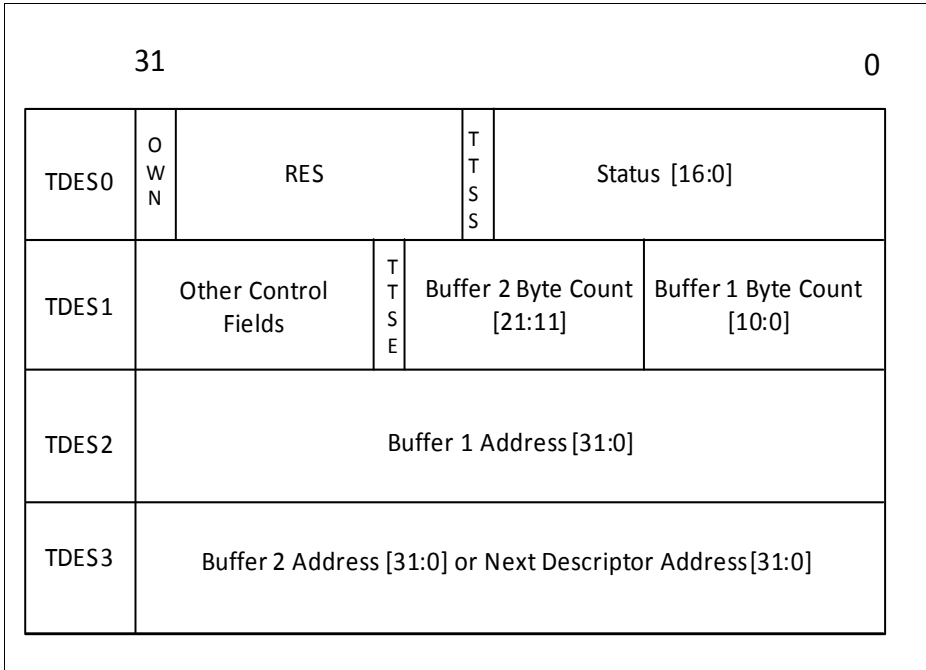


Figure 14-10 Transmit Descriptor Fields

Transmit Time Stamp Control and Status Fields

The value of this field shall be preserved by the DMA at the time of closing the descriptor.

Updates to [Table 14-8](#) and [Table 14-9](#) are described below.

Table 14-15 Transmit Time Stamp Status – Normal Descriptor Format Case (TDES0RAM)

Bit	Description
17	<p>TTSS: Transmit Time Stamp Status</p> <p>This field is a status bit indicating that a time stamp was captured for the corresponding transmit frame. When this bit is set, both TDES2RAM and TDES3RAM have a time stamp value that was captured for the transmit frame. This field is valid only when the Last Segment control bit (TDES1[30]RAM in the descriptor) is set.</p>

Table 14-16 Transmit Time Stamp Control – Normal Descriptor Format Case (TDES1_{RAM})

Bit	Description
22	<p>TTSE: Transmit Time Stamp Enable</p> <p>When set, this field enables IEEE1588 hardware time stamping for the transmit frame described by the descriptor.</p> <p>This field is valid only when the First Segment control bit (TDES1[29]_{RAM} in the descriptor) is set.</p>

Transmit Time Stamp Field

The transmit descriptor format and field descriptions remain unchanged when they are created by software (when the Own bit is set). However, when the DMA closes the last descriptor and IEEE 1588 functionality is enabled (the Own bit is cleared), the TDES2_{RAM} and TDES3 descriptor fields are updated with the time stamp, if taken, for that frame.

[Table 14-17](#) and [Table 14-18](#) describe the fields that have a different meaning when the descriptor is closed.

Table 14-17 Transmit Descriptor Fields (TDES2_{RAM})

Bit	Description
31:0	<p>TTSL: Transmit Frame Time Stamp Low</p> <p>This field is updated by DMA with the least significant 32 bits of the time stamp captured for the corresponding transmit frame. This field has the time stamp only if the Last Segment control bit (LS) in the descriptor is set.</p>

Table 14-18 Transmit Descriptor Fields (TDES3)

Bit	Description
31:0	<p>TTSH: Transmit Frame Time Stamp High</p> <p>This field is updated by DMA with the most significant 32 bits of the time stamp captured for the corresponding transmit frame. This field has the time stamp only if the Last Segment control bit (LS) in the descriptor is set.</p>

Alternate or Enhanced Descriptors

The alternate (or enhanced) descriptor structure can have 8 DWORDS (32-bytes) instead of the 4 DWORDS as in the case of normal descriptor format. The features of the alternate descriptor structure are

- The normal descriptor structure allows data buffers of up to 2.048 bytes. The alternative descriptor structure has been implemented to support buffers of up to 8 KB (useful for Jumbo frames).
- There is a re-assignment of control and status bits in TDES0, TDES1, RDES0 (Advanced time stamp or IPC full offload configuration), RDES1.
- The transmit descriptor stores the time stamp in TDES6 and TDES7 when advanced time stamp feature is selected.
- This receive descriptor structure is also used for storing the extended status (RDES4) and time stamp (RDES6 and RDES7) when advanced time stamp feature or IPC full offload is selected.
- When alternate descriptor mode is selected, and Time-stamping feature is enabled, the software needs to allocate 32-bytes (8 DWORDS) of memory for every descriptor. When Time-stamping or Receive IPC FullOffload engine are not enabled, the extended descriptors are not required and the SW can use alternate descriptors with the default size of 16 bytes. The core also needs to be configured for this change using the DMA Bus Mode Register[7].
- When alternate descriptor is chosen without Time Stamp or Full IPC Offload feature, the descriptor size is always 4 DWORDs (DES0-DES3).

The description or bit-mapping alternate descriptor structure (in Little Endian mode) is given below.

Note: When alternate descriptor with only Full IPC Offload (Type 2) is selected, it is not backward compatible to the previous release 3.4x with respect to status bits[7,5,0] in RDES0. In this mode, you should enable the extended descriptor mode (8 DWORDS) to get the IPC checksum engine status in RDES4.

Transmit Descriptor

The transmit descriptor structure is shown in [Figure 14-11](#). The application software must program the control bits TDES0[31:20] during descriptor initialization. When the DMA updates the descriptor, it write backs all the control bits except the OWN bit (which it clears) and updates the status bits[19:0]. The contents of the transmitter descriptor word 0 (TDES0) through word 3 (TDES3) are given in [Table 14-19](#) through [Table 14-22](#), respectively.

With the advance time stamp support, the snapshot of the time stamp to be taken can be enabled for a given frame by setting the “TTSE: Transmit Time Stamp Enable” (bit-25 of TDES0). When the descriptor is closed (i.e. when the OWN bit is cleared), the time-stamp is written into TDES6 and TDES7. This is indicated by the status bit “TTSS: Transmit Time Stamp Status” (bit-17 of TDES0). This is shown in [Figure 14-11](#). The contents of TDES6 and TDES7 are mentioned in [Table 14-23](#) and [Table 14-24](#).

Note: When either of Advanced Time Stamp or IPC Offload (Type 2) features is enabled, the SW should set the DMA Bus Mode register[7], so that the DMA operates with extended descriptor size. When this control bit is reset, the TDES4-TDES7 descriptor space are not valid.

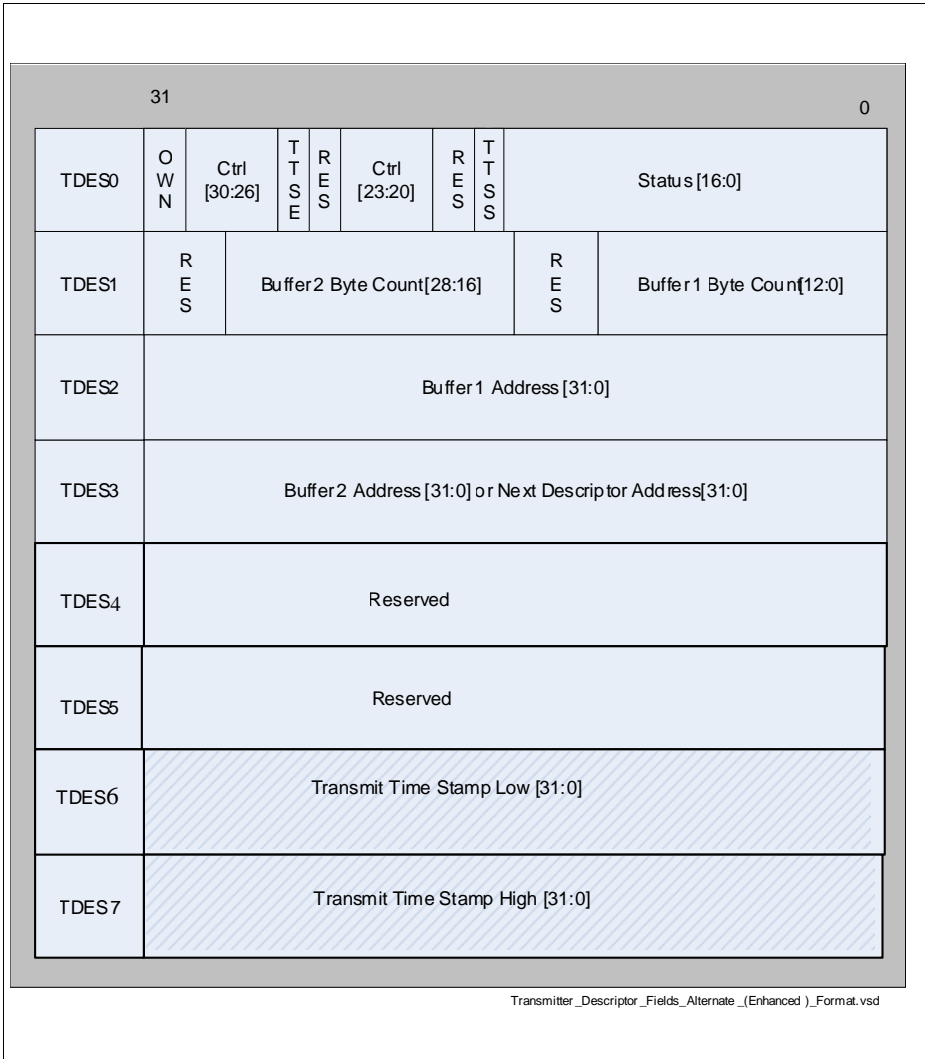


Figure 14-11 Transmitter Descriptor Fields - Alternate (Enhanced) Format

Table 14-19 Transmit Descriptor Word 0 (TDES0)

Bit	Description
31	<p>OWN: Own Bit</p> <p>When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, it indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are read completely. The ownership bit of the frame's first descriptor must be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.</p>
30	<p>IC: Interrupt on Completion</p> <p>When set, this bit sets the Transmit Interrupt (Register 5[0]) after the present frame has been transmitted.</p>
29	<p>LS: Last Segment</p> <p>When set, this bit indicates that the buffer contains the last segment of the frame.</p>
28	<p>FS: First Segment</p> <p>When set, this bit indicates that the buffer contains the first segment of a frame.</p>
27	<p>DC: Disable CRC</p> <p>When this bit is set, the GMAC does not append a cyclic redundancy check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES0[28]) is set.</p>
26	<p>DP: Disable Pad</p> <p>When set, the GMAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes, and the CRC field is added despite the state of the DC (TDES0[27]) bit. This is valid only when the first segment (TDES0[28]) is set.</p>
25	<p>TTSE: Transmit Time Stamp Enable</p> <p>When set, this bit enables IEEE1588 hardware time stamping for the transmit frame referenced by the descriptor. This field is valid only when the First Segment control bit (TDES0[28]) is set.</p>
24	<p>Reserved</p>

Table 14-19 Transmit Descriptor Word 0 (TDES0) (cont'd)

Bit	Description
23:22	<p>CIC: Checksum Insertion Control</p> <p>These bits control the checksum calculation and insertion. Bit encodings are as shown below.</p> <ul style="list-style-type: none"> • 2'b00: Checksum Insertion Disabled. • 2'b01: Only IP header checksum calculation and insertion are enabled. • 2'b10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. • 2'b11: IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware. <p>This field is reserved when the IPC_FULL_OFFLOAD configuration parameter is not selected.</p>
21	<p>TER: Transmit End of Ring</p> <p>When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring.</p>
20	<p>TCH: Second Address Chained</p> <p>When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES0[20] is set, TBS2 (TDES1[28:16]) is a “don't care” value. TDES0[21] takes precedence over TDES0[20].</p>
19:18	Reserved
17	<p>TTSS: Transmit Time Stamp Status</p> <p>This field is used as a status bit to indicate that a time stamp was captured for the described transmit frame. When this bit is set, TDES2 and TDES3 have a time stamp value captured for the transmit frame. This field is only valid when the descriptor's Last Segment control bit (TDES0[29]) is set.</p>
16	<p>IHE: IP Header Error</p> <p>When set, this bit indicates that the GMAC transmitter detected an error in the IP datagram header. The transmitter checks the header length in the IPv4 packet against the number of header bytes received from the application and indicates an error status if there is a mismatch. For IPv6 frames, a header error is reported if the main header length is not 40 bytes. Furthermore, the Ethernet Length/Type field value for an IPv4 or IPv6 frame must match the IP header version received with the packet. For IPv4 frames, an error status is also indicated if the Header Length field has a value less than 0x5.</p>

Table 14-19 Transmit Descriptor Word 0 (TDES0) (cont'd)

Bit	Description
15	<p>ES: Error Summary Indicates the logical OR of the following bits:</p> <ul style="list-style-type: none"> • TDES0[14]: Jabber Timeout • TDES0[13]: Frame Flush • TDES0[11]: Loss of Carrier • TDES0[10]: No Carrier • TDES0[9]: Late Collision • TDES0[8]: Excessive Collision • TDES0[2]: Excessive Deferral • TDES0[1]: Underflow Error • TDES0[16]: IP Header Error • TDES0[12]: IP Payload Error
14	<p>JT: Jabber Timeout When set, this bit indicates the GMAC transmitter has experienced a jabber time-out. This bit is only set when the GMAC configuration register's JD bit is not set.</p>
13	<p>FF: Frame Flushed When set, this bit indicates that the DMA/MTL flushed the frame due to a software Flush command given by the CPU.</p>
12	<p>IPE: IP Payload Error When set, this bit indicates that GMAC transmitter detected an error in the TCP, UDP, or ICMP IP datagram payload. The transmitter checks the payload length received in the IPv4 or IPv6 header against the actual number of TCP, UDP, or ICMP packet bytes received from the application and issues an error status in case of a mismatch.</p>
11	<p>LC: Loss of Carrier When set, this bit indicates that a loss of carrier occurred during frame transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during frame transmission). This is valid only for the frames transmitted without collision when the GMAC operates in Half-Duplex mode.</p>
10	<p>NC: No Carrier When set, this bit indicates that the Carrier Sense signal from the PHY was not asserted during transmission.</p>
9	<p>LC: Late Collision When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte-times, including preamble, in MII mode and 512 byte-times, including preamble and carrier extension, in GMII mode). This bit is not valid if the Underflow Error bit is set.</p>

Table 14-19 Transmit Descriptor Word 0 (TDES0) (cont'd)

Bit	Description
8	EC: Excessive Collision When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the DR (Disable Retry) bit in the GMAC Configuration register is set, this bit is set after the first collision, and the transmission of the frame is aborted.
7	VF: VLAN Frame When set, this bit indicates that the transmitted frame was a VLAN-type frame.
6:3	CC: Collision Count This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the Excessive Collisions bit (TDES0[8]) is set.
2	ED: Excessive Deferral When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1,000-Mbit/s mode or if Jumbo Frame is enabled) if the Deferral Check (DC) bit in the GMAC Control register is set high.
1	UF: Underflow Error When set, this bit indicates that the GMAC aborted the frame because data arrived late from the Host memory. Underflow Error indicates that the DMA encountered an empty transmit buffer while transmitting the frame. The transmission process enters the Suspended state and sets both Transmit Underflow (Register 5[5]) and Transmit Interrupt (Register 5[0]).
0	DB: Deferred Bit When set, this bit indicates that the GMAC defers before transmission because of the presence of carrier. This bit is valid only in Half-Duplex mode.

Table 14-20 Transmit Descriptor Word 1 (TDES1)

Bit	Description
31:29	Reserved
28:16	TBS2: Transmit Buffer 2 Size These bits indicate the second data buffer size in bytes. This field is not valid if TDES0[20] is set. See Buffer Size Calculations for further detail on calculating buffer sizes.

Table 14-20 Transmit Descriptor Word 1 (TDES1) (cont'd)

Bit	Description
15:13	Reserved
12:0	<p>TBS1: Transmit Buffer 1 Size</p> <p>These bits indicate the first data buffer byte size, in bytes. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor, depending on the value of TCH (TDES0[20]).</p>

Table 14-21 Transmit Descriptor 2 (TDES2)

Bit	Description
31:0	<p>Buffer 1 Address Pointer</p> <p>These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment. See XMC4300 Data Buffer Alignment for further detail on buffer address alignment.</p>

Table 14-22 Transmit Descriptor 3 (TDES3)

Bit	Description
31:0	<p>Buffer 2 Address Pointer (Next Descriptor Address)</p> <p>Indicates the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (TDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES1[24] is set. (LSBs are ignored internally.)</p>

Table 14-23 Transmit Descriptor 6 (TDES6)

Bit	Description
31:0	<p>TTSL: Transmit Frame Time Stamp Low</p> <p>This field is updated by DMA with the least significant 32 bits of the time stamp captured for the corresponding transmit frame. This field has the time stamp only if the Last Segment bit (LS) in the descriptor is set and Time stamp status (TTSS) bit is set.</p>

Table 14-24 Transmit Descriptor 7 (TDES7)

Bit	Description
31:0	<p>TTSH: Transmit Frame Time Stamp High</p> <p>This field is updated by DMA with the most significant 32 bits of the time stamp captured for the corresponding receive frame. This field has the time stamp only if the Last Segment bit (LS) in the descriptor is set and Time stamp status (TTSS) bit is set.</p>

Receive Descriptor

The structure of the received descriptor is shown in [Figure 14-12](#). This can have 32 bytes of descriptor data (8 DWORDs) when Advanced Time Stamping or IPC Full Offload feature is selected.

Note: When either of these features is enabled, the SW should set the DMA Bus Mode register[7] so that the DMA operates with extended descriptor size. When this control bit is reset, RDES0[7] and RDES0[0] will be always cleared and the RDES4-RDES7 descriptor space are not valid

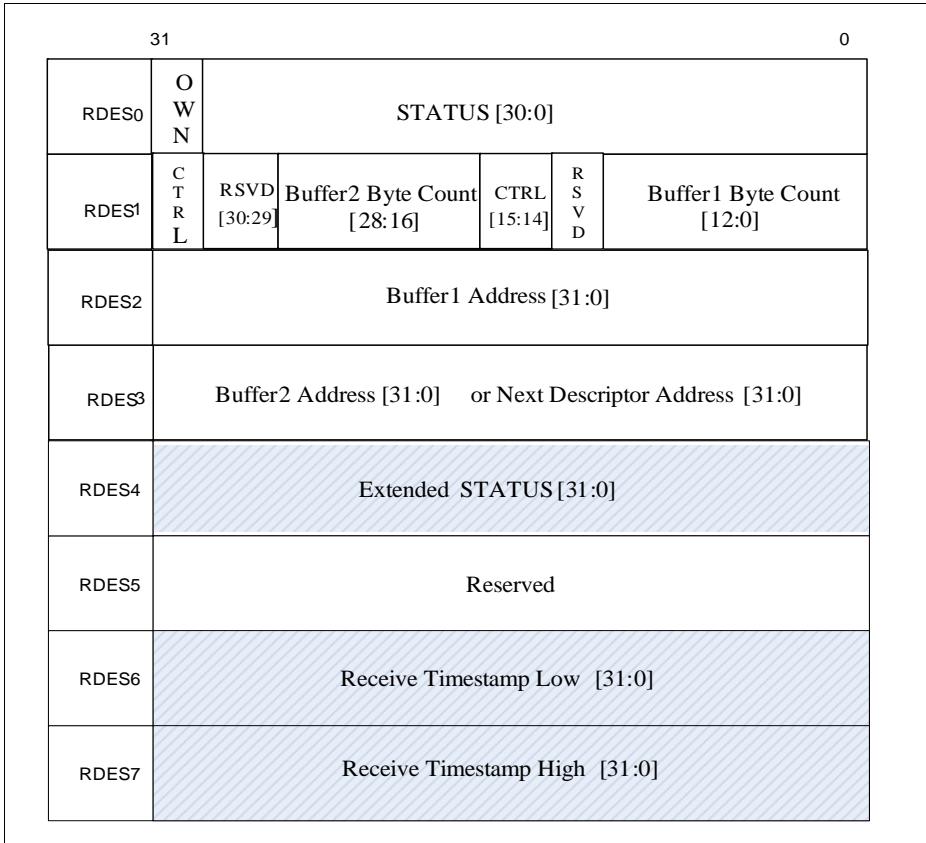


Figure 14-12 Receive Descriptor Fields - Alternate (Enhanced) Format

The contents of RDES0 are identified in [Table 14-25](#). The contents of RDES1 through RDES3 are identified in [Table 14-26](#) through [Table 14-28](#), respectively.

Note: Some of the bit functions of RDES0 are not backward compatible to Release 3.41a and previous versions. These bits are Bit[7], Bit[0] and Bit[5]. The function of Bit[5] is backward compatible to Release 3.30a and previous versions.

Table 14-25 Receive Descriptor Fields (RDES0)

Bit	Description
31	<p>OWN: Own Bit</p> <p>When set, this bit indicates that the descriptor is owned by the DMA of the GMAC Subsystem. When this bit is reset, this bit indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full.</p>
30	<p>AFM: Destination Address Filter Fail</p> <p>When set, this bit indicates a frame that failed in the DA Filter in the GMAC Core.</p>
29:16	<p>FL: Frame Length</p> <p>These bits indicate the byte length of the received frame that was transferred to host memory (including CRC). This field is valid when Last Descriptor (RDES0[8]) is set and either the Descriptor Error (RDES0[14]) or Overflow Error bits are reset. The frame length also includes the two bytes appended to the Ethernet frame when IP checksum calculation (Type 1) is enabled and the received frame is not a MAC control frame.</p> <p>This field is valid when Last Descriptor (RDES0[8]) is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current frame.</p>
15	<p>ES: Error Summary</p> <p>Indicates the logical OR of the following bits:</p> <ul style="list-style-type: none"> • RDES0[1]: CRC Error • RDES0[3]: Receive Error • RDES0[4]: Watchdog Timeout • RDES0[6]: Late Collision • RDES0[7]: Giant Frame • RDES4[4:3]: IP Header/Payload Error • RDES0[11]: Overflow Error • RDES0[14]: Descriptor Error <p>This field is valid only when the Last Descriptor (RDES0[8]) is set.</p>
14	<p>DE: Descriptor Error</p> <p>When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next Descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]) is set.</p>
13	<p>SAF: Source Address Filter Fail</p> <p>When set, this bit indicates that the SA field of frame failed the SA Filter in the GMAC Core.</p>

Table 14-25 Receive Descriptor Fields (RDES0) (cont'd)

Bit	Description
12	<p>LE: Length Error</p> <p>When set, this bit indicates that the actual length of the frame received and that the Length/ Type field does not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset.</p>
11	<p>OE: Overflow Error</p> <p>When set, this bit indicates that the received frame was damaged due to buffer overflow in MTL.</p>
10	<p>VLAN: VLAN Tag</p> <p>When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the GMAC Core.</p>
9	<p>FS: First Descriptor</p> <p>When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame.</p>
8	<p>LS: Last Descriptor</p> <p>When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame</p>
7	<p>Time Stamp Available/IP Checksum Error (Type1) /Giant Frame</p> <p>When Advanced Time Stamp feature is present, this bit, when set, indicates that a snapshot of the timestamp is written in descriptor words 6 (RDES6) and 7 (RDES7). This is valid only when the Last Descriptor bit (RDES0[8]) is set. When IP Checksum Engine (Type 1) is selected, this bit, when set, indicates that the 16-bit IPv4 Header checksum calculated by the core did not match the received checksum bytes.</p> <p>Otherwise, this bit, when set, indicates the Giant Frame Status. Giant frames are larger-than-1,518-byte (or 1,522-byte for VLAN) normal frames and larger-than-9,018-byte (9,022-byte for VLAN) frame when Jumbo Frame processing is enabled.</p>
6	<p>LC: Late Collision</p> <p>When set, this bit indicates that a late collision has occurred while receiving the frame in Half-Duplex mode.</p>
5	<p>FT: Frame Type When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 16'h0600). When this bit is reset, it indicates that the received frame is an IEEE802.3 frame. This bit is not valid for Runt frames less than 14 bytes.</p>

Table 14-25 Receive Descriptor Fields (RDES0) (cont'd)

Bit	Description
4	RWT: Receive Watchdog Timeout When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout.
3	RE: Receive Error When set, this bit indicates that the gmii_rxr_i signal is asserted while gmii_rxdv_i is asserted during frame reception. This error also includes carrier extension error in GMII and Half-duplex mode. Error can be of less/no extension, or error (rxd \neq 0f) during extension.
2	DE: Dribble Bit Error When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in MII Mode.
1	CE: CRC Error When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8]) is set.
0	Extended Status Available/Rx MAC Address When either Advanced Time Stamp or IP Checksum Offload (Type 2) is present, this bit, when set, indicates that the extended status is available in descriptor word 4 (RDES4). This is valid only when the Last Descriptor bit (RDES0[8]) is set. When Advance Time Stamp Feature or IPC Full Offload is not selected, this bit indicates Rx MAC Address status. When set, this bit indicates that the Rx MAC Address registers value (1 to 31) matched the frame's DA field. When reset, this bit indicates that the Rx MAC Address Register 0 value matched the DA field.

Table 14-26 Receive Descriptor Fields 1 (RDES1)

Bit	Description
31	DIC: Disable Interrupt on Completion When set, this bit prevents setting the Status Register's RI bit (CSR5[6]) for the received frame ending in the buffer indicated by this descriptor. This, in turn, disables the assertion of the interrupt to Host due to RI for that frame.
30:29	Reserved

Table 14-26 Receive Descriptor Fields 1 (RDES1) (cont'd)

Bit	Description
28:16	<p>RBS2: Receive Buffer 2 Size</p> <p>These bits indicate the second data buffer size, in bytes. The buffer size must be a multiple of 4, 8, or 16, depending on the bus widths (32, 64, or 128, respectively), even if the value of RDES3 (buffer2 address pointer) is not aligned to bus width. If the buffer size is not an appropriate multiple of 4, 8, or 16, the resulting behavior is undefined. This field is not valid if RDES1[14] is set. See Buffer Size Calculations for further details on calculating buffer sizes.</p>
15	<p>RER: Receive End of Ring</p> <p>When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring.</p>
14	<p>RCH: Second Address Chained</p> <p>When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When this bit is set, RBS2 (RDES1[28:16]) is a “don't care” value. RDES1[15] takes precedence over RDES1[14].</p>
13	<p>Reserved</p>
12:0	<p>RBS1: Receive Buffer 1 Size</p> <p>Indicates the first data buffer size in bytes. The buffer size must be a multiple of 4, 8, or 16, depending upon the bus widths (32, 64, or 128), even if the value of RDES2 (buffer1 address pointer) is not aligned. When the buffer size is not a multiple of 4, 8, or 16, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (Bit 14). See Buffer Size Calculations for further details on calculating buffer sizes.</p>

Table 14-27 Receive Descriptor Fields 2 (RDES2)

Bit	Description
31:0	<p>Buffer 1 Address Pointer</p> <p>These bits indicate the physical address of Buffer 1. There are no limitations on the buffer address alignment except for the following condition: The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. Note that the DMA performs a write operation with the RDES2[3/2/1:0] bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[3/2/1:0] (corresponding to bus width of 128/64/32) if the address pointer is to a buffer where the middle or last part of the frame is stored. See XMC4300 Data Buffer Alignment for further details on buffer address alignment.</p>

Table 14-28 Receive Descriptor Fields 3 (RDES3)

Bit	Description
31:0	<p>Buffer 2 Address Pointer (Next Descriptor Address)</p> <p>These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present.</p> <p>If RDES1[24] is set, the buffer (Next Descriptor) address pointer must be bus width-aligned (RDES3[3, 2, or 1:0] = 0, corresponding to a bus width of 128, 64, or 32. LSBs are ignored internally.) However, when RDES1[24] is reset, there are no limitations on the RDES3 value, except for the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3[3, 2, or 1:0] (corresponding to a bus width of 128, 64, or 32) if the address pointer is to a buffer where the middle or last part of the frame is stored.</p>

The extended status written is as shown in [Table 14-29](#). The extended status is written only when there is status related to IPC or Time Stamp available. The availability of extended status is indicated by bit-0 of RDES0. This status is available only when Advance Time Stamp or IPC Full Offload feature is selected.

Table 14-29 Receive Descriptor Fields 4 (RDES4)

Bit	Description
31:14	Reserved
13	<p>PTP Version</p> <p>When set, indicates that the received PTP message is having the IEEE 1588 version 2 format. When reset, it has the version 1 format. This is valid only if the message type is non-zero. This bit is available only if Advance Time Stamp feature is selected else it is reserved</p>
12	<p>PTP Frame Type</p> <p>When set, this bit that the PTP message is sent directly over Ethernet. When this bit is not set and the message type is non-zero, it indicates that the PTP message is sent over UDP-IPv4 or UDP-IPv6. The information on IPv4 or IPv6 can be obtained from bits 6 and 7. This bit is available only if Advanced Time Stamp feature is selected</p>
11:8	<p>Message Type</p> <p>These bits are encoded to give the type of the message received.</p> <ul style="list-style-type: none"> • 0000: No PTP message received • 0001: SYNC (all clock types) • 0010: Follow_Up (all clock types) • 0011: Delay_Req (all clock types) • 0100: Delay_Resp (all clock types) • 0101: Pdelay_Req (in peer-to-peer transparent clock) • 0110: Pdelay_Resp (in peer-to-peer transparent clock) • 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) • 1000: Announce • 1001: Management • 1010: Signaling • 1011-1111: Reserved <p>These bits are valid only when you select the Advance Time Stamp feature.</p>
7	<p>IPv6 Packet Received</p> <p>When set, this bit indicates that the received packet is an IPv6 packet.</p>
6	<p>IPv4 Packet Received</p> <p>When set, this bit indicates that the received packet is an IPv4 packet.</p>
5	<p>IP Checksum Bypassed</p> <p>When set, this bit indicates that the checksum offload engine is bypassed.</p>

Table 14-29 Receive Descriptor Fields 4 (RDES4) (cont'd)

Bit	Description
4	<p>IP Payload Error</p> <p>When set, this bit indicates that the 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) that the core calculated does not match the corresponding checksum field in the received segment. It is also set when the TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field.</p>
3	<p>IP Header Error</p> <p>When set, this bit indicates either that the 16-bit IPv4 header checksum calculated by the core does not match the received checksum bytes, or that the IP datagram version is not consistent with the Ethernet Type value.</p>
2:0	<p>IP Payload Type</p> <p>These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE). The COE also sets these bits to 2'b00 if it does not process the IP datagram's payload due to an IP header error or fragmented IP.</p> <ul style="list-style-type: none"> • 3'b000: Unknown or did not process IP payload • 3'b001: UDP • 3'b010: TCP • 3'b011: ICMP • 3'b1xx: Reserved

RDES6 and RDES7 contain the snapshot of the time-stamp. The availability of the snapshot of the time-stamp in RDES6 and RDES7 is indicated by bit-7 in the RDES0 descriptor. The contents of RDES6 and RDES7 are identified in [Table 14-30](#) and [Table 14-31](#), respectively.

Table 14-30 Receive Descriptor Fields 6 (RDES6)

Bit	Description
31:0	<p>RTSL: Receive Frame Time Stamp Low</p> <p>This field is updated by DMA with the least significant 32 bits of the time stamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]).</p>

Table 14-31 Receive Descriptor Fields 7 (RDES7)

Bit	Description
31:0	<p>RTSH: Receive Frame Time Stamp High</p> <p>This field is updated by DMA with the most significant 32 bits of the time stamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]).</p>

14.2.5 MAC Management Counters

The MMC module maintains a set of registers for gathering statistics on the received and transmitted frames. These include a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the MAC Control Interface (MCI). Each register is 32 bits wide. Non-32-bit accesses are allowed as long as the address is word-aligned.

The organization of these registers is shown in [Table 14-40](#). The MMCs are accessed using transactions, in the same way the CSR address space is accessed. The following sections in the chapter describe the various counters and list the address for each of the statistics counters. This address will be used for Read/Write accesses to the desired transmit/receive counter.

The Receive MMC counters are updated for frames that are passed by the Address Filter (AFM) block. Statistics of frames that are dropped by the AFM module are not updated unless they are runt frames of less than 6 bytes (DA bytes are not received fully).

The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet frames. The address map of the corresponding registers, 0200_H–02FC_H, is given in [Table 14-40](#).

14.2.6 Power Management Block

This section describes the power management (PMT) mechanisms supported by the ETH. PMT supports the reception of network (remote) wake-up frames and Magic Packet frames. PMT does not perform the clock gate function, but generates interrupts for wake-up frames and Magic Packets received by the ETH. The PMT block sits on the receiver path of the ETH and is enabled with remote wake-up frame enable and Magic Packet enable. These enables are in the PMT_CONTROL_STATUS register and are programmed by the Application.

PMT registers are accessed in the same manner as with ETH-CSR registers. Refer to [Figure 14-40](#) for mapping information.

When the power down mode is enabled in the PMT, then all received frames are dropped by the core and they are not forwarded to the application. The core comes out of the power down mode only when either a Magic Packet or a Remote Wake-up frame is received and the corresponding detection is enabled.

14.2.6.1 PMT Block Description

PMT Control and Status Register

The PMT CSR program the request wake-up events and monitor the wake-up events. See [ETH0_PMT_CONTROL_STATUS](#) register for a full description

Remote Wake-Up Frame Filter Register

The Remote Wake up Frame Filter consists of eight words which are programmed via the [ETH0_REMOTE_WAKE_UP_FRAME_FILTER](#) Register. The eight words of the Remote Wake Up Frame Filter must be written sequentially to the REMOTE_WAKE_UP_FRAME_FILTER Register. The structure of the Remote Wake Up Frame Filter is described below. The Remote Wake Up Frame Filter values must be loaded sequentially starting with wkupfilter0 through to wkupfilter7. The REMOTE_WAKE_UP_FRAME_FILTER Register is read in the same way.

Note: The internal counter to access the appropriate wkupfilter_reg is incremented when lane3 is accessed by the CPU. This should be kept in mind if you are accessing these registers in byte or half-word mode.

Wake-up frame filter reg0	Filter 0 Byte Mask							
Wake-up frame filter reg1	Filter 1 Byte Mask							
Wake-up frame filter reg2	Filter 2 Byte Mask							
Wake-up frame filter reg3	Filter 3 Byte Mask							
Wake-up frame filter reg4	RSVD	Filter 3 Command	RSVD	Filter 2 Command	RSVD	Filter 1 Command	RSVD	Filter 0 Command
Wake-up frame filter reg5	Filter 3 Offset		Filter 2 Offset		Filter 1 Offset		Filter 0 Offset	
Wake-up frame filter reg6	Filter 1 CRC - 16				Filter 0 CRC - 16			
Wake-up frame filter reg7	Filter 3 CRC - 16				Filter 2 CRC - 16			

Figure 14-13 Wake-Up Frame Filter Register

Filter i Byte Mask

This register defines which bytes of the frame are examined by filter i (0, 1, 2, and 3) in order to determine whether or not the frame is a wake-up frame. The Most Significant Bit (thirty-first bit) must be zero. Bit j [30:0] is the Byte Mask. If bit j (byte number) of the Byte Mask is set, then Filter i Offset + j of the incoming frame is processed by the CRC block; otherwise Filter i Offset + j is ignored.

Filter i Command

This 4-bit command controls the filter i operation. Bit 3 specifies the address type, defining the pattern's destination address type. When the bit is set, the pattern applies to only multicast frames; when the bit is reset, the pattern applies only to unicast frame. Bit 2 and Bit 1 are reserved. Bit 0 is the enable for filter i; if Bit 0 is not set, filter i is disabled.

Filter i Offset

This register defines the offset (within the frame) from which the frames are examined by filter i. This 8-bit pattern-offset is the offset for the filter i first byte to be examined. The minimum allowed is 12, which refers to the 13th byte of the frame (offset value 0 refers to the first byte of the frame).

Filter i CRC-16

This register contains the CRC_16 value calculated from the pattern, as well as the byte mask programmed to the wake-up filter register block.

14.2.6.2 Remote Wake-Up Frame Detection

When the ETH is in sleep mode and the remote wake-up bit is enabled in PMT Control and Status register, normal operation is resumed after receiving a remote wake-up frame. The Application writes all eight wake-up filter registers, by performing a sequential Write to **ETH0_REMOTE_WAKE_UP_FRAME_FILTER** Register. The Application enables remote wake-up by setting **ETH0_PMT_CONTROL_STATUS.PWRDWN**.

PMT supports four programmable filters that allow support of different receive frame patterns. If the incoming frame passes the address filtering of Filter Command, and if Filter CRC-16 matches the incoming examined pattern, then the wake-up frame is received.

Filter_offset (minimum value 12, which refers to the 13th byte of the frame) determines the offset from which the frame is to be examined. Filter Byte Mask determines which bytes of the frame must be examined. The thirty-first bit of Byte Mask must be set to zero.

The remote wake-up CRC block determines the CRC value that is compared with Filter CRC-16. The wake-up frame is checked only for length error, FCS error, dribble bit error, MII error, collision, and to ensure that it is not a runt frame. Even if the wake-up frame is more than 512 bytes long, if the frame has a valid CRC value, it is considered valid. Wake-up frame detection is updated in the PMT_Control_Status register for every remote Wake-up frame received. A PMT interrupt to the Application triggers a Read to the PMT_CONTROL_STATUS register to determine reception of a wake-up frame.

14.2.6.3 Magic Packet Detection

The Magic Packet frame is based on a method that uses Advanced Micro Device's Magic Packet technology to power up the sleeping device on the network. The ETH receives a specific packet of information, called a Magic Packet, addressed to the node on the network.

Only Magic Packets that are addressed to the device or a broadcast address will be checked to determine whether they meet the wake-up requirements. Magic Packets that pass the address filtering (unicast or broadcast) will be checked to determine whether they meet the remote Wake-on-LAN data format of 6 bytes of all ones followed by a ETH Address appearing 16 times.

The application enables Magic Packet wake-up by writing a 1 to Bit 1 of the **ETH0_PMT_CONTROL_STATUS** register. The PMT block constantly monitors each frame addressed to the node for a specific Magic Packet pattern. Each frame received is checked for a FFFF FFFF FFFF_H pattern following the destination and source address field. The PMT block then checks the frame for 16 repetitions of the ETH address without

any breaks or interruptions. In case of a break in the 16 repetitions of the address, the FFFF FFFF FFFF_H pattern is scanned for again in the incoming frame. The 16 repetitions can be anywhere in the frame, but must be preceded by the synchronization stream (FFFF FFFF FFFF_H). The device will also accept a multicast frame, as long as the 16 duplications of the ETH address are detected.

If the MAC address of a node is 0011 2233 4455_H, then the ETH scans for the data sequence:

```
Destination Address Source Address ..... FF FF FF FF FF FF
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33
44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33
44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33
44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33
44 55
...CRC
```

Magic Packet detection is updated in the PMT_CONTROL_STATUS register for Magic Packet received. A PMT interrupt to the Application triggers a read to the PMT CSR to determine whether a Magic Packet frame has been received.

14.2.6.4 System Considerations During Power-Down

The ETH neither gates nor stops clocks when Power-Down mode is enabled. Power saving by clock gating must be done outside the core by the application. The receive data path must be clocked during Power-Down mode, because it is involved in magic packet/wake-on-LAN frame detection. However, the transmit path and the application path clocks can be gated off during Power-Down mode.

The power management interrupt signal is asserted when a valid wake-up frame is received. This signal is generated in the receive clock domain.

The recommended power-down and wake-up sequence is as follows.

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. These transmissions can be detected when Transmit Interrupt, **ETH0_STATUS.TI** is received.
2. Disable the MAC transmitter and MAC receiver by clearing the appropriate bits in the **ETH0_MAC_CONFIGURATION** register.
3. Wait until the Receive DMA empties all the frames from the Rx FIFO (a software timer may be required).
4. Enable Power-Down mode by appropriately configuring the PMT registers.
5. Enable the MAC Receiver and enter Power-Down mode.

6. Gate the application and transmit clock inputs to the core (and other relevant clocks in the system) to reduce power and enter Sleep mode.
7. On receiving a valid wake-up frame, the ETH asserts the power management interrupt signal and exits Power-Down mode.
8. On receiving the interrupt, the system must enable the application and transmit clock inputs to the core.
9. Read the **ETH0_PMT_CONTROL_STATUS** register to clear the interrupt, then enable the other modules in the system and resume normal operation.

14.2.7 PHY Interconnect

The ETH supports two external interconnects to external PHY devices. The ETH peripheral may be connected to the external PHY by either a Media Independent Interface (MII) or by a Reduced Media Independent Interface (RMII). Additionally a Station Management Interface (SMI) provides a two wire serial interface between the external PHY and the ETH. The SMI allows the ETH to program the internal PHY configuration registers. The SMI supports connection of up to 32 external PHY devices.

14.2.7.1 PHY Interconnect selection

Selection of the external interconnect configuration between MII or RMII is made by the ETH0_CON.INFSEL register. This must be done while the ETH peripheral is held in reset. Once the PHY interconnect has been configured it may not be changed without placing the ETH back into reset.

14.2.8 Station Management Interface

The Station Management Agent (SMA) module allows the Application to access any PHY registers through a 2-wire Station Management interface (SMI). The PHY interconnect supports accessing up to 32 PHYs.

The application can select one of the 32 PHYs and one of the 32 registers within any PHY and send control data or receive status information. Only one register in one PHY can be addressed at any given time. For more details on the communication from the Application to the PHYs, refer to the Reconciliation Sublayer and Media Independent Interface Specifications section of the IEEE 802.3z specification, 1000BASE Ethernet. The application sends the control data to the PHY and receives status information from the PHY through the SMA module, as shown in **Figure 14-14**.

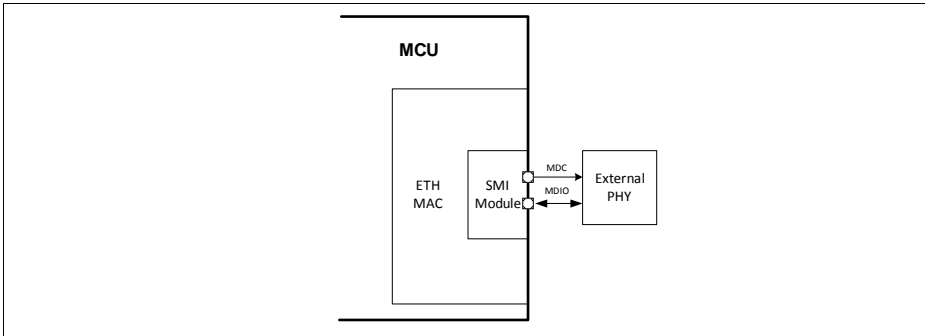


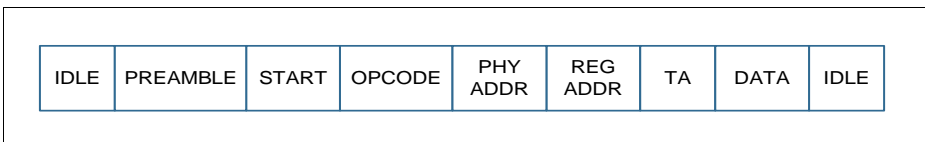
Figure 14-14 SMA Interface Block

14.2.8.1 Station Management Functions

The ETH initiates the Management Write/Read operation. The MDC clock is a divided clock from the ETH MAC clock. The divide factor depends on the clock range setting in the MII Address register. Clock range is set as follows:

Selection	ETH MAC Clock	MDC Clock
0000	60-100 MHz	ETH Clock/42
0001	100-150 MHz	ETH Clock/62
0010	20-35 MHz	ETH Clock/16
0011	35-60 MHz	ETH Clock/26
0100	150-250 MHz	ETH Clock/102
0101	250-300 MHz	ETH Clock/124
0110, 0111	Reserved	

The frame structure on the MDIO line is shown below.



IDLE The mdio line is Tri-state; there is no clock on mdc
PREAMBLE 32 continuous bits of value 1
START Start-of-frame is 01_b
OPCODE 10_b for Read and b01_b for Write
PHY ADDR 5-bit address select for one of 32 PHYs

REG_ADDR Register address in the selected PHY
 TA Turnaround is $Z0_B$ for Read and 10_B for Write
 DATA Any 16-bit value. In a Write operation, the ETH drives mdio; in a Read operation, PHY drives it.

14.2.8.2 Station Management Write Operation

When the user sets the MII Write and Busy bits (**ETH0_GMII_ADDRESS.MW** and **GMII_ADDRESS.MB**) the ETH CSR module transfers the PHY address, the register address in PHY, and the write data to the SMA to initiate a Write operation into the PHY registers. At this point, the SMA module starts a Write operation on the MII Management Interface using the Management Frame Format specified in the MII specifications (Section 22.2.4.5 of IEEE Standard). The application should not change the **GMII_ADDRESS** register contents or the **GMII_DATA** register while the transaction is ongoing. Write operations to the **GMII_ADDRESS** register or the **ETH0_GMII_DATA** Register during this period are ignored (the Busy bit is high), and the transaction is completed without any error on the MCI interface.

After the Write operation has completed, the SMA indicates this to the CSR which then resets the Busy bit. The SMA module divides the CSR (Application) clock with the clock divider programmed (CR bits of MII Address Register) to generate the MDC clock for this interface. The ETH drives the MDIO line for the complete duration of the frame. The frame format for the Write operation is as follows:

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111...11	01	01	AAAAA	RRRRR	10	DDD... .DDD	Z

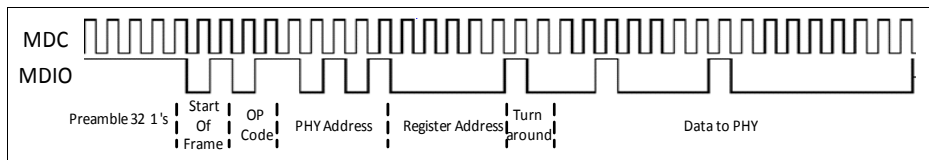


Figure 14-15 Management Write Operation

Figure 14-15 is a reference for the Write operation.

14.2.8.3 Station Management Read Operation

When the user sets the MII Busy bit (**ETH0_GMII_ADDRESS.MB**) with the MII Write bit (**GMII_ADDRESS.WB**) as 0, the ETH CSR module transfers the PHY address and the

register address in PHY to the SMA to initiate a Read operation in the PHY registers. At this point, the SMA module starts a Read operation on the MII Management Interface using the Management Frame Format specified in the MII specifications (Section 22.2.4.5 of IEEE Standard). The application should not change the GMII_ADDRESS register contents or the GMII_DATA register while the transaction is ongoing. Write operations to the GMII_ADDRESS register or **ETH0_GMII_DATA** Register during this period are ignored (the Busy bit is high) and the transaction completed without any error on the MCI interface.

After the Read operation has completed, the SMA indicates this to the CSR, which then resets the Busy bit and updates the GMII_DATA register with the data read from the PHY. The SMA module divides the CSR (Application) clock with the clock divider programmed (GMII_ADDRESS.CR bits) to generate the MDC clock for this interface. The ETH drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line. The frame format for the Read operation is as follows:

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111...11	01	10	AAAAA	RRRRR	Z0	DDDDDD	Z

Figure 14-16 is a reference for the read operation.

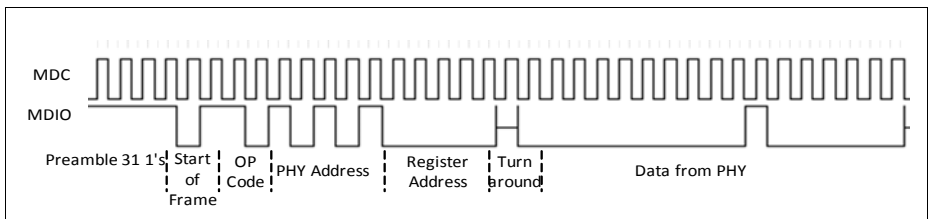


Figure 14-16 Management Read Operation

14.2.9 Media Independent interface

The Media Independent Interface (MII) provides an interconnect to external PHY devices standardised by IEEE 802.3u. The MII interconnect consists of 16 pins for data and control. The MII interconnect provides two separate nibble wide busses for transmit and receive each with a dedicated clock running at 2.5Mhz for 10Mbit/s and 25Mhz for 100Mbit/s speeds. Transmit and receive control signals consist of a TX Enable (TX_EN) that allows the ETH to present data to the PHY and a Receive Data Valid (RX_DV) that allows the PHY to present data to the ETH. A Receive Error (RX_ER) signal is also provided that allows the PHY signal the ETH when an error was detected somewhere in

the current received packet. The two remaining control signals are MII collision detect (MII_COL) which is asserted by the PHY when an arbitration collision occurs and MII carrier sense (MII_CR_S) which is asserted by the PHY when either Transmit or Receive are not idle.

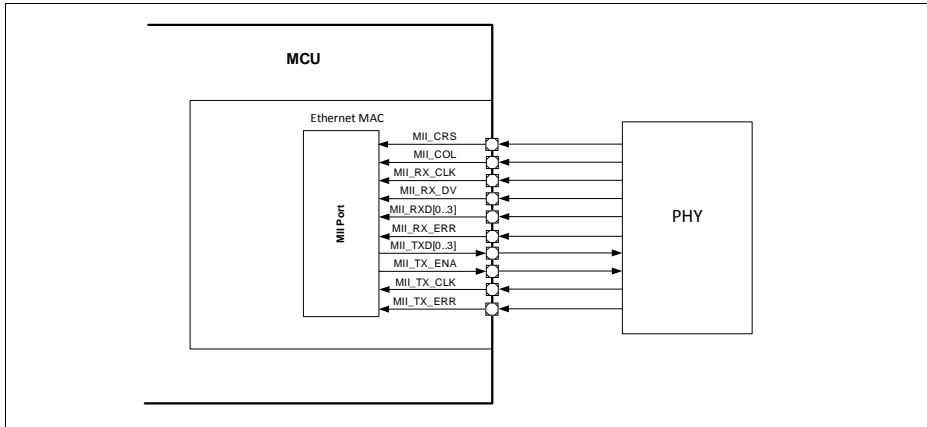


Figure 14-17 Media Independent Interface

14.2.10 Reduced Media Independent Interface

The Reduced Media Independent Interface (RMII) specification reduces the pin count between Ethernet PHYs and Switch ASICs. According to the IEEE 802.3u standard, an MII contains 16 pins for data and control. In devices incorporating multiple MAC or PHY interfaces (such as switches), the number of pins adds significant cost with increase in port count. The RMII specification addresses this problem by reducing the pin count to 7 for each port — a 62.5% decrease in pin count.

- The RMII module is instantiated between the ETH and the PHY. This helps translation of the MAC's MII into the RMII. The RMII block has the following characteristics:
- Supports 10 Mbit/s and 100 Mbit/s operating rates.
- Two clock references are sourced externally, providing independent, 2-bit wide transmit and receive paths.

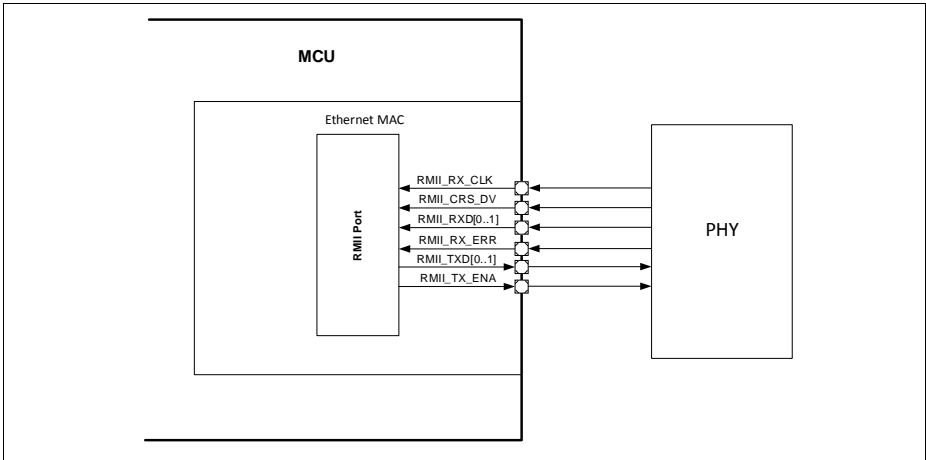


Figure 14-18 Reduced Media Independent Interface

Note: The MAC Configuration.FES bit configures the RMII to operate at 10 Mbit/s or 100 Mbit/s.

14.2.10.1 RMII Block Diagram

Figure 14-19 shows the position of the RMII block relative to the ETH and RMII PHY. The RMII block is placed in front of the ETH to translate the MII signals to RMII signals.

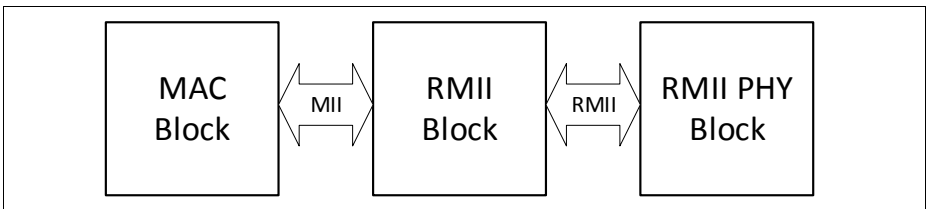


Figure 14-19 RMII Block Diagram

14.2.10.2 RMII Block Overview

The following list describes the RMII's hardware components, which are shown in **Figure 14-19**. Each of these blocks is briefly described in the following sections.

MII-RMII Transmit (MRT) Block: This block translates all MII transmit signals to RMII transmit signals.

MII-RMII Receive (MRR) Block: This block translates all RMII receive signals to MII receive signals.

14.2.10.3 Transmit Bit Ordering

Each nibble from the MII must be transmitted on the RMIIL a di-bit at a time with the order of di-bit transmission shown in [Figure 14-20](#). The lower order bits (D1 and D0) are transmitted first followed by higher order bits (D2 and D3).

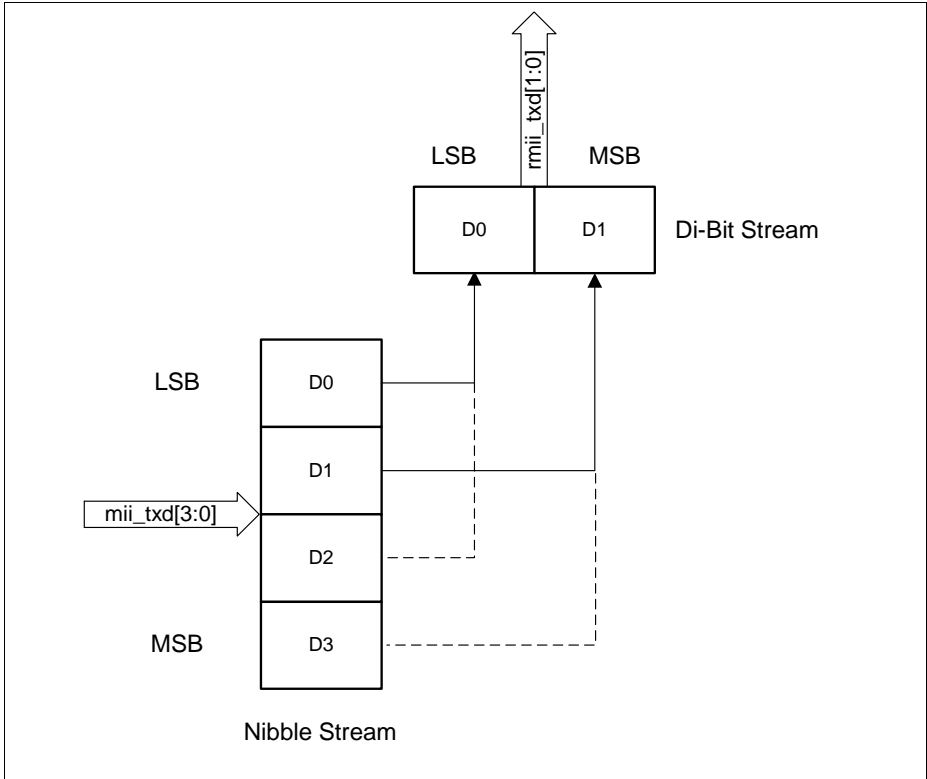


Figure 14-20 Transmission Bit Ordering

14.2.10.4 RMII Transmit Timing Diagrams

[Figure 14-21](#) through [Figure 14-24](#) show MII-to-RMII transaction timing.

[Figure 14-21](#) shows the start of MII transmission and the following RMII transmission in 100 Mbit/s mode.

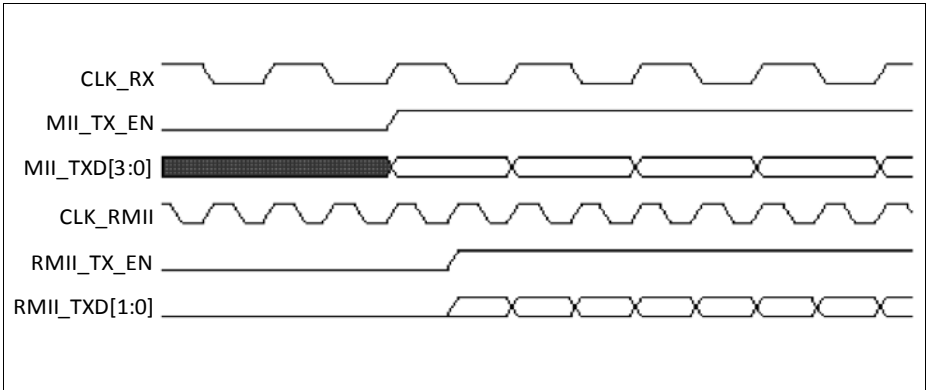


Figure 14-21 Start of MII and RMII Transmission in 100 Mbit/s Mode

Figure 14-22 shows the end of frame transmission for MII and RMII in 100 Mbit/s mode.

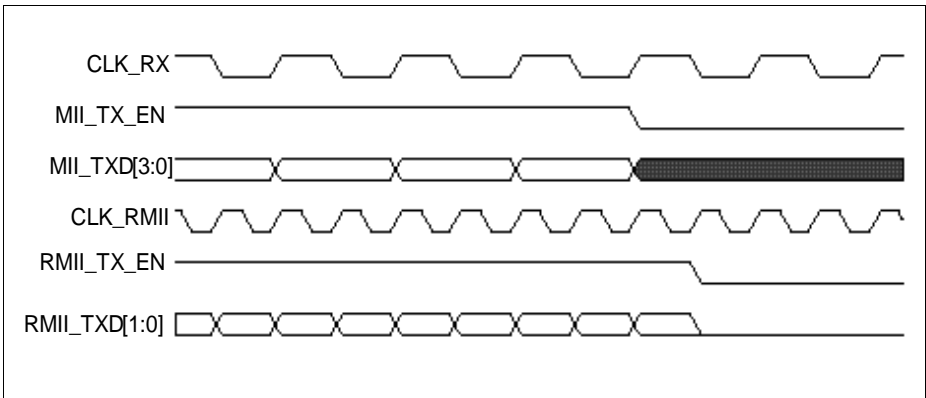


Figure 14-22 End of MII and RMII Transmission in 100 Mbit/s Mode

Figure 14-23 shows the start of MII transmission and the following RMII transmission in 10 Mbit/s mode.

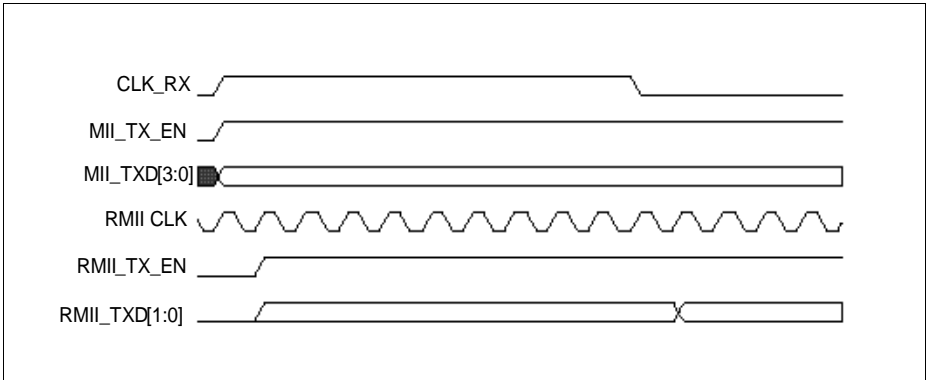


Figure 14-23 Start of MII and RMII Transmission in 10 Mbit/s Mode

Figure 14-24 shows the end of MII transmission and RMII transmission in 10 Mbit/s mode.

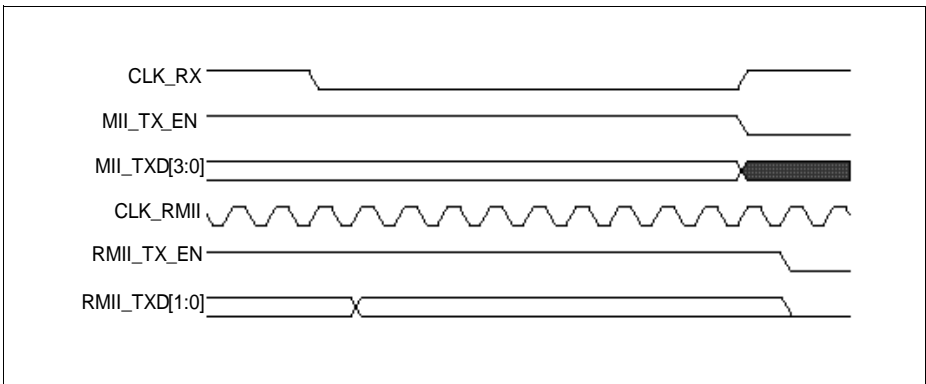


Figure 14-24 End of MII and RMII Transmission in 10 Mbit/s Mode

Receive Bit Ordering

Each nibble is transmitted to the MII from the di-bit received from the RMII in the nibble transmission order shown in **Figure 14-25**. The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

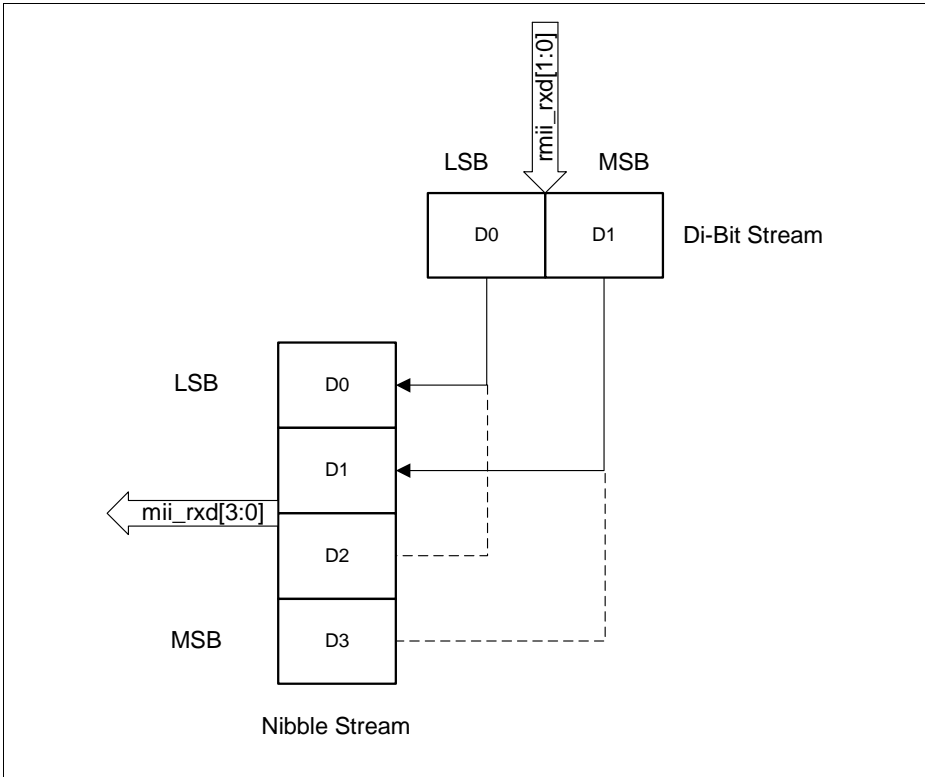


Figure 14-25 Receive Bit Ordering

14.2.11 IEEE 1588-2002 Overview

The IEEE 1588 standard defines a protocol enabling precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The protocol applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

The message-based protocol, named Precision Time Protocol (PTP), is transported over UDP/IP. The system or network is classified into Master and Slave nodes for distributing the timing/clock information. The protocol's technique for synchronizing a slave node to a master node by exchanging PTP messages is depicted in [Figure 14-26](#).

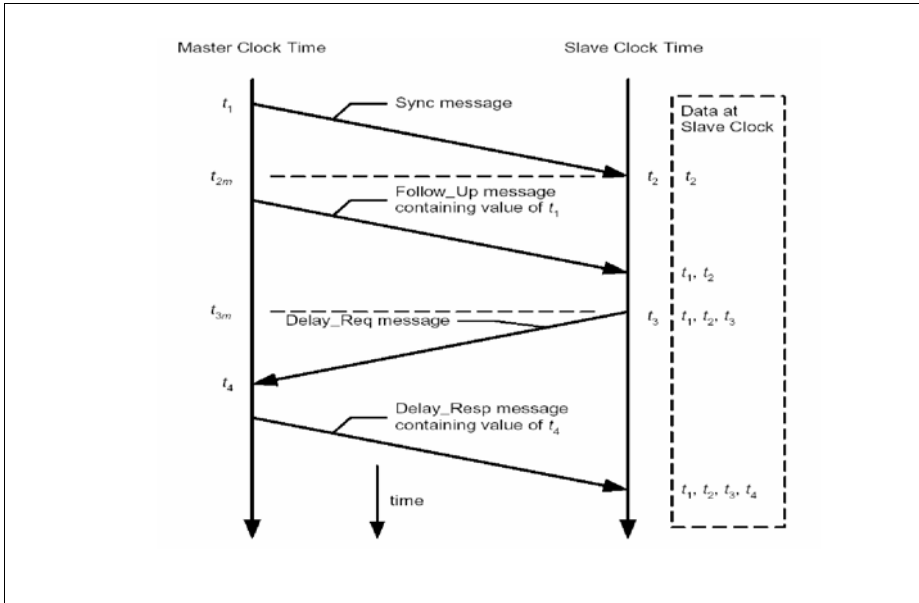


Figure 14-26 Networked Time Synchronization

1. The master broadcasts PTP Sync messages to all its nodes. The Sync message contains the master's reference time information. The time at which this message leaves the master's system is t_1 and must, for Ethernet ports, be captured at the MII.
2. A slave receives the Sync message and also captures the exact time, t_2 , using its timing reference.
3. The master then sends the slave a Follow_up message, which contains t_1 information for later use.
4. The slave sends the master a Delay_Req message, noting the exact time, t_3 , at which this frame leaves the MII.
5. The master receives this message, capturing the exact time, t_4 , at which it enters its system.
6. The master sends the t_4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t_1, t_2, t_3 , and t_4 to synchronize its local timing reference to the master's timing reference.

Most of the protocol implementation occurs in the software, above the UDP layer. As described above, however, hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the MII. This timing information must be captured and returned to the software for the proper implementation of PTP with high accuracy.

14.2.11.1 Reference Timing Source

To get a snapshot of the time, the core requires a reference time in 64-bit format (split into two 32-bit channels, with the upper 32-bits providing time in seconds, and the lower 32-bits indicating time in nanoseconds) as defined in the IEEE 1588 specification.

Internal Reference Time

This takes only the reference clock input and uses it to generate the Reference time (also called the System Time) internally and use it to capture time stamps. The generation, update, and modification of the System Time are described in [System Time Register Module](#).

14.2.11.2 Transmit Path Functions

When a frame's SFD is output on the MII, a time stamp is captured. Frames for which capturing a time stamp is required are controllable on a per-frame basis. In other words, each transmit frame can be marked to indicate whether or not a time stamp must be captured for that frame.

No snooping or processing of the transmitted frames is performed to identify PTP frames. Framewise control is exercised through control bits in the transmit descriptor (as described in [Descriptor Format With IEEE 1588 Time Stamping Enabled](#)).

Captured time stamps are returned to the application in a manner similar to that in which status is provided for frames. time stamp is returned to software inside the corresponding transmit descriptor, thus connecting the time stamp automatically to the specific PTP frame. The 64-bit time stamp information is written back to the TDES2_{RAM} and TDES3_{RAM} fields, with TDES2 holding the time stamp's 32 least significant bits, except as described in [Transmit Time Stamp Field](#).

Note: When the alternate (enhanced) descriptor is selected, the 64-bit time-stamp is written in TDES6_{RAM} and TDES7_{RAM}, respectively

14.2.11.3 Receive Path Functions

When the IEEE 1588 time-stamping feature is selected and enabled, the Ethernet MAC captures the time stamp of all frames received on the MII. No snooping or processing of the received frames is performed to identify PTP frames in the default mode (Advanced Time Stamp feature is not selected).

The core returns the time-stamp to the software in the corresponding receive descriptor. The 64-bit time stamp information is written back to the RDES2 and RDES3 fields, with RDES2 holding the time stamp's 32 least significant bits, except as mentioned in [Receive Time Stamp](#). The time stamp is only written to the receive descriptor for which the Last Descriptor status field has been set to 1 (the EOF marker). When the time stamp is not available (for example, due to an RxFIFO overflow) an all-ones pattern is written

to the descriptors (RDES2 and RDES3), indicating that time stamp is not correct. If the software uses a control register bit to disable time stamping, the DMA does not alter RDES2 or RDES3.

Note: When the alternate (enhanced) descriptor is selected, the 64-bit time-stamp is written in RDES6 and RDES7, respectively. RDES0[7] will indicate whether the time-stamp is updated in RDES6/7 or not.

14.2.11.4 Time Stamp Error Margin

According to the IEEE 1588 specifications, the time stamp must be captured at the SFD of transmitted and received frames at the MII interface. Since the reference timing source is different from the MII clocks, a small error margin is introduced, due to the transfer of information across asynchronous clock domains.

In the transmit path, the captured and reported time stamp has a maximum error margin of 2 PTP clocks. In other words, the captured time stamp has the value of the reference time source given within 2 clocks after the SFD has been transmitted on the MII.

Similarly, on the receive path, the error margin is 3 MII clocks, plus up to 2 PTP clocks. You can ignore the error margin due to the 3 MII clocks by assuming that this constant delay is present in the system (or link) before the SFD data reaches the ETH's MII interface.

14.2.11.5 Frequency Range of Reference Timing Clock

Because asynchronous logic is in place for time stamp information transfers across clock domain, a minimum delay is required between two consecutive time stamp captures. This delay is 3 clock cycles of both the MII and PTP clocks. If the gap is shorter, the ETH does not take a time stamp snapshot for the second frame.

The maximum PTP clock frequency is limited by the maximum resolution of the reference time and the timing constraints achievable for logic operating on the PTP clock. Another factor to consider is that the resolution, or granularity, of the reference time source determines the accuracy of the synchronization. Hence, a higher PTP clock frequency gives better system performance. The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes. Because the MII clock frequency is fixed by IEEE specification, the minimum PTP clock frequency required for proper operation depends on the core's operating mode and operating speed.

For example, in 100 Mbit/s full-duplex operation, the minimum gap between two SFDs is 160 MII clocks (128 clocks for a 64-byte frame + 24 clocks of min IFG + 8 clocks of preamble).

In the example, $(3 \times \text{PTP}) + (3 \times \text{MII}) \leq 160 \times \text{MII}$; thus, the minimum PTP clock frequency is about 0.5 MHz $((160 - 3) \times 40 \text{ ns} \div 3 = 2.093 \text{ ns period})$

14.2.11.6 Advanced Time Stamp Feature Support

In addition to the basic features for time stamp mentioned in [Receive Time Stamp](#), the advanced time stamp option has the following features.

- Support for the IEEE 1588-2008 (Version 2) timestamp format.
- Option to take snapshot for all frames or for PTP type frames.
- Option for taking snapshot for event messages only.
- Option to take the snapshot based on the clock type (ordinary, boundary, end-to-end and peer-to-peer)
- Option to select the node to be a Master or Slave for ordinary and boundary clock.
- Identification of PTP message type, version, and PTP payload sent directly over Ethernet given as status.
- Option to measure time in digital or binary format.

14.2.11.7 Peer-to-Peer PTP (Pdelay) Transparent Clock (P2P TC) Message Support

The IEEE 1588-2008 version supports Pdelay message in addition to SYNC, Delay Request, Follow-up and Delay Response messages. [Figure 14-27](#) shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

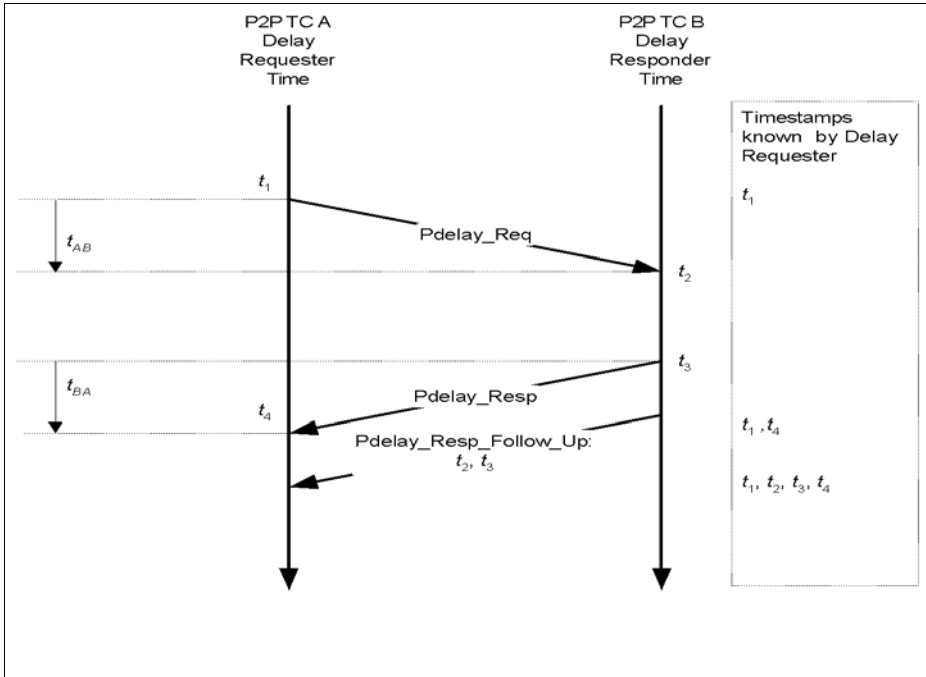


Figure 14-27 Propagation Delay Calculation in Clocks Supporting Peer-to-Peer Path Correction

The link delay measurement starts with port-1 issuing a “Pdelay_Req” message and generating a timestamp, for the Pdelay_Req message. Port-2 receives the “Pdelay_Req” message and generates a timestamp, t₂, for this message. Port-2 returns a Pdelay_Resp message and generates a timestamp, t₃, for this message. To minimize errors due to any frequency offset between the two ports, Port-2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message.

Port-2 either:

- Returns the difference between the timestamps t₂ and t₃ in the Pdelay_Resp message,
- Returns the difference between the timestamps t₂ and t₃ in a Pdelay_Resp_Follow_Up message, or
- Returns the timestamps t₂ and t₃ in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages respectively.

Port-1 generates a timestamp, t₄, upon receiving the Pdelay_Resp message. Port-1 then uses these four timestamps to compute the mean link delay.

14.2.11.8 Clock Types

The type of clock nodes supported in IEEE 1588-2008 is described in this section. The corresponding support provided by the advanced time stamp feature for each of the clock type is also mentioned.

1. Ordinary clock support: In this type the clock can be a grandmaster or a slave clock. This clock has a single PTP state.

Table 14-32 shows the messages for which time-stamp snapshot is taken on the receive side for Master and Slave nodes.

The ordinary clock in the domain supports a single copy of the protocol and has a single PTP state and will typically be a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as sensors and actuators. In telecom applications, the ordinary clock may be associated with a timing demarcation device.

Typically for ordinary clock, you will need to take snapshot for only one type of PTP messages. For e.g. you will require supporting either version 1 or 2 PTP messages, not both.

The following features are supported.

- a) Sends and receives PTP messages. The time stamp snapshot can be controlled as described by the **ETH0_TIMESTAMP_CONTROL** Register.
 - b) Maintains the data sets (e.g., time stamp values).
2. Boundary clock support: This type of clock is similar to the ordinary clock except for the following.

Hence the features of ordinary clock holds good for the boundary clock also.

The boundary clock typically has several physical ports communicating with the network. The messages related to synchronization, master-slave hierarchy and signaling terminate in the protocol engine of the boundary clock and are not forwarded. The PTP message type status given by the core (refer to **Receive Path Functions**) will help you to quickly identify the type of message and take appropriate action.

- a) The clock data sets are common to all ports of the boundary clock
 - b) The local clock is common to all ports of the boundary clock.
3. End to end transparent clock support: The end-to-end transparent clock forwards all messages like normal bridge, router or repeater. The residence time needs to be computed to update the correctionField. Hence snapshot needs to be taken for the messages mentioned in **Table 14-33**.

In the end-to-end transparent clock, the residence times are accumulated in a special field (correctionField) of the PTP event (SYNC) message or the associated Follow-up (FOLLOW_UP) Message. Hence it is important to take a snapshot for these messages alone. This can be quickly done by setting the control bit (TSEVNTENA), which enables snapshot to be taken for event messages and also selecting the type of clock in the **ETH0_TIMESTAMP_CONTROL** Register.

The residence time is also corrected for Delay_Req messages (but snapshot of the

timestamp is not required). The message type statuses provided helps you to quickly identify the message and update the correctionField.

The message type status provided will also help in taking appropriate action depending on the type of PTP message received.

4. Peer to peer transparent clock support: In this type of clock the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp and Pdelay_Resp_Follow_Up messages with the link peer. Hence support for taking snapshot for the event messages related to Pdelay is added. **Table 14-34**.

The transparent clock corrects only the SYNC and Follow-up message. As discussed earlier this can be achieved using the message status provided.

The type of clock to be implemented will be configurable through **ETH0_TIMESTAMP_CONTROL** register. To ensure that the snapshot is taken only for the messages indicated in the table for the corresponding clock type, the **ETH0_TIMESTAMP_CONTROL.TSEVNTENA** bit has to be set.

Table 14-32 PTP Messages for which Snapshot is Taken on Receive Side for Ordinary Clock

Master	Slave
Delay_Req	SYNC

Table 14-33 PTP Messages for which Snapshot is Taken for Transparent Clock Implementation

SYNC
FOLLOW_UP

Table 14-34 PTP Messages for which Snapshot is Taken for Peer-to-Peer Transparent Clock Implementation

SYNC
Pdelay_Req
Pdelay_Resp

14.2.11.9 PTP Processing and Control

The common message header for PTP messages is shown below. This format is taken from IEEE standard 1588-2008 (Revision of IEEE Std. 1588-2002).

Table 14-35 Message Format Defined in IEEE 1588-2008

BITS		OCTETS	OFFSET
transportSpecific	messageType	1	0
Reserved	versionPTP	1	1
messageLength		2	2
domainNumber		1	4
Reserved		1	5
flagField		2	6
correctionField		8	8
Reserved		4	16
sourcePortIdentity		10	20
sequenceId		2	30
controlField ¹⁾		1	32
logMessageInterva		1	33

1) controlField is used in version 1. For version 2, messageType field will be used for detecting different message types.

There are some fields in the PTP frame that are used to detect the type and control the snapshot to be taken. This is different for PTP frames sent directly over Ethernet, PTP frames sent over UDP / IPv4 and PTP frames that are sent over UDP / IPv6. The following sections provide information on the fields that are used to control taking the snapshot.

PTP Frame Over IPv4

Table 14-36 gives the details of the fields that will be matched to control snapshot for PTP packets over UDP over IPv4 for IEEE 1588 version 1 and 2. Note that the octet positions for tagged frames will be offset by 4. This is based on Appendix-D of the IEEE 1588-2008 standard and the message format defined in **Table 14-35**.

Table 14-36 IPv4-UDP PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Frame type	12, 13	0800 _H	IPv4 datagram
IP Version and Header Length	14	45 _H	IP version is IPv4
Layer-4 protocol	23	11 _H	UDP

Table 14-36 IPv4-UDP PTP Frame Fields Required for Control and Status (cont'd)

Field Matched	Octet Position	Matched Value	Description
IP Multicast address (IEEE 1588 version 1)	30, 31, 32, 33	$E0_H, 00_H,$ $01_H, 81_H$ (or 82_H or 83_H or 84_H)	Multicast IPv4 addresses allowed. 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP Multicast address (IEEE 1588 version 2)	30, 31, 32, 33	$E0_H, 00_H, 01_H,$ 81_H $E0_H, 00_H, 00_H,$ $6B_H$	PTP-primary multicast address: 224.0.1.129 PTP-Pdelay multicast address: 224.0.0.107
UDP destination port	36, 37	$013F_H,$ 0140_H	$013F_H$ – PTP event message ¹⁾ 0140_H – PTP general messages
PTP control field (IEEE version 1)	74	$00_H/01_H/02_H/03_H$ $/04_H$	00_H – SYNC, 01_H – Delay_Req, 02_H – Follow_Up 03_H – Delay_Resp 04_H – Management
PTP Message Type Field (IEEE version 2)	42 (nibble)	$0_H/1_H/2_H/3_H/8_H/9_H$ $/B_H/C_H/D_H$	0_H – SYNC 1_H – Delay_Req 2_H – Pdelay_Req 3_H – Pdelay_Resp 8_H – Follow_Up 9_H – Delay_Resp A_H – Pdelay_Resp_Follow_Up B_H – Announce C_H – Signaling D_H – Management
PTP version field	43 (nibble)	1_H or 2_H	1 – Supports PTP version 1 2 – Supports PTP version 2

1) PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only).

PTP Frame Over IPv6

Table 14-37 gives the details of the fields that will be matched to control snapshot for PTP packets over UDP over IPv6 for IEEE 1588 version 1 and 2. Note that the octet positions for tagged frames will be offset by 4. This is based on Appendix-E of the IEEE 1588-2008 standard and the message format defined in **Table 14-35**.

Table 14-37 IPv6-UDP PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Frame type	12, 13	86DD _H	IP datagram
IP version	14 (bits [7:4])	6 _H	IP version is IPv6
Layer-4 protocol	20 ¹⁾	11 _H	UDP
PTP Multicast address	38 – 53	FF0:0:0:0:0:0:181 _H FF02:0:0:0:0:0:0:6B _H	PTP – primary multicast address: FF0:0:0:0:0:0:0:181 _H PTP – Pdelay multicast address: FF02:0:0:0:0:0:0:6B _H
UDP destination port	56, 57 (*)	013F _H , 140 _H	013F _H – PTP event message 0140 _H – PTP general messages
PTP control field (IEEE 1588 Version 1)	93 (*)	00 _H /01 _H /02 _H /03 _H /04 _H	00 _H – SYNC, 01 _H – Delay_Req, 02 _H – Follow_Up 03 _H – Delay_Resp 04 _H – Management (version1)
PTP Message Type Field (IEEE version 2)	74 (*) (nibble)	0 _H /1 _H /2 _H /3 _H /8 _H /9 _H / B _H /C _H /D _H	0 _H – SYNC 1 _H – Delay_Req 2 _H – Pdelay_Req 3 _H – Pdelay_Resp 8 _H – Follow_Up 9 _H – Delay_Resp A _H – Pdelay_Resp_Follow_Up B _H – Announce C _H – Signaling D _H – Management
PTP version field	75 (nibble)	1 _H or 2 _H	1 _H – Supports PTP version 1 2 _H – Supports PTP version 2

1) The Extension Header is not defined for PTP packets.

PTP Frame Over Ethernet

Table 14-38 gives the details of the fields that will be matched to control snapshot for PTP packets over Ethernet for IEEE 1588 version 1 and 2. Note that the octet positions

for tagged frames will be offset by 4. This is based on Appendix-E of the IEEE 1588-2008 standard and the message format defined in [Table 14-35](#).

Table 14-38 Ethernet PTP Frame Fields Required for Control And Status

Field Matched	Octet Position	Matched Value	Description
MAC Frame type	12, 13	88F7 _H	PTP Ethernet frame.
PTP control field (IEEE Version 1)	45	00 _H /01 _H /02 _H / 03 _H /04 _H	00 _H – SYNC 01 _H – Delay_Req 02 _H – Follow_Up 03 _H – Delay_Resp 04 _H – Management
PTP Message Type Field (IEEE version 2)	14 (nibble)	0 _H /1 _H /2 _H /3 _H /8 _H /9 _H /B H/ C _H /D _H	0 _H – SYNC 1 _H – Delay_Req 2 _H – Pdelay_Req 3 _H – Pdelay_Resp 8 _H – Follow_Up 9 _H – Delay_Resp A _H – Pdelay_Resp_Follow_Up B _H – Announce C _H – Signaling D _H – Management
MAC Destination multicast address ¹⁾	0-5	01-1B-19-00-00- 00 _H 01-80-C2-00-00- 0E _H	All except peer delay messages - 01-1B-19-00-00-00 _H Pdelay messages - 01-80-C2-00-00-0E _H
PTP version field	15 (nibble)	1 _H or 2 _H	1 _H – Supports PTP version 1 2 _H – Supports PTP version 2

1) In addition, the address match of destination addresses (DA) programmed in MAC address 1 to 31 will be used, if the control bit 18 (TSENMACADDR: Enable MAC address for PTP frame filtering) of the Time Stamp Control register is set.

14.2.11.10 Reference Timing Source (for Advance Timestamp Feature)

The updated functionality for advanced timestamp support is mentioned in the following points.

1. The IEEE 1588-2008 standard defines the seconds field of the time to be 48 bits wide. The fields to time-stamp will be the following.
 - a) UInteger48- seconds field

b) UInteger32-nanoseconds field

The “seconds” field is the integer portion of the timestamp in units of seconds. The “nanoseconds” field is the fractional portion of the timestamp in units of nanoseconds. E.g. 2.000000001 seconds is represented as secondsField = 0000 0000 0002_H and nanoSeconds = 0000 0001_H. Thus the maximum value in nanoseconds field in this format will be 3B9A C9FF_H value (i.e (10e9-1) nano-seconds). This is defined as digital rollover mode of operation. It will also support the older mode in which the nano-seconds field will roll-over and increment the seconds field after the value of 7FFF FFFF_H. (Accuracy is ~0.466 ns per bit). This is defined as the binary rollover mode. The modes can be controlled using the **“ETH0_TIMESTAMP_CONTROL.TSCTRLSSR** bit.

2. When the Advanced IEEE 1588 time-stamp feature is selected time maintained in the core will still be 64-bit wide, as the overflow to the upper 16-bits of seconds register happens once in 130 years. The value of the upper 16-bits of the seconds field can only be obtained from the CSR register.

14.2.11.11 Transmit Path Functions

There are no changes in the transmit path functions for ETH-CORE and ETH-MTL configuration for the Advanced time stamp option.

structure of the descriptor changes when Advanced IEEE 1588 version support is enabled. The IEEE 1588 timestamp feature is supported using Alternate (Enhanced) descriptors format only. The descriptor is 32-bytes long (8 DWORDS) and the snapshot of the timestamp is written in descriptor 6 and 7.

14.2.11.12 Receive Path Functions

When the advanced time stamp feature is selected, processing of the received frames to identify valid PTP frames is done. The snapshot of the time to be sent to the application can be controlled.

The following options are provided in the **TIMESTAMP_CONTROL** register to control the snapshot.

1. Option to enable snapshot for all frames.
2. Enable snapshot for IEEE 1588 version 2 or version 1 time stamp.
3. Enable snapshot for PTP frames transmitted directly over Ethernet or UDP-IP-Ethernet.
4. Enable time stamp snapshot for the received frame for IPv4 or IPv6.
5. Enable time stamp snapshot for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ or PDELAY_RESP) only.
6. Enable the node to be a Master or Slave. This will control the type of messages for which snap-shot will be taken (this depends on the type of clock that is selected and is valid for ordinary or boundary clock only).

Note that PTP messages over VLAN frames are also supported.

14.2.12 System Time Register Module

A system time clock is maintained in this module. A 64 bit timer is incremented using the PTP clock (f_{CPU}) as reference. This time is the source for taking snapshots (time stamps) of Ethernet frames being transmitted or received at the MII.

The System Time counter can be initialized or corrected using the coarse correction method. In this method, the initial value or the offset value is written to the Time Stamp Update register. For initialization, the System Time counter (`ETH0_SYSTEM_TIME_SECONDS` and `ETH0_SYSTEM_TIME_NANOSECONDS`) is written with the value in the Time Stamp Update registers (`ETH0_SYSTEM_TIME_SECONDS_UPDATE` and `ETH0_SYSTEM_TIME_NANOSECONDS_UPDATE`), while for system time correction, the offset value is added to or subtracted from the system time.

In the fine correction method, a slave clock's (f_{CPU}) frequency drift with respect to the master clock (as defined in IEEE 1588) is corrected over a period of time instead of in one clock, as in coarse correction. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register, as shown in [Figure 14-28](#). The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. Here, the accumulator acts as a high-precision frequency multiplier or divider.

This algorithm is depicted in [Figure 14-28](#):

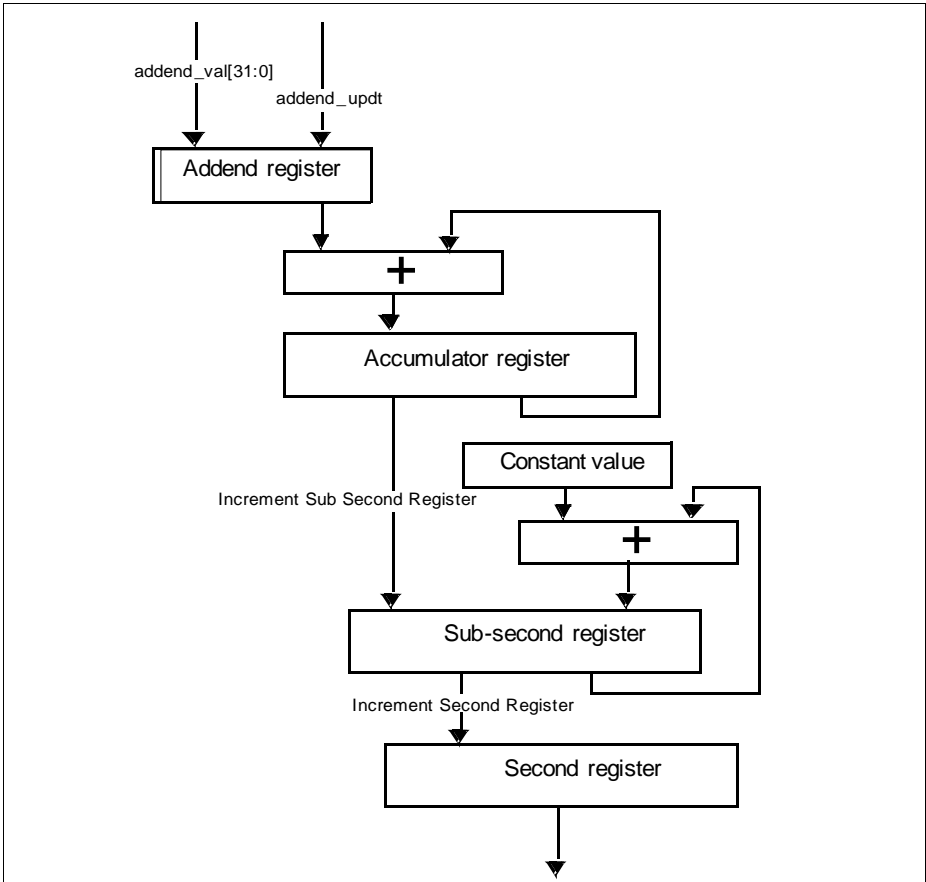


Figure 14-28 System Time Update Using Fine Method

The System Time Update logic requires a 50-MHz clock frequency to achieve 20-ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. Hence, if the reference clock (f_{CPU}) is, for example, 66 MHz, this ratio is calculated as $66 \text{ MHz} / 50 \text{ MHz} = 1.32$. Hence, the default addend value to be set in the register is $2^{32} / 1.32$, $0C1F07C1F_H$.

If the reference clock drifts lower, to 65 MHz for example, the ratio is $65 / 50$, or 1.3 and the value to set in the addend register is $2^{32} / 1.30$, or $0C4EC4EC4_H$. If the clock drifts higher, to 67 MHz for example, the addend register must be set to $0BF0B7672_H$. When the clock drift is nil, the default addend value of $0C1F07C1F_H$ ($2^{32} / 1.32$) must be programmed.

In **Figure 14-28**, the constant value used to accumulate the sub-second register is decimal 43, which achieves an accuracy of 20 ns in the system time (in other words, it is incremented in 20-ns steps). Two different methods are used to update the System Time register, depending on which configuration you choose (See **Block Diagram**).

The software must calculate the drift in frequency based on the Sync messages and update the Addend register accordingly.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$$\text{FreqCompensationValue}_0 = 2^{32} / \text{FreqDivisionRatio}$$

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm described below must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

- At time MasterSyncTime_n the master sends the slave clock a Sync message. The slave receives this message when its local clock is SlaveClockTime_n and computes MasterClockTime_n as:
MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n
- The master clock count for current Sync cycle, MasterClockCount_n is given by:
MasterClockCount_n = MasterClockTime_n – MasterClockTime_{n – 1} (assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)
- The slave clock count for current Sync cycle, SlaveClockCount_n is given by:
SlaveClockCount_n = SlaveClockTime_n – SlaveClockTime_{n – 1}
- The difference between master and slave clock counts for current Sync cycle, ClockDiffCount_n is given by:
ClockDiffCount_n = MasterClockCount_n – SlaveClockCount_n
- The frequency-scaling factor for slave clock, FreqScaleFactor_n is given by:
FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n
- The frequency compensation value for Addend register, FreqCompensationValue_n is given by:
FreqCompensationValue_n = FreqScaleFactor_n * FreqCompensationValue_{n – 1}

In theory, this algorithm achieves lock in one Sync cycle; however, it may take several cycles, due to changing network propagation delays and operating conditions.

This algorithm is self-correcting: if for any reason the slave clock is initially set to a value from the master that is incorrect, the algorithm will correct it at the cost of more Sync cycles.

14.2.13 Application BUS Interface

In the ETH core, the DMA Controller interfaces with the CPU through the Bus Interface. The Bus Master Interface controls data transfers while the Bus Slave interface accesses

CSR space. The DMA can be used in applications where DMA is required to optimize data transfer between the ETH and system memory.

The Bus Master interface converts the internal DMA request cycles into Bus cycles.

Characteristics of this interface include the following:

- You can choose fixed burst length of SINGLE, INCR4, INCR8 by programming the **ETH0_BUS_MODE**.MB bits
 - When transferring fixed burst length data, the Bus master always initiates a burst with SINGLE or INCR4/8 type. But when such a burst is responded with SPLIT/RETRY/early burst termination, the Bus master will re-initiate the pending transfers of the burst with INCR or SINGLE burst-length type. It will terminate such INCR bursts when the original requested fixed-burst is transferred. In Fixed Burst-Length mode, if the DMA requests a burst transfer that is not equal to INCR4/8, the Bus interface splits the transfer into multiple burst transactions. For example, if the DMA requests a 15-beat burst transfer, the Bus interface splits it into multiple transfers of INCR8 and INCR4 and 3 SINGLE transactions.
- Takes care of Bus SPLIT, RETRY, and ERROR conditions. Any ERROR response will halt all further transactions for that DMA, and indicate the error as fatal through the CSR and interrupt. The application must give a hard or soft reset to the module to restart the operation.
- Takes care of Bus 1K boundary breaking
- Handles all data transfers, except for Descriptor Status Write accesses (which are always 32-bit). In any burst data transfer, the address bus value is always aligned to the data bus width and need not be aligned to the beat size.

All Bus burst transfers can be aligned to an address value by enabling the **ETH0_BUS_MODE**.AAL bit. If both the FB and AAL bits are set to 1, the Bus interface and the DMA together ensure that all initiated beats are aligned to the address, completing the frame transfer in the minimum number of required beats. For example, if a data buffer transfer's start address is F000 0008_H and the DMA is configured for a maximum beat size of, the Bus transfers occur in the following sequence:

 - 2 SINGLE transfers at addresses F000 0008_H and F000 000C_H
 - 1 INCR4 transfer at address F000 0010_H
- The DMA Controller requests an Bus Burst Read transfer only when it can accept the received burst data completely. Data read from the Bus is always pushed into the DMA without any delay or BUSY cycles.
- The DMA requests an Bus Burst Write transfer only when it has the sufficient data to transfer the burst completely. The Bus interface always assumes that it has data available to push into the bus. However, the DMA can prematurely indicate end-of-valid data (due to the transfer of end-of-frame of an Ethernet frame) during the burst. The Bus Master interface continues the burst with dummy data until the specified length is completed.

The Bus 32-bit Slave interface provides access to the DMA and ETH CSR space. Characteristics of this interface include the following:

- Supports single and INCR4/8transfers
- Supports busy and early terminations
- Supports 32-bit, 16-bit, and 8-bit write/read transfers to the CSR; 32-bit access to the CSR are recommended to avoid any SW synchronization problems.
- Generates OKAY only response; does not generate SPLIT, RETRY, or ERROR responses.

14.3 Service Request Generation

Service requests can be generated from the ETH core as a result of various events in the modules within the ETH peripheral. There are four sources of service request, the ETH DMA, The Power Management module, the System timer module and the MAC management counters. Each of the events raised by these modules are ORed together and connected to an ETH Service Request line which is connected to the NVIC. The events are not queued and the application software must check all the status bits to ensure all events are serviced. Before exiting the service request routine the application software must ensure all status bits are de asserted or spurious service requests will be generated

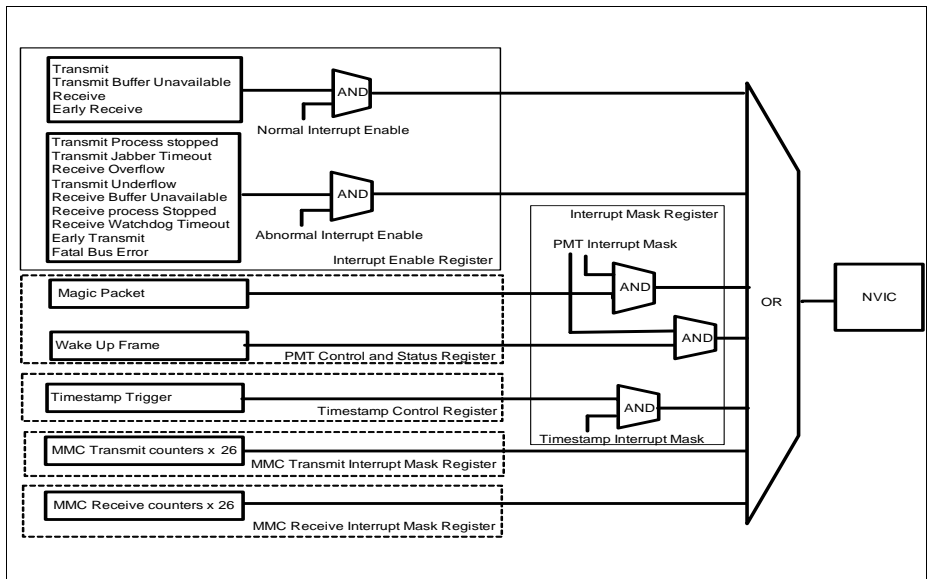


Figure 14-29 ETH Core Service Request Structure

14.3.1 DMA Service Requests

The ETH DMA has two groups of Service Request, Normal and Abnormal requests. Each Service Request source is enabled in the **ETH0_INTERRUPT_ENABLE** register. Each group of Service Requests must also be enabled by setting the Normal Interrupt enable and Abnormal Interrupt enable bits in the same register. When a service request is raised the matching Service request bits will be set in the **ETH0_STATUS** register. Global summary bits for the Power management MAC management counters and system time module are also provided in the status register.

14.3.2 Power Management Service Requests

The power management module provides two Service Requests, Wake up Frame and Magic packet. Both of these service requests may be enabled and monitored in the **PMT_CONTROL_STATUS** Register. To enable any power management service request is it also necessary to clear the **INTERRUPT_MASK.PMTIM** bit. The **INTERRUPT_STATUS.PMTIS** bit provides a global status bit for power management service requests.

14.3.3 System Time Module

The system time module provides a single Timestamp trigger service request which can be enabled in the timestamp control register. The **INTERRUPT_MASK.TSIM** bit must also be cleared to enable the timestamp trigger Service request. The **INTERRUPT_STATUS.TSIS** is set when a system time module service request occurs.

14.3.4 MAC Management Counter Service Requests

Each of the MAC Management Counters can generate a service request. The service requests are split into two groups of transmit and receive counters. Each counter may be individually enabled in either the **MMC_RECEIVE_INTERRUPT_MASK** register or the **MMC_TRANSMIT_INTERRUPT_MASK** register. When a MMC service request is generated status bits for each counter are set in the **MMC_RECEIVE_INTERRUPT** register or the **MMC_TRANSMIT_INTERRUPT** register. Two global status bits for the transmit and receive counters are provided in the **INTERRUPT_STATUS** register

14.4 Debug

Module specific debug behavior TBD

In addition the ETH has a number of intrinsic features to assist debugging, these are described below.

- The **DEBUG** register provides flags which indicate the operating status of the ETH MAC and MTL.
- The **STATUS** register provides information on the operating status of the DMA.
- The **MAC Management Counters** provide extensive information about the Received and transmitted Ethernet frames.
- The **MAC_CONFIGURATION.LM** bit places the ETH in internal loopback mode for self test and debug
- External loopback is supported via the integrated MDIO controlling the PHY
- The **CURRENT_HOST_TRANSMIT_DESCRIPTOR** and **CURRENT_HOST_RECEIVE_DESCRIPTOR** provide pointers to the current location of the transmit and receive frame buffers held in RAM

14.5 Power Reset and Clock

The module, including all registers, can be reset to its default state by a system reset or a software reset triggered through the setting of corresponding bits in the SCU PRSETx registers.

Important

After the XMC4300 is released from reset the ETH module remains held in reset. While the ETH is held in reset the software driver must select the PHY interconnect see [Section 14.2.7.1](#) . Once the PHY interconnect has been selected ETH reset line must be deasserted by setting PRCLR2.ETH0RS in the system control unit.

For proper operation of the Ethernet Interface it is required that $f_{SYS} \geq 100$ MHz..

14.6 ETH Registers

The application controls the ETH by reading from and writing to the Control and Status Registers (CSRs) through the BUS Slave interface. These registers are 32 bits wide and the addresses are 32-bit block aligned

14.6.1 Register Description

ETH Register Map

Table 14-40 provides the address map of the ETH core registers.

0000 _H	ETH Core
00DC _H	Reserved
0100 _H	ETH MAC Management Counters
0288 _H	Reserved
0700 _H	IEEE1588 System time module
07FC _H	Reserved
1000 _H	ETH DMA
1058 _H	

Figure 14-30 ETH Register memory Map

Table 14-39 Registers Address Space - ETH Module

Module	Base Address	End Address	Note
ETH0	5000 C000 _H	5000 FFFF _H	-

14.6.2 Registers Overview

Table 14-40 ETH Registers Overview

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
MAC Configuration Registers					
MAC_CONFIGURATI ON	MAC Configuration Register	0000 _H	U,PV	U,PV	Page 14-125
MAC_FRAME_FILTE R	MAC Frame Filter Register	0004 _H	U,PV	U,PV	Page 14-132
HASH_TABLE_HIGH	Hash Table High Register	0008 _H	U,PV	U,PV	Page 14-137
HASH_TABLE_LOW	Hash Table Low Register	000C _H	U,PV	U,PV	Page 14-139
GMII_ADDRESS	MII Address Register	0010 _H	U,PV	U,PV	Page 14-140
GMII_DATA	MII Data Register	0014 _H	U,PV	U,PV	Page 14-143
FLOW_CONTROL	Flow Control Register	0018 _H	U,PV	U,PV	Page 14-144
VLAN_TAG	VLAN Tag Register	001C _H	U,PV	U,PV	Page 14-148
VERSION	Version Register	0020 _H	U,PV	NC	Page 14-150
DEBUG	Debug Register	0024 _H	U,PV	U,PV	Page 14-151
REMOTE_WAKE_UP _FRAME_FILTER	Remote Wake Up Frame Filter Register	0028 _H	U,PV	U,PV	Page 14-154
PMT_CONTROL_ST ATUS	PMT Control Status Register	002C _H	U,PV	U,PV	Page 14-155
Reserved		0030 _H - 0034 _H	nBE	nBE	Do not use
INTERRUPT_STATU S	Interrupt Status Register	0038 _H	U,PV	U,PV	Page 14-157
INTERRUPT_MASK	Interrupt Mask Register	003C _H	U,PV	U,PV	Page 14-159
MAC_ADDRESS0_HI GH	MAC Address 0 High Register	0040 _H	U,PV	U,PV	Page 14-160
MAC_ADDRESS0_L OW	MAC Address0 Low Register	0044 _H	U,PV	U,PV	Page 14-161

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
MAC_ADDRESS1_HIGH	MAC Address1 High Register	0048 _H	U,PV	U,PV	Page 14-162
MAC_ADDRESS1_LOW	MAC Address1 Low Register	004C _H	U,PV	U,PV	Page 14-164
MAC_ADDRESS2_HIGH	MAC Address High Register	0050 _H	U,PV	U,PV	Page 14-165
MAC_ADDRESS2_LOW	MAC Address1 Low Register	0054 _H	U,PV	U,PV	Page 14-167
MAC_ADDRESS3_HIGH	MAC Address High Register	0058 _H	U,PV	U,PV	Page 14-168
MAC_ADDRESS3_LOW	MAC Address1 Low Register	005C _H	U,PV	U,PV	Page 14-170
Reserved		00DC _H - 00FC _H	nBE	nBE	Do not use
MAC Management Counters					
MMC_CONTROL	MMC Control	0100 _H	U,PV	U,PV	Page 14-171
MMC_RECEIVE_INTERRUPT	MMC Receive Interrupt	0104 _H	U,PV	U,PV	Page 14-173
MMC_TRANSMIT_INTERRUPT	MMC Transmit Interrupt	0108 _H	U,PV	U,PV	Page 14-178
MMC_RECEIVE_INTERRUPT_MASK	MMC Receive Interrupt mask	010C _H	U,PV	U,PV	Page 14-183
MMC_TRANSMIT_INTERRUPT_MASK	MMC Transmit Interrupt Mask	0110 _H	U,PV	U,PV	Page 14-188
TX_OCTET_COUNT_GOOD_BAD	Number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad frames.	0114 _H	U,PV	U,PV	Page 14-193
TX_FRAME_COUNT_GOOD_BAD	Number of good and bad frames transmitted, exclusive of retried frames.	0118 _H	U,PV	U,PV	Page 14-194

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
TX_BROADCAST_FRAMES_GOOD	Number of good broadcast frames transmitted.	011C _H	U,PV	U,PV	Page 14-195
TX_MULTICAST_FRAMES_GOOD	Number of good multicast frames transmitted.	0120 _H	U,PV	U,PV	Page 14-196
TX_64OCTETS_FRAMES_GOOD_BAD	Number of good and bad frames transmitted with length 64 bytes, exclusive of preamble and retried frames.	0124 _H	U,PV	U,PV	Page 14-197
TX_65TO127OCTETS_FRAMES_GOOD_BAD	Number of good and bad frames transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.	0128 _H	U,PV	U,PV	Page 14-198
TX_128TO255OCTETS_FRAMES_GOOD_BAD	Number of good and bad frames transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames.	012C _H	U,PV	U,PV	Page 14-199
TX_256TO511OCTETS_FRAMES_GOOD_BAD	Number of good and bad frames transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.	0130 _H	U,PV	U,PV	Page 14-200

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
TX_512TO1023OCTETS_FRAMES_GOOD_BAD	Number of good and bad frames transmitted with length between 512 and 1.023 (inclusive) bytes, exclusive of preamble and retried frames.	0134 _H	U,PV	U,PV	Page 14-201
TX_1024TOMAXOCTETS_FRAMES_GOOD_BAD	Number of good and bad frames transmitted with length between 1.024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.	0138 _H	U,PV	U,PV	Page 14-202
TX_UNICAST_FRAMES_GOOD_BAD	Number of good and bad unicast frames transmitted.	013C _H	U,PV	U,PV	Page 14-203
TX_MULTICAST_FRAMES_GOOD_BAD	Number of good and bad multicast frames transmitted.	0140 _H	U,PV	U,PV	Page 14-204
TX_BROADCAST_FRAMES_GOOD_BAD	Number of good and bad broadcast frames transmitted.	0144 _H	U,PV	U,PV	Page 14-205
TX_UNDERFLOW_ERROR_FRAMES	Number of frames aborted due to frame underflow error.	0148 _H	U,PV	U,PV	Page 14-206
TX_SINGLE_COLLISION_GOOD_FRAMES	Number of successfully transmitted frames after a single collision in Half-duplex mode.	014C _H	U,PV	U,PV	Page 14-207
TX_MULTIPLE_COLLISION_GOOD_FRAMES	Number of successfully transmitted frames after more than a single collision in Half-duplex mode.	0150 _H	U,PV	U,PV	Page 14-208

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
TX_DEFERRED_FRAMES	Number of successfully transmitted frames after a deferral in Half-duplex mode.	0154 _H	U,PV	U,PV	Page 14-209
TX_LATE_COLLISION_FRAMES	Number of frames aborted due to late collision error.	0158 _H	U,PV	U,PV	Page 14-210
TX_EXCESSIVE_COLLISION_FRAMES	Number of frames aborted due to excessive (16) collision errors.	015C _H	U,PV	U,PV	Page 14-211
TX_CARRIER_ERROR_FRAMES	Number of frames aborted due to carrier sense error (no carrier or loss of carrier).	0160 _H	U,PV	U,PV	Page 14-212
TX_OCTET_COUNT_GOOD	Number of bytes transmitted, exclusive of preamble, in good frames only.	0164 _H	U,PV	U,PV	Page 14-213
TX_FRAME_COUNT_GOOD	Number of good frames transmitted.	0168 _H	U,PV	U,PV	Page 14-214
TX_EXCESSIVE_DEFERRAL_ERROR	Number of frames aborted due to excessive deferral error (deferred for more than two max-sized frame times).	016C _H	U,PV	U,PV	Page 14-215
TX_PAUSE_FRAMES	Number of good PAUSE frames transmitted.	0170 _H	U,PV	U,PV	Page 14-216
TX_VLAN_FRAMES_GOOD	Number of good VLAN frames transmitted, exclusive of retried frames.	0174 _H	U,PV	U,PV	Page 14-217

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
TX_OSIZE_FRAME S_GOOD	Number of transmitted good Oversize frames, exclusive of retried frames.	0178 _H	U,PV	U,PV	Page 14-218
Reserved		017C _H	nBE	nBE	Do not use
RX_FRAMES_COUNT_GOOD_BAD	Number of good and bad frames received.	0180 _H	U,PV	U,PV	Page 14-219
RX_OCTET_COUNT_GOOD_BAD	Number of bytes received, exclusive of preamble, in good and bad frames.	0184 _H	U,PV	U,PV	Page 14-220
RX_OCTET_COUNT_GOOD	Number of bytes received, exclusive of preamble, only in good frames.	0188 _H	U,PV	U,PV	Page 14-221
RX_BROADCAST_FRAMES_GOOD	Number of good broadcast frames received.	018C _H	U,PV	U,PV	Page 14-222
RX_MULTICAST_FRAMES_GOOD	Number of good multicast frames received.	0190 _H	U,PV	U,PV	Page 14-223
RX_CRC_ERROR_FRAMES	Number of frames received with CRC error.	0194 _H	U,PV	U,PV	Page 14-224
RX_ALIGNMENT_ERROR_FRAMES	Number of frames received with alignment (dribble) error.	0198 _H	U,PV	U,PV	Page 14-225
RX_RUNT_ERROR_FRAMES	Number of frames received with runt (<64 bytes and CRC error) error.	019C _H	U,PV	U,PV	Page 14-226

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
RX_JABBER_ERROR_FRAMES	Number of giant frames received with length (including CRC) greater than 1.518 bytes (1.522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames.	01A0 _H	U,PV	U,PV	Page 14-227
RX_UNDERSIZE_FRAMES_GOOD	Number of frames received with length less than 64 bytes, without any errors.	01A4 _H	U,PV	U,PV	Page 14-228
RX_OVERSIZE_FRAMES_GOOD	Number of frames received with length greater than the maxsize (1.518 or 1.522 for VLAN tagged frames), without errors.	01A8 _H	U,PV	U,PV	Page 14-229
RX_64OCTETS_FRAMES_GOOD_BAD	Number of good and bad frames received with length 64 bytes, exclusive of preamble.	01AC _H	U,PV	U,PV	Page 14-230
RX_65TO127OCTETS_FRAMES_GOOD_BAD	Number of good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble.	01B0 _H	U,PV	U,PV	Page 14-231

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
RX_128TO255OCTE TS_FRAMES_GOO D_BAD	Number of good and bad frames received with length between 128 and 255 (inclusive) bytes, exclusive of preamble.	01B4 _H	U,PV	U,PV	Page 14-232
RX_256TO511OCTE TS_FRAMES_GOO D_BAD	Number of good and bad frames received with length between 256 and 511 (inclusive) bytes, exclusive of preamble.	01B8 _H	U,PV	U,PV	Page 14-233
RX_512TO1023OCT ETS_FRAMES_GO OD_BAD	Number of good and bad frames received with length between 512 and 1.023 (inclusive) bytes, exclusive of preamble.	01BC _H	U,PV	U,PV	Page 14-234
RX_1024TOMAXOC TETS_FRAMES_GO OD_BAD	Number of good and bad frames received with length between 1.024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.	01C0 _H	U,PV	U,PV	Page 14-235
RX_UNICAST_FRA MES_GOOD	Number of good unicast frames received.	01C4 _H	U,PV	U,PV	Page 14-236
RX_LENGTH_ERRO R_FRAMES	Number of frames received with length error (Length type field ¼ frame size), for all frames with valid length field.	01C8 _H	U,PV	U,PV	Page 14-237

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
RX_OUT_OF_RANG E_TYPE_FRAMES	Number of frames received with length field not equal to the valid frame size (greater than 1.500 but less than 1.536).	01CC _H	U,PV	U,PV	Page 14-238
RX_PAUSE_FRAME S	Number of good and valid PAUSE frames received.	01D0 _H	U,PV	U,PV	Page 14-239
RX_FIFO_OVERFL OW_FRAMES	Number of missed received frames due to FIFO overflow.	01D4 _H	U,PV	U,PV	Page 14-240
RX_VLAN_FRAMES _GOOD_BAD	Number of good and bad VLAN frames received.	01D8 _H	U,PV	U,PV	Page 14-241
RX_WATCHDOG_E RROR_FRAMES	Number of frames received with error due to watchdog timeout error (frames with a data load larger than 2.048 bytes).	01DC _H	U,PV	U,PV	Page 14-242
RX_RECEIVE_ERR OR_FRAMES	Number of frames received with error because of the MII RXER error.	01E0 _H	U,PV	U,PV	Page 14-243
RX_CONTROL_FRA MES_GOOD	Number of god control frames received.	01E4 _H	U,PV	U,PV	Page 14-244
Reserved		01E8 _H - 01FC _H	nBE	nBE	Do not use
MMC_IPC_RECEIV E_INTERRUPT_MA SK	MMC IPC Receive Checksum Offload Interrupt Mask maintains the mask for the interrupt generated from the receive IPC statistic counters.	0200 _H	U,PV	U,PV	Page 14-245

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
Reserved		0204 _H	nBE	nBE	
MMC_IPC_RECEIVE_INTERRUPT	MMC Receive Checksum Offload Interrupt maintains the interrupt that the receive IPC statistic counters generate.	0208 _H	U,PV	U,PV	Page 14-250
Reserved		020C _H	nBE	nBE	Do not use
RXIPV4_GOOD_FRAMES	Number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload	0210 _H	U,PV	U,PV	Page 14-255
RXIPV4_HEADER_ERROR_FRAMES	Number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors	0214 _H	U,PV	U,PV	Page 14-256
RXIPV4_NO_PAYLOAD_FRAMES	Number of IPv4 datagram frames received that did not have a TCP, UDP, or ICMP payload processed by the Checksum engine	0218 _H	U,PV	U,PV	Page 14-257
RXIPV4_FRAGMENTED_FRAMES	Number of good IPv4 datagrams with fragmentation	021C _H	U,PV	U,PV	Page 14-258
RXIPV4_UDP_CHECKSUM_DISABLED_FRAMES	Number of good IPv4 datagrams received that had a UDP payload with checksum disabled	0220 _H	U,PV	U,PV	Page 14-259
RXIPV6_GOOD_FRAMES	Number of good IPv6 datagrams received with TCP, UDP, or ICMP payloads	0224 _H	U,PV	U,PV	Page 14-260

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
RXIPv6_HEADER_ERROR_FRAMES	Number of IPv6 datagrams received with header errors (length or version mismatch)	0228 _H	U,PV	U,PV	Page 14-261
RXIPv6_NO_PAYLOAD_FRAMES	Number of IPv6 datagram frames received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers	022C _H	U,PV	U,PV	Page 14-262
RXUDP_GOOD_FRAMES	Number of good IP datagrams with a good UDP payload. This counter is not updated when the RXIPv4_UDP_CHECKSUM_DISABLED_FRAMES counter is incremented.	0230 _H	U,PV	U,PV	Page 14-263
RXUDP_ERROR_FRAMES	Number of good IP datagrams whose UDP payload has a checksum error	0234 _H	U,PV	U,PV	Page 14-264
RXTCP_GOOD_FRAMES	Number of good IP datagrams with a good TCP payload	0238 _H	U,PV	U,PV	Page 14-265
RXTCP_ERROR_FRAMES	Number of good IP datagrams whose TCP payload has a checksum error	023C _H	U,PV	U,PV	Page 14-266
RXICMP_GOOD_FRAMES	Number of good IP datagrams with a good ICMP payload	0240 _H	U,PV	U,PV	Page 14-267

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
RXICMP_ERROR_FRAMES	Number of good IP datagrams whose ICMP payload has a checksum error	0244 _H	U,PV	U,PV	Page 14-268
Reserved		0248 _H – 024C _H	nBE	nBE	Do not use
RXIPV4_GOOD_OCTETS	Number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter or in the octet counters listed below).	0250 _H	U,PV	U,PV	Page 14-269
RXIPV4_HEADER_ERROR_OCTETS	Number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter.	0254 _H	U,PV	U,PV	Page 14-275
RXIPV4_NO_PAYLOAD_OCTETS	Number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv4 header's Length field is used to update this counter.	0258 _H	U,PV	U,PV	Page 14-271

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
RXIPV4_FRAGMENTED_OCTETS	Number of bytes received in fragmented IPv4 datagrams. The value in the IPv4 header's Length field is used to update this counter.	025C _H	U,PV	U,PV	Page 14-272
RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS	Number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes.	0260 _H	U,PV	U,PV	Page 14-273
RXIPV6_GOOD_OCTETS	Number of bytes received in good IPv6 datagrams encapsulating TCP, UDP or ICMPv6 data	0264 _H	U,PV	U,PV	Page 14-274
RXIPV6_HEADER_ERROR_OCTETS	Number of bytes received in IPv6 datagrams with header errors (length, version mismatch). The value in the IPv6 header's Length field is used to update this counter.	0268 _H	U,PV	U,PV	Page 14-275
RXIPV6_NO_PAYLOAD_OCTETS	Number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv6 header's Length field is used to update this counter.	026C _H	U,PV	U,PV	Page 14-276

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
RXUDP_GOOD_OC TETS	Number of bytes received in a good UDP segment. This counter (and the counters below) does not count IP header bytes.	0270 _H	U,PV	U,PV	Page 14-277
RXUDP_ERROR_O CTETS	Number of bytes received in a UDP segment that had checksum errors	0274 _H	U,PV	U,PV	Page 14-278
RXTCP_GOOD_OC TETS	Number of bytes received in a good TCP segment	0278 _H	U,PV	U,PV	Page 14-279
RXTCP_ERROR_O CTETS	Number of bytes received in a TCP segment with checksum errors	027C _H	U,PV	U,PV	Page 14-280
RXICMP_GOOD_O CTETS	Number of bytes received in a good ICMP segment	0280 _H	U,PV	U,PV	Page 14-281
RXICMP_ERROR_O CTETS	Number of bytes received in an ICMP segment with checksum errors	0284 _H	U,PV	U,PV	Page 14-282
Reserved		0288 _H - 02FC _H	nBE	nBE	Do not use
Reserved		0300 _H - 06FC _H	nBE	nBE	Do not use
System Time Registers					
TIMESTAMP_CONT ROL.	Timestamp Control Register	0700 _H	U,PV	U,PV	Page 14-283
SUB_SECOND_INC REMENT	Sub Second Increment Register	0704 _H	U,PV	U,PV	Page 14-286
SYSTEM_TIME_SEC ONDS	System Time Seconds Register	0708 _H	U,PV	U,PV	Page 14-287

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
SYSTEM_TIME_NAN OSECONDS	System Time Nanoseconds Register	070C _H	U,PV	U,PV	Page 14-288
SYSTEM_TIME_SEC ONDS_UPDATE	System Time Seconds Update Register	0710 _H	U,PV	U,PV	Page 14-289
SYSTEM_TIME_NAN OSECONDS_UPDAT E	System Time Nanoseconds Update Register	0714 _H	U,PV	U,PV	Page 14-290
TIMESTAMP_ADDE ND	Timestamp Addend Register	0718 _H	U,PV	U,PV	Page 14-291
TARGET_TIME_SEC ONDS	Target Time Seconds Register	071C _H	U,PV	U,PV	Page 14-292
TARGET_TIME_NAN OSECONDS	Target Time Nanoseconds Register	0720 _H	U,PV	U,PV	Page 14-293
SYSTEM_TIME_HIG HER_WORD_SECO NDS	System Time Higher Word Seconds Register	0724 _H	U,PV	U,PV	Page 14-295
TIMESTAMP_STATU S	Timestamp Status Register	0728 _H	U,PV	U,PV	Page 14-296
Do not use	Do not use	072C _H - 07FC _H	nBE	nBE	Do not use
Do not use	Do not use	0738 _H - 07FC _H	nBE	nBE	Do not use
DMA Registers					
BUS_MODE	BUS Mode Register	1000 _H	U,PV	U,PV	Page 14-299
TRANSMIT_POLL_D EMAND	Transmit Poll Demand Register	1004 _H	U,PV	U,PV	Page 14-304
RECEIVE_POLL_DE MAND	Receive Poll Demand Register	1008 _H	U,PV	U,PV	Page 14-305
RECEIVE_DESCRIP TOR_LIST_ADDRES S	Receive Descriptor List Address Register	100C _H	U,PV	U,PV	Page 14-306
TRANSMIT_DESCRI PTOR_LIST_ADDRE SS	Transmit Descriptor List Address Register	1010 _H	U,PV	U,PV	Page 14-307

Table 14-40 ETH Registers Overview (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Protection		Description See
			Read	Write	
STATUS	Status Register	1014 _H	U,PV	U,PV	Page 14-308
OPERATION_MODE	Operation Mode Register	1018 _H	U,PV	U,PV	Page 14-314
INTERRUPT_ENABLE	Interrupt Enable Register	101C _H	U,PV	U,PV	Page 14-320
MISSED_FRAME_AND_BUFFER_OVERFLOW_COUNTER	Missed Frame And Buffer Overflow Counter Register	1020 _H	U,PV	U,PV	Page 14-323
RECEIVE_INTERRUPT_WATCHDOG_TIMER	Receive Interrupt Watchdog Timer Register	1024 _H	U,PV	U,PV	Page 14-324
Reserved		1028 _H	nBE	nBE	Do not use
Reserved		1030 _H -1044 _H	nBE	nBE	Do not use
CURRENT_HOST_TRANSMIT_DESCRIPTOR	Current Host Transmit Descriptor Register	1048 _H	U,PV	U,PV	Page 14-326
CURRENT_HOST_RECEIVE_DESCRIPTOR	Current Host Receive Descriptor Register	104C _H	U,PV	U,PV	Page 14-327
CURRENT_HOST_TRANSMIT_BUFFER_ADDRESS	Current Host Transmit Buffer Address Register	1050 _H	U,PV	U,PV	Page 14-328
CURRENT_HOST_RECEIVE_BUFFER_ADDRESS	Current Host Receive Buffer Address Register	1054 _H	U,PV	U,PV	Page 14-329
HW_FEATURE	HW Feature Register	1058 _H	U,PV	NC	Page 14-330

1) The absolute register address is calculated as follows:
Module Base Address + Offset Address (shown in this column)

Field	Bits	Type	Description
TE	3	rw	<p>Transmitter Enable</p> <p>When this bit is set, the transmit state machine of the MAC is enabled for transmission on the MII. When this bit is reset, the MAC transmit state machine is disabled after the completion of the transmission of the current frame, and does not transmit any further frames.</p>
DC	4	rw	<p>Deferral Check</p> <p>When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Frame Abort status, along with the excessive deferral error bit set in the transmit frame status, when the transmit state machine is deferred for more than 24,288 bit times . If the Jumbo frame mode is enabled in the 10 or 100 Mbps mode, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but is prevented because of an active carrier sense signal (CRS) on the MII. Defer time is not cumulative. When the transmitter defers for 10,000 bit times, it transmits, collides, backs off, and then defers again after completion of back-off. The deferral timer resets to 0 and restarts.</p> <p>When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. This bit is applicable only in the half-duplex mode and is reserved (RO) in the full-duplex-only configuration.</p>

Field	Bits	Type	Description
BL	[6:5]	rw	<p>Back-Off Limit</p> <p>The Back-Off limit determines the random integer number (r) of slot time delays (512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. This bit is applicable only in the half-duplex mode and is reserved (RO) in the full-duplex-only configuration.</p> <p>* 00_B: $k = \min(n, 10)$ * 01_B: $k = \min(n, 8)$ * 10_B: $k = \min(n, 4)$ * 11_B: $k = \min(n, 1)$</p> <p>where $\langle i \rangle n \langle /i \rangle =$ retransmission attempt. The random integer $\langle i \rangle r \langle /i \rangle$ takes the value in the range $0 \leq r < k$th power of 2</p>
ACS	7	rw	<p>Automatic Pad or CRC Stripping</p> <p>When this bit is set, the MAC strips the Pad or FCS field on the incoming frames only if the value of the length field is less than 1,536 bytes. All received frames with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field.</p> <p>When this bit is reset, the MAC passes all incoming frames, without modifying them, to the XMC4300 Memory.</p>
Reserved_8	8	r	Reserved
DR	9	rw	<p>Disable Retry</p> <p>When this bit is set, the MAC attempts only one transmission. When a collision occurs on the MII interface, the MAC ignores the current frame transmission and reports a Frame Abort with excessive collision error in the transmit frame status.</p> <p>When this bit is reset, the MAC attempts retries based on the settings of the BL field (Bits [6:5]). This bit is applicable only in the half-duplex mode and is reserved (RO with default value) in the full-duplex-only configuration.</p>

Field	Bits	Type	Description
IPC	10	rw	<p>Checksum Offload</p> <p>When this bit is set, the MAC calculates the 16-bit ones complement of the ones complement sum of all received Ethernet frame payloads. It also checks whether the IPv4 Header checksum (assumed to be bytes 2526 or 2930 (VLAN-tagged) of the received Ethernet frame) is correct for the received frame and gives the status in the receive status word. The MAC also appends the 16-bit checksum calculated for the IP header datagram payload (bytes after the IPv4 header) and appends it to the Ethernet frame transferred to the application (when Type 2 COE is deselected).</p> <p>When this bit is reset, this function is disabled.</p> <p>When Type 2 COE is selected, this bit, when set, enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled and the corresponding PCE and IP HCE status bits are always cleared.</p>
DM	11	rw	<p>Duplex Mode</p> <p>When this bit is set, the MAC operates in the full-duplex mode where it can transmit and receive simultaneously.</p>
LM	12	rw	<p>Loopback Mode</p> <p>When this bit is set, the MAC operates in the loopback mode using the MII. The MII Receive clock input is required for the loopback to work properly, because the Transmit clock is not looped-back internally.</p>
DO	13	rw	<p>Disable Receive Own</p> <p>When this bit is set, the MAC disables the reception of frames in the half-duplex mode.</p> <p>When this bit is reset, the MAC receives all packets that are given by the PHY while transmitting.</p> <p>This bit is not applicable if the MAC is operating in the full-duplex mode.</p>
FES	14	rw	<p>Speed</p> <p>This bit selects the speed in the MII or RMII:</p> <ul style="list-style-type: none"> * 0_B: 10 Mbps * 1_B: 100 Mbps <p>This bit generates link speed encoding when TC (Bit 24) is set in the RMII mode.</p>

Field	Bits	Type	Description
Reserved	15	r	Reserved
DCRS	16	rw	<p>Disable Carrier Sense During Transmission</p> <p>When set high, this bit makes the MAC transmitter ignore the MII CRS signal during frame transmission in the half-duplex mode. This request results in no errors generated because of Loss of Carrier or No Carrier during such transmission. When this bit is low, the MAC transmitter generates such errors because of Carrier Sense and can even abort the transmissions.</p>
IFG	[19:17]	rw	<p>Inter-Frame Gap</p> <p>These bits control the minimum IFG between frames during transmission.</p> <ul style="list-style-type: none"> * 000_B: 96 bit times * 001_B: 88 bit times * 010_B: 80 bit times * ... * 111_B: 40 bit times <p>In the half-duplex mode, the minimum IFG can be configured only for 64 bit times (IFG = 100_B). Lower values are not considered.</p>
JE	20	rw	<p>Jumbo Frame Enable</p> <p>When this bit is set, the MAC allows Jumbo frames of 9,018 bytes (9,022 bytes for VLAN tagged frames) without reporting a giant frame error in the receive frame status.</p>
BE	21	r	<p>Frame Burst Enable</p> <p>When this bit is set, the MAC allows frame bursting during transmission in the MII half-duplex mode. This bit is reserved (and RO) in the 10/100 Mbps only or full-duplex-only configurations.</p>
JD	22	rw	<p>Jabber Disable</p> <p>When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer frames of up to 16,384 bytes.</p> <p>When this bit is reset, the MAC cuts off the transmitter if the application sends out more than 2,048 bytes of data (10,240 if JE is set high) during transmission.</p>

Field	Bits	Type	Description
WD	23	rw	<p>Watchdog Disable</p> <p>When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive frames of up to 16,384 bytes.</p> <p>When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the frame being received. The MAC cuts off any bytes received after 2,048 bytes.</p>
TC	24	r	<p>Transmit Configuration in RMII</p> <p>When set, this bit enables the transmission of duplex mode, link speed, and link up or down information to the PHY in RMII. When this bit is reset, no such information is driven to the PHY.</p>
CST	25	rw	<p>CRC Stripping of Type Frames</p> <p>When set, the last 4 bytes (FCS) of all frames of Ether type (type field greater than 0600_H) are stripped and dropped before forwarding the frame to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver.</p>
Reserved_26	26	r	Reserved
TWOKPE	27	rw	<p>IEEE 802.3as support for 2K packets Enable</p> <p>When set, the MAC considers all frames, with up to 2,000 bytes length, as normal packets. When Bit 20 (Jumbo Enable) is not set, the MAC considers all received frames of size more than 2K bytes as Giant frames.</p> <p>When this bit is reset and Bit 20 (Jumbo Enable) is not set, the MAC considers all received frames of size more than 1,518 bytes (1,522 bytes for tagged) as Giant frames.</p> <p>When Bit 20 (Jumbo Enable) is set, setting this bit has no effect on Giant Frame status.</p>

Field	Bits	Type	Description
SARC	[30:28]	r	<p>Source Address Insertion or Replacement Control</p> <p>This field controls the source address insertion or replacement for all transmitted frames. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits [29:28]:</p> <p>* 10_B:</p> <ul style="list-style-type: none"> - If Bit 30 is set to 0, the MAC inserts the content of the MAC Address 0 registers (ETH0_MAC_ADDRESS0_HIGH and ETH0_MAC_ADDRESS0_LOW) in the SA field of all transmitted frames. - If Bit 30 is set to 1 the MAC inserts the content of the MAC Address 1 registers (ETH0_MAC_ADDRESS1_HIGH and ETH0_MAC_ADDRESS1_LOW) in the SA field of all transmitted frames. <p>* 11_B:</p> <ul style="list-style-type: none"> - If Bit 30 is set to 0, the MAC replaces the content of the MAC Address 0 registers (ETH0_MAC_ADDRESS0_HIGH and ETH0_MAC_ADDRESS0_LOW) in the SA field of all transmitted frames. - If Bit 30 is set to 1 and the Enable MAC Address Register 1 option is selected during core configuration, the MAC replaces the content of the MAC Address 1 registers (ETH0_MAC_ADDRESS1_HIGH and ETH0_MAC_ADDRESS1_LOW) in the SA field of all transmitted frames. <p>Note:</p> <ul style="list-style-type: none"> - Changes to this field take effect only on the start of a frame. If you write this register field when a frame is being transmitted, only the subsequent frame can use the updated value, that is, the current frame does not use the updated value.
Reserved_31	31	r	Reserved

MAC_FRAME_FILTER

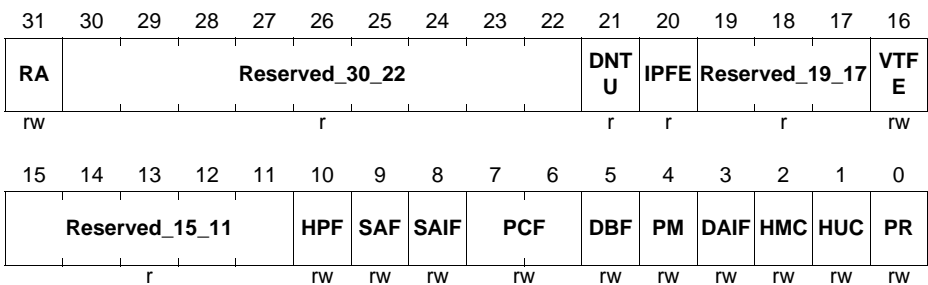
The MAC Frame Filter register contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering. The second level of filtering is performed on the incoming frame, based on other controls such as Pass Bad Frames and Pass Control Frames.

ETH0_MAC_FRAME_FILTER

MAC Frame Filter

(4_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PR	0	rw	Promiscuous Mode When this bit is set, the Address Filter module passes all incoming frames regardless of its destination or source address. The SA or DA Filter Fails status bits of the Receive Status Word are always cleared when PR is set.
HUC	1	rw	Hash Unicast When set, MAC performs destination address filtering of unicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for unicast frames, that is, it compares the DA field with the values programmed in DA registers. If Hash Filter is not selected during core configuration, this bit is reserved (and RO).

Field	Bits	Type	Description
HMC	2	rw	<p>Hash Multicast</p> <p>When set, MAC performs destination address filtering of received multicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for multicast frames, that is, it compares the DA field with the values programmed in DA registers. If Hash Filter is not selected during core configuration, this bit is reserved (and RO).</p>
DAIF	3	rw	<p>DA Inverse Filtering</p> <p>When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast frames. When reset, normal filtering of frames is performed.</p>
PM	4	rw	<p>Pass All Multicast</p> <p>When set, this bit indicates that all received frames with a multicast destination address (first bit in the destination address field is '1') are passed. When reset, filtering of multicast frame depends on HMC bit.</p>
DBF	5	rw	<p>Disable Broadcast Frames</p> <p>When this bit is set, the AFM module filters all incoming broadcast frames. In addition, it overrides all other filter settings. When this bit is reset, the AFM module passes all received broadcast frames.</p>

Field	Bits	Type	Description
PCF	[7:6]	rw	<p>Pass Control Frames</p> <p>These bits control the forwarding of all control frames (including unicast and multicast PAUSE frames).</p> <ul style="list-style-type: none"> * 00B: MAC filters all control frames from reaching the application. * 01B: MAC forwards all control frames except PAUSE control frames to application even if they fail the Address filter. * 10B: MAC forwards all control frames to application even if they fail the Address Filter. * 11B: MAC forwards control frames that pass the Address Filter. <p>The following conditions should be true for the PAUSE control frames processing:</p> <ul style="list-style-type: none"> * Condition 1: The MAC is in the full-duplex mode and flow control is enabled by setting FLOW_CONTROL.RFE. * Condition 2: The destination address (DA) of the received frame matches the special multicast address or the MAC Address 0 when FLOW_CONTROL.UP is set. * Condition 3: The Type field of the received frame is 8808H and the OPCODE field is 0001H. <p>Note: This field should be set to 01 only when the Condition 1 is true, that is, the MAC is programmed to operate in the full-duplex mode and the RFE bit is enabled. Otherwise, the PAUSE frame filtering may be inconsistent. When Condition 1 is false, the PAUSE frames are considered as generic control frames. Therefore, to pass all control frames (including PAUSE control frames) when the full-duplex mode and flow control is not enabled, you should set the PCF field to 10 or 11 (as required by the application).</p>
SAIF	8	rw	<p>SA Inverse Filtering</p> <p>When this bit is set, the Address Check block operates in inverse filtering mode for the SA address comparison. The frames whose SA matches the SA registers are marked as failing the SA Address filter.</p> <p>When this bit is reset, frames whose SA does not match the SA registers are marked as failing the SA Address filter.</p>

Field	Bits	Type	Description
SAF	9	rw	<p>Source Address Filter Enable</p> <p>When this bit is set, the MAC compares the SA field of the received frames with the values programmed in the enabled SA registers. If the comparison matches, then the SA Match bit of RxStatus Word is set high. When this bit is set high and the SA filter fails, the MAC drops the frame.</p> <p>When this bit is reset, the MAC forwards the received frame to the application and with the updated SA Match bit of the RxStatus depending on the SA address comparison.</p>
HPF	10	rw	<p>Hash or Perfect Filter</p> <p>When this bit is set, it configures the address filter to pass a frame if it matches either the perfect filtering or the hash filtering as set by the HMC or HUC bits. When this bit is low and the HUC or HMC bit is set, the frame is passed only if it matches the Hash filter.</p>
Reserved_15_11	[15:11]	r	Reserved
VTFE	16	rw	<p>VLAN Tag Filter Enable</p> <p>When set, this bit enables the MAC to drop VLAN tagged frames that do not match the VLAN Tag comparison.</p> <p>When reset, the MAC forwards all frames irrespective of the match status of the VLAN Tag.</p>
Reserved_19_17	[19:17]	r	Reserved
IPFE	20	r	<p>Layer 3 and Layer 4 Filter Enable</p> <p>When set, this bit enables the MAC to drop frames that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect.</p> <p>When reset, the MAC forwards all frames irrespective of the match status of the Layer 3 and Layer 4 fields.</p> <p>If the Layer 3 and Layer 4 Filtering feature is not selected during core configuration, this bit is reserved (RO with default value).</p>

Field	Bits	Type	Description
DNTU	21	r	<p>Drop non-TCP/UDP over IP Frames</p> <p>When set, this bit enables the MAC to drop the non-TCP or UDP over IP frames. The MAC forward only those frames that are processed by the Layer 4 filter.</p> <p>When reset, this bit enables the MAC to forward all non-TCP or UDP over IP frames.</p>
Reserved_30_22	[30:22]	r	Reserved
RA	31	rw	<p>Receive All</p> <p>When this bit is set, the MAC Receiver module passes all received frames, irrespective of whether they pass the address filter or not, to the Application. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bits in the Receive Status Word.</p> <p>When this bit is reset, the Receiver module passes only those frames to the Application that pass the SA or DA address filter.</p>

HASH_TABLE_HIGH

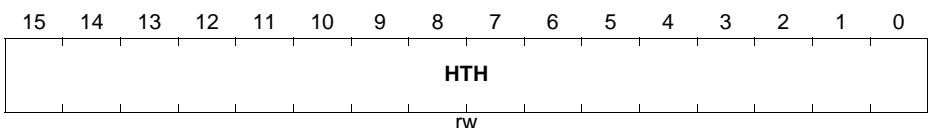
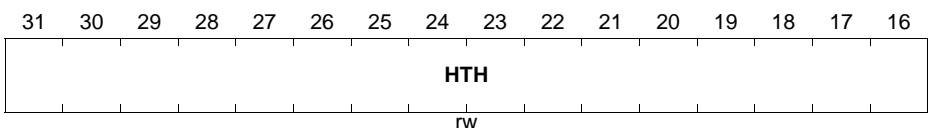
The 64-bit Hash table is used for group address filtering. For hash filtering, the contents of the destination address in the incoming frame is passed through the CRC logic, and the upper 6 bits of the CRC register are used to index the contents of the Hash table. The most significant bit determines the register to be used (Hash Table High or Hash Table Low), and the other 5 bits determine which bit within the register. A hash value of 00000B selects Bit 0 of the selected register, and a value of 11111B selects Bit 31 of the selected register. The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 6 bits from the value obtained in Step 2.

For example, if the DA of the incoming frame is received as 1F52 419C B6AFH (1FH is the first byte received on MII interface), then the internally calculated 6-bit Hash value is 2CH and Bit 12 of Hash Table High register is checked for filtering. If the DA of the incoming frame is received as A00A 9800 0045H, then the calculated 6-bit Hash value is 07H and Bit 7 of Hash Table Low register is checked for filtering. Note: To help you program the hash table, a sample C routine that generates a DA's 6-bit hash is included in the /sample_codes/ directory of your workspace. If the corresponding bit value of the register is 1, the frame is accepted. Otherwise, it is rejected. If the PM (Pass All Multicast) bit is set in the MAC Frame Filter Register, then all multicast frames are accepted regardless of the multicast hash values. If the Hash Table register is configured to be double-synchronized to the MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) of the Hash Table High or Low registers are written. Consecutive writes to these register should be performed only after at least four clock cycles in the destination clock domain when double-synchronization is enabled. The Hash Table High register contains the higher 32 bits of the Hash table.

ETH0_HASH_TABLE_HIGH

Hash Table High Register (8_H) **Reset Value: 0000 0000_H**



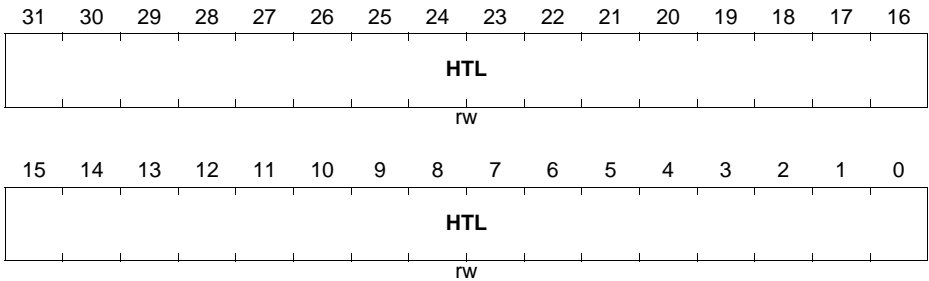
Field	Bits	Type	Description
HTH	[31:0]	rw	Hash Table High This field contains the upper 32 bits of the Hash table.

HASH_TABLE_LOW

The Hash Table Low register contains the lower 32 bits of the Hash table. Both Register 2 and Register 3 are reserved if the Hash Filter Function is disabled or the 128-bit or 256-bit Hash Table is selected during core configuration.

ETH0_HASH_TABLE_LOW

Hash Table Low Register (C_H) **Reset Value: 0000 0000_H**



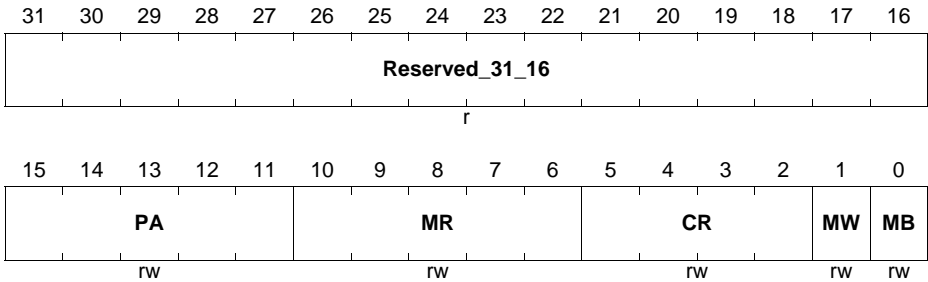
Field	Bits	Type	Description
HTL	[31:0]	rw	Hash Table Low This field contains the lower 32 bits of the Hash table.

GMII_ADDRESS

The MII Address register controls the management cycles to the external PHY through the management interface.

ETH0_GMII_ADDRESS

MII Address Register (10_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
MB	0	rw	<p>MII Busy</p> <p>This bit should read logic 0 before writing to the MII Address and Data registers. During a PHY register access, the software sets this bit to 1 to indicate that a Read or Write access is in progress.</p> <p>The MII Data Register is invalid until this bit is cleared by the MAC. Therefore the MII Data Register should be kept valid until the MAC clears this bit during a PHY Write operation. Similarly for a read operation, the contents of the MII Data Register are not valid until this bit is cleared.</p> <p>The subsequent read or write operation should happen only after the previous operation is complete. Because there is no acknowledgment from the PHY to MAC after a read or write operation is completed, there is no change in the functionality of this bit even when the PHY is not present.</p>
MW	1	rw	<p>MII Write</p> <p>When set, this bit indicates to the PHY that this is a Write operation using the MII Data register. If this bit is not set, it indicates that this is a Read operation, that is, placing the data in the MII Data register.</p>

Field	Bits	Type	Description
CR	[5:2]	rw	<p>CSR Clock Range</p> <p>The CSR Clock Range selection determines the frequency of the MDC clock according to the ETH Clock frequency used in your design. The suggested range of ETH Clock frequency applicable for each value (when Bit[5] = 0) ensures that the MDC clock is approximately between the frequency range 1.0 MHz - 2.5 MHz.</p> <ul style="list-style-type: none"> - 0000B: The frequency of the ETH Clock is 60-100 MHz and the MDC clock is ETH Clock /42. - 0001B: The frequency of the ETH Clock is 100-150 MHz and the MDC clock is ETH Clock /62. - 0010B: The frequency of the ETH Clock is 20-35 MHz and the MDC clock is ETH Clock /16. - 0011B: The frequency of the ETH Clock is 35-60 MHz and the MDC clock is ETH Clock /26. - 0100B: The frequency of the ETH Clock is 150-250 MHz and the MDC clock is ETH Clock /102. - 0100B: The frequency of the ETH Clock is 250-300 MHz and the MDC clock is ETH Clock /124. - 0110B and 0111B: Reserved <p>When Bit 5 is set, you can achieve MDC clock of frequency higher than the IEEE 802.3 specified frequency limit of 2.5 MHz and program a clock divider of lower value. For example, when the ETH Clock is of 100 MHz frequency and you program these bits as 1010B, then the resultant MDC clock is of 12.5 MHz which is outside the limit of IEEE 802.3 specified range. Program the following values only if the interfacing chips support faster MDC clocks:</p> <ul style="list-style-type: none"> - 1000: ETH Clock /4 - 1001: ETH Clock /6 - 1010: ETH Clock /8 - 1011: ETH Clock /10 - 1100: ETH Clock /12 - 1101: ETH Clock /14 - 1110: ETH Clock /16 - 1111:ETH Clock /18
MR	[10:6]	rw	<p>MII Register</p> <p>These bits select the desired MII register in the selected PHY device.</p>

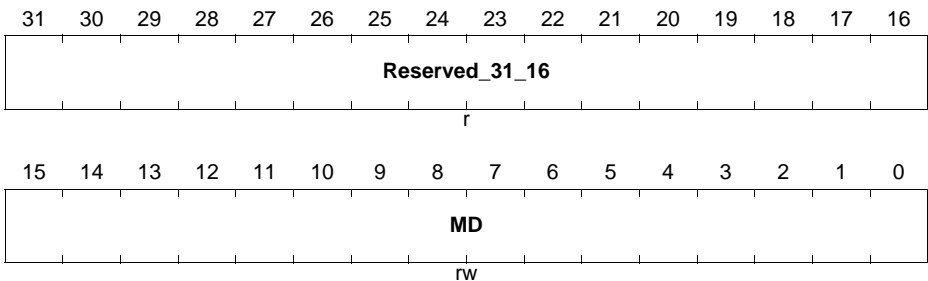
Field	Bits	Type	Description
PA	[15:11]	rw	Physical Layer Address This field indicates which of the 32 possible PHY devices are being accessed.
Reserved_31_16	[31:16]	r	Reserved

GMII_DATA

The MII Data register stores Write data to be written to the PHY register located at the address specified in the MII Address Register. This register also stores the Read data from the PHY register located at the address specified by the MII Address Register.

ETH0_GMII_DATA

MII Data Register (14_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
MD	[15:0]	rw	MII Data This field contains the 16-bit data value read from the PHY or RevMII after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.
Reserved_31_16	[31:16]	r	Reserved

Field	Bits	Type	Description
FCA_BPA	0	rw	<p>Flow Control Busy or Backpressure Activate</p> <p>This bit initiates a Pause Control frame in the full-duplex mode and activates the backpressure function in the half-duplex mode if the TFE bit is set.</p> <p>In the full-duplex mode, this bit should be read as 0 before writing to the Flow Control register. To initiate a Pause control frame, the Application must set this bit to 1. During a transfer of the Control Frame, this bit continues to be set to signify that a frame transmission is in progress. After the completion of Pause control frame transmission, the MAC resets this bit to 1'b0. The Flow Control register should not be written to until this bit is cleared.</p> <p>In the half-duplex mode, when this bit is set (and TFE is set), then backpressure is asserted by the MAC. During backpressure, when the MAC receives a new frame, the transmitter starts sending a JAM pattern resulting in a collision. When the MAC is configured for the full-duplex mode, the BPA is automatically disabled.</p>
TFE	1	rw	<p>Transmit Flow Control Enable</p> <p>In the full-duplex mode, when this bit is set, the MAC enables the flow control operation to transmit Pause frames. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause frames.</p> <p>In half-duplex mode, when this bit is set, the MAC enables the back-pressure operation. When this bit is reset, the back-pressure feature is disabled.</p>
RFE	2	rw	<p>Receive Flow Control Enable</p> <p>When this bit is set, the MAC decodes the received Pause frame and disables its transmitter for a specified (Pause) time. When this bit is reset, the decode function of the Pause frame is disabled.</p>

Field	Bits	Type	Description
UP	3	rw	<p>Unicast Pause Frame Detect</p> <p>When this bit is set, then in addition to the detecting Pause frames with the unique multicast address, the MAC detects the Pause frames with the station's unicast address specified in the MAC Address0 High Register and MAC Address0 Low Register. When this bit is reset, the MAC detects only a Pause frame with the unique multicast address specified in the 802.3x standard.</p>
PLT	[5:4]	rw	<p>Pause Low Threshold</p> <p>This field configures the threshold of the PAUSE timer at which the input flow control signal is checked for automatic retransmission of PAUSE Frame. The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot-times), and PLT = 01, then a second PAUSE frame is automatically transmitted if the flow control signal is asserted at 228 (256 - 28) slot times after the first PAUSE frame is transmitted. The following list provides the threshold values for different values:</p> <ul style="list-style-type: none"> - 00B: The threshold is Pause time minus 4 slot times (PT - 4 slot times). - 01B: The threshold is Pause time minus 28 slot times (PT - 28 slot times). - 10B: The threshold is Pause time minus 144 slot times (PT - 144 slot times). - 11B: The threshold is Pause time minus 256 slot times (PT - 256 slot times). <p>The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the MII interface.</p>
Reserved_6	6	r	Reserved
DZPQ	7	rw	<p>Disable Zero-Quanta Pause</p> <p>When this bit is set, it disables the automatic generation of the Zero-Quanta Pause Control frames on the de-assertion of the flow-control signal from the FIFO layer. When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled.</p>

Field	Bits	Type	Description
Reserved_15_8	[15:8]	r	Reserved
PT	[31:16]	rw	<p>Pause Time</p> <p>This field holds the value to be used in the Pause Time field in the transmit control frame. If the Pause Time bits is configured to be double-synchronized to the MII clock domain, then consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.</p>

VLAN_TAG

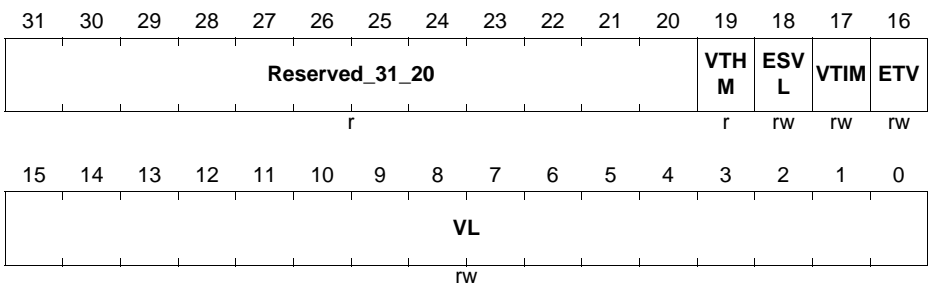
The VLAN Tag register contains the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the receiving frame (Length/Type) with 8100H, and the following two bytes are compared with the VLAN tag. If a match occurs, the MAC sets the received VLAN bit in the receive frame status. The legal length of the frame is increased from 1,518 bytes to 1,522 bytes. If the VLAN Tag register is configured to be double-synchronized to the MII clock domain, then consecutive writes to these register should be performed only after at least four clock cycles in the destination clock domain.

ETH0_VLAN_TAG

VLAN Tag Register

(1C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
VL	[15:0]	rw	<p>VLAN Tag Identifier for Receive Frames</p> <p>This field contains the 802.1Q VLAN tag to identify the VLAN frames and is compared to the 15th and 16th bytes of the frames being received for VLAN frames. The following list describes the bits of this field:</p> <ul style="list-style-type: none"> * Bits [15:13]: User Priority * Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) * Bits[11:0]: VLAN tag's VLAN Identifier (VID) field <p>When the ETV bit is set, only the VID (Bits[11:0]) is used for comparison.</p> <p>If VL (VL[11:0] if ETV is set) is all zeros, the MAC does not check the fifteenth and 16th bytes for VLAN tag comparison, and declares all frames with a Type field value of 8100H or 88A8H as VLAN frames.</p>

Field	Bits	Type	Description
ETV	16	rw	<p>Enable 12-Bit VLAN Tag Comparison</p> <p>When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits 11-0 of VLAN tag are compared with the corresponding field in the received VLAN-tagged frame. Similarly, when enabled, only 12 bits of the VLAN tag in the received frame are used for hash-based VLAN filtering.</p> <p>When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN frame are used for comparison and VLAN hash filtering.</p>
VTIM	17	rw	<p>VLAN Tag Inverse Match Enable</p> <p>When set, this bit enables the VLAN Tag inverse matching. The frames that do not have matching VLAN Tag are marked as matched.</p> <p>When reset, this bit enables the VLAN Tag perfect matching. The frames with matched VLAN Tag are marked as matched.</p>
ESVL	18	rw	<p>Enable S-VLAN</p> <p>When this bit is set, the MAC transmitter and receiver also consider the S-VLAN (Type = 88A8H) frames as valid VLAN tagged frames.</p>
VTHM	19	r	<p>VLAN Tag Hash Table Match Enable</p> <p>When set, the most significant four bits of the VLAN tags CRC are used to index the content of Register 354 [VLAN Hash Table Register]. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the frame matched the VLAN hash table. When Bit 16 (ETV) is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison whereas when ETV is reset, the CRC of the 16-bit VLAN tag is used for comparison.</p> <p>When reset, the VLAN Hash Match operation is not performed.</p>
Reserved_31_20	[31:20]	r	Reserved

VERSION

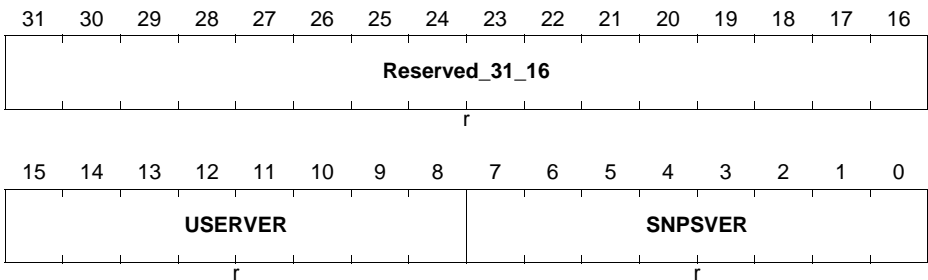
The VERSION registers identifies the version of the ETH. This register contains two bytes: one that Synopsys uses to identify the core release number, and the other that you set during core configuration.

ETH0_VERSION

Version Register

(20_H)

Reset Value: 0000 1037_H



Field	Bits	Type	Description
SNPSVER	[7:0]	r	Synopsys-defined Version (3.7)
USERVER	[15:8]	r	User-defined Version (Configured with the coreConsultant)
Reserved_31_16	[31:16]	r	Reserved

DEBUG

The DEBUG register gives the status of all main modules of the transmit and receive data-paths and the FIFOs. An all-zero status indicates that the MAC is in idle state (and FIFOs are empty) and no activity is going on in the data-paths.

ETH0_DEBUG

Debug Register

(24_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved_31_26						TXS TSF STS	TXF STS	Rese rved _23	TWC STS	TRCSTS	TXP AUS ED	TFCSTS	TPE STS			
r						r	r	r	r	r	r	r	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved_15_10						RXFSTS	Rese rved _7	RRCSTS	RWC STS	Rese rved _3	RFCFCST S	RPE STS				
r						r	r	r	r	r	r	r	r			

Field	Bits	Type	Description
RPESTS	0	r	MAC MII Receive Protocol Engine Status When high, this bit indicates that the MAC MII receive protocol engine is actively receiving data and not in IDLE state.
RFCFCST S	[2:1]	r	MAC Receive Frame Controller FIFO Status When high, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Frame Controller Module.
Reserved_3	3	r	Reserved
RWCSTS	4	r	MTL Rx FIFO Write Controller Active Status When high, this bit indicates that the MTL Rx FIFO Write Controller is active and is transferring a received frame to the FIFO.

Field	Bits	Type	Description
RRCSTS	[6:5]	r	MTL Rx FIFO Read Controller State This field gives the state of the Rx FIFO read Controller: * 00B: IDLE state * 01B: Reading frame data * 10B: Reading frame status (or timestamp) * 11B: Flushing the frame data and status
Reserved_7	7	r	Reserved
RXFSTS	[9:8]	r	MTL Rx FIFO Fill-level Status This field gives the status of the fill-level of the Rx FIFO: * 00B: Rx FIFO Empty * 01B: Rx FIFO fill level is below the flow-control deactivate threshold * 10B: Rx FIFO fill level is above the flow-control activate threshold * 11B: Rx FIFO Full
Reserved_15_10	[15:10]	r	Reserved
TPESTS	16	r	MAC MII Transmit Protocol Engine Status When high, this bit indicates that the MAC MII transmit protocol engine is actively transmitting data and is not in the IDLE state.
TFCSTS	[18:17]	r	MAC Transmit Frame Controller Status This field indicates the state of the MAC Transmit Frame Controller module: * 00B: IDLE state * 01B: Waiting for Status of previous frame or IFG or backoff period to be over * 10B: Generating and transmitting a PAUSE control frame (in the full-duplex mode) * 11B: Transferring input frame for transmission
TXPAUSE D	19	r	MAC transmitter in PAUSE When high, this bit indicates that the MAC transmitter is in the PAUSE condition (in the full-duplex only mode) and hence does not schedule any frame for transmission.

Field	Bits	Type	Description
TRCSTS	[21:20]	r	MTL Tx FIFO Read Controller Status This field indicates the state of the Tx FIFO Read Controller: * 00B: IDLE state * 01B: READ state (transferring data to MAC transmitter) * 10B: Waiting for TxStatus from MAC transmitter * 11B: Writing the received TxStatus or flushing the Tx FIFO
TWCSTS	22	r	MTL Tx FIFO Write Controller Active Status When high, this bit indicates that the MTL Tx FIFO Write Controller is active and transferring data to the Tx FIFO.
Reserved_23	23	r	Reserved
TXFSTS	24	r	MTL Tx FIFO Not Empty Status When high, this bit indicates that the MTL Tx FIFO is not empty and some data is left for transmission.
TXSTSFS TS	25	r	MTL TxStatus FIFO Full Status When high, this bit indicates that the MTL TxStatus FIFO is full. Therefore, the MTL cannot accept any more frames for transmission. This bit is reserved in the ETH-AHB and ETH-DMA configurations.
Reserved_31_26	[31:26]	r	Reserved

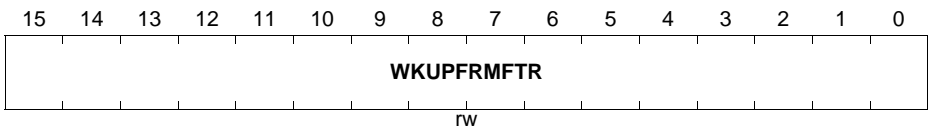
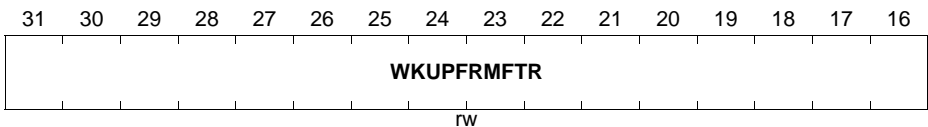
REMOTE_WAKE_UP_FRAME_FILTER

This is the address through which the application writes or reads the remote wake-up frame filter registers (wkupfilter_reg). The wkupfilter_reg register is a pointer to eight wkupfilter_reg registers. The wkupfilter_reg register is loaded by sequentially loading the eight register values. Eight sequential writes to this address (0028H) writes all wkupfilter_reg registers. Similarly, eight sequential reads from this address (0028H) read all wkupfilter_reg registers. This register contains the higher 16 bits of the seventh MAC address.

ETH0_REMOTE_WAKE_UP_FRAME_FILTER

Remote Wake Up Frame Filter Register (28_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
WKUPFR MFTR	[31:0]	rw	Remote Wake-Up Frame Filter

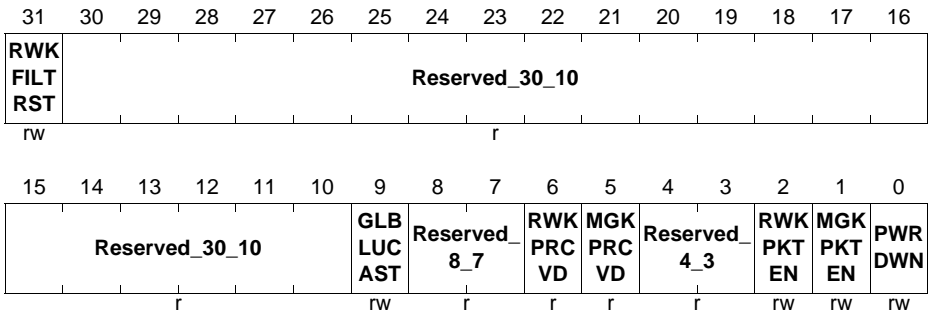
PMT_CONTROL_STATUS

ETH0_PMT_CONTROL_STATUS

PMT Control and Status Register

(2C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PWRDWN	0	rw	<p>Power Down</p> <p>When set, the MAC receiver drops all received frames until it receives the expected magic packet or wake-up frame. This bit is then self-cleared and the power-down mode is disabled. The Software can also clear this bit before the expected magic packet or wake-up frame is received. The frames, received by the MAC after this bit is cleared, are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Wake-Up Frame Enable bit is set high.</p> <p>Note: You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit.</p>
MGKPKTEN	1	rw	<p>Magic Packet Enable</p> <p>When set, enables generation of a power management event because of magic packet reception.</p>
RWKPKTEN	2	rw	<p>Wake-Up Frame Enable</p> <p>When set, enables generation of a power management event because of wake-up frame reception.</p>

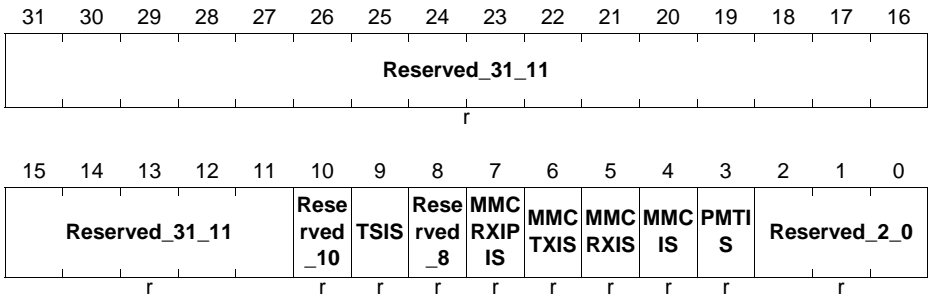
Field	Bits	Type	Description
Reserved_4_3	[4:3]	r	Reserved
MGKPRC VD	5	r	Magic Packet Received When set, this bit indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared by a Read into this register.
RWKPRC VD	6	r	Wake-Up Frame Received When set, this bit indicates the power management event is generated because of the reception of a wake-up frame. This bit is cleared by a Read into this register.
Reserved_8_7	[8:7]	r	Reserved
GLBLUCA ST	9	rw	Global Unicast When set, enables any unicast packet filtered by the MAC (DAF) address recognition to be a wake-up frame.
Reserved_30_10	[30:10]	r	Reserved
RWKFILTER RST	31	rw	Wake-Up Frame Filter Register Pointer Reset When set, resets the remote wake-up frame filter register pointer to 000B. It is automatically cleared after 1 clock cycle.

INTERRUPT_STATUS

The Interrupt Status register identifies the events in the MAC that can generate interrupt.

ETH0_INTERRUPT_STATUS

Interrupt Register (38_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
Reserved_2_0	[2:0]	r	Reserved
PMTIS	3	r	PMT Interrupt Status This bit is set when a Magic packet or Wake-on-LAN frame is received in the power-down mode. This bit is cleared when both PMT_CONTROL_STATUS.MGKPRC VD and PMT_CONTROL_STATUS.RWKPRCVD are cleared because of a read operation to the PMT Control and Status register.
MMCIS	4	r	MMC Interrupt Status This bit is set high when any of the Bits MMCRXIS, MMCTXIS or MMCRXIPIS is set high and cleared only when all of these bits are low.
MMCRXIS	5	r	MMC Receive Interrupt Status This bit is set high when an interrupt is generated in the MMC Receive Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared.
MMCTXIS	6	r	MMC Transmit Interrupt Status This bit is set high when an interrupt is generated in the MMC Transmit Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared.

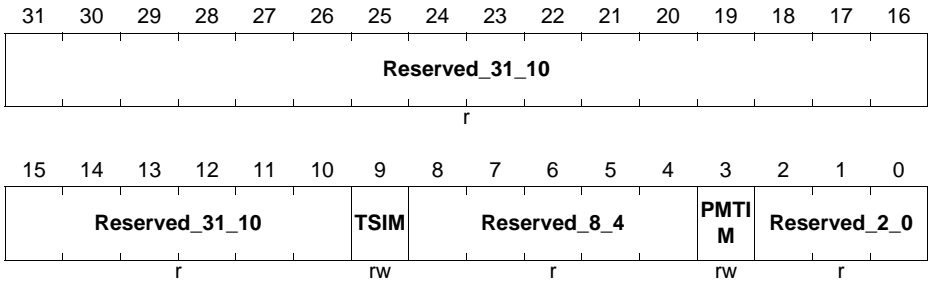
Field	Bits	Type	Description
MMCRXIPIS	7	r	MMC Receive Checksum Offload Interrupt Status This bit is set high when an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared.
Reserved_8	8	r	Reserved
TSIS	9	r	Timestamp Interrupt Status When the Advanced Timestamp feature is enabled, this bit is set when any of the following conditions is true: * The system time value equals or exceeds the value specified in the Target Time High and Low registers. * There is an overflow in the seconds register. This bit is cleared on reading Timestamp STATUS.TSSOVF Register. If default Timestamping is enabled, when set, this bit indicates that the system time value is equal to or exceeds the value specified in the Target Time registers. In this mode, this bit is cleared after the completion of the read of this bit. In all other modes, this bit is reserved.
Reserved_10	10	r	Reserved
Reserved_31_11	[31:11]	r	Reserved

INTERRUPT_MASK

The Interrupt Mask Register bits enable you to mask the interrupt signal because of the corresponding event in the Interrupt Status Register.

ETH0_INTERRUPT_MASK

Interrupt Mask Register (3C_H) **Reset Value: 0000 0000_H**



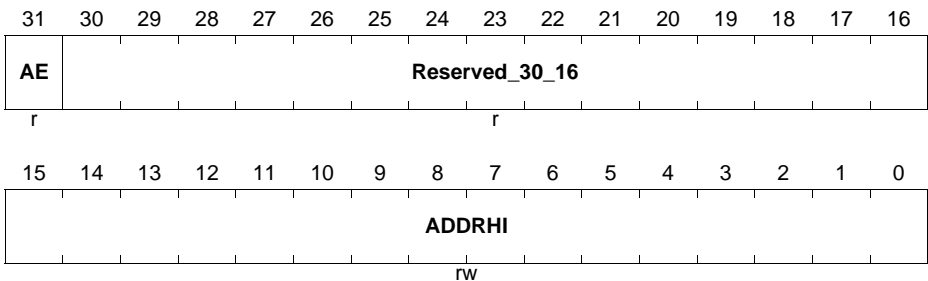
Field	Bits	Type	Description
Reserved_2_0	[2:0]	r	Reserved
PMTIM	3	rw	PMT Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of PMT Interrupt Status bit INTERRUPT_STATUS.PMTIS.
Reserved_8_4	[8:4]	r	Reserved
TSIM	9	rw	Timestamp Interrupt Mask When set, this bit disables the assertion of the interrupt signal because of the setting of Timestamp Interrupt Status bit INTERRUPT_STATUS.TSIS. This bit is valid only when IEEE1588 timestamping is enabled. In all other modes, this bit is reserved.
Reserved_31_10	[31:10]	r	Reserved

MAC_ADDRESS0_HIGH

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 1122 3344 5566H is received (11H in lane 0 of the first column) on the MII as the destination address, then the MacAddress0 Register [47:0] is compared with 6655 4433 2211H. If the MAC address registers are configured to be double-synchronized to the MII clock domains, then the synchronization is triggered only when Bits[31:24] of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH0_MAC_ADDRESS0_HIGH

MAC Address0 High Register (40_H) **Reset Value: 8000 FFFF_H**



Field	Bits	Type	Description
ADDRHI	[15:0]	rw	MAC Address0 [47:32] This field contains the upper 16 bits (47:32) of the first 6-byte MAC address. The MAC uses this field for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.
Reserved_30_16	[30:16]	r	Reserved
AE	31	r	Address Enable This bit is always set to 1.

MAC_ADDRESS0_LOW

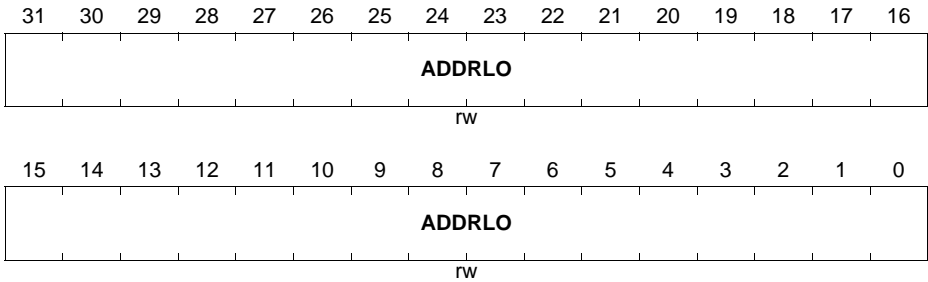
The MAC Address0 Low register holds the lower 32 bits of the first 6-byte MAC address of the station.

ETH0_MAC_ADDRESS0_LOW

MAC Address0 Low Register

(44_H)

Reset Value: FFFF FFFF_H



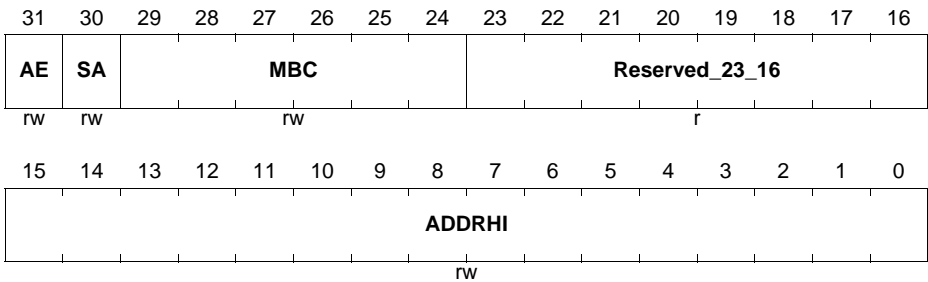
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	MAC Address0 [31:0] This field contains the lower 32 bits of the first 6-byte MAC address. This is used by the MAC for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.

32-bit Register - MAC_ADDRESS1_HIGH

The MAC Address1 High register holds the upper 16 bits of the second 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the MII clock domains, then the synchronization is triggered only when Bits[31:24] of the MAC Address1 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH0_MAC_ADDRESS1_HIGH

MAC Address1 High Register (48_H) **Reset Value: 0000 FFFF_H**



Field	Bits	Type	Description
ADDRHI	[15:0]	rw	MAC Address1 [47:32] This field contains the upper 16 bits (47:32) of the second 6-byte MAC address.
Reserved_23_16	[23:16]	r	Reserved

Field	Bits	Type	Description
MBC	[29:24]	rw	<p>Mask Byte Control</p> <p>These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:</p> <ul style="list-style-type: none"> * Bit 29: MAC_ADDRESS1_HIGH [15:8] * Bit 28: MAC_ADDRESS1_HIGH [7:0] * Bit 27: MAC_ADDRESS1_LOW [31:24] * ... * Bit 24: MAC_ADDRESS1_LOW [7:0] <p>You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.</p>
SA	30	rw	<p>Source Address</p> <p>When this bit is set, the MAC Address1[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address1[47:0] is used to compare with the DA fields of the received frame.</p>
AE	31	rw	<p>Address Enable</p> <p>When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.</p>

MAC_ADDRESS1_LOW

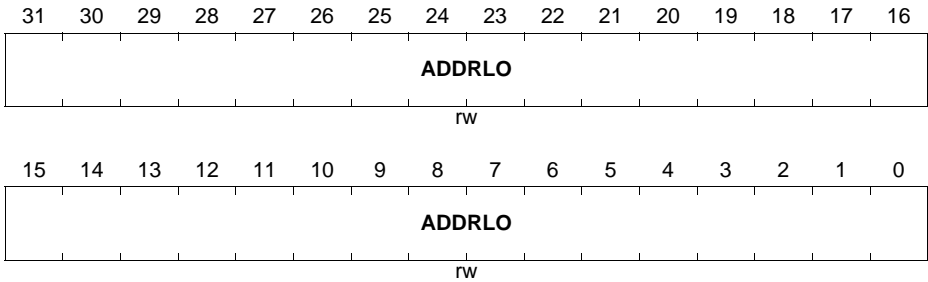
The MAC Address1 Low register holds the lower 32 bits of the second 6-byte MAC address of the station.

ETH0_MAC_ADDRESS1_LOW

MAC Address1 Low Register

(4C_H)

Reset Value: FFFF FFFF_H



Field	Bits	Type	Description
ADDRLO	[31:0]	rw	MAC Address1 [31:0] This field contains the lower 32 bits of the second 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

Field	Bits	Type	Description
SA	30	rw	<p>Source Address</p> <p>When this bit is set, the MAC Address2[47:0] is used to compare with the SA fields of the received frame.</p> <p>When this bit is reset, the MAC Address2[47:0] is used to compare with the DA fields of the received frame.</p>
AE	31	rw	<p>Address Enable</p> <p>When this bit is set, the address filter module uses the third MAC address for perfect filtering.</p> <p>When this bit is reset, the address filter module ignores the address for filtering.</p>

MAC_ADDRESS2_LOW

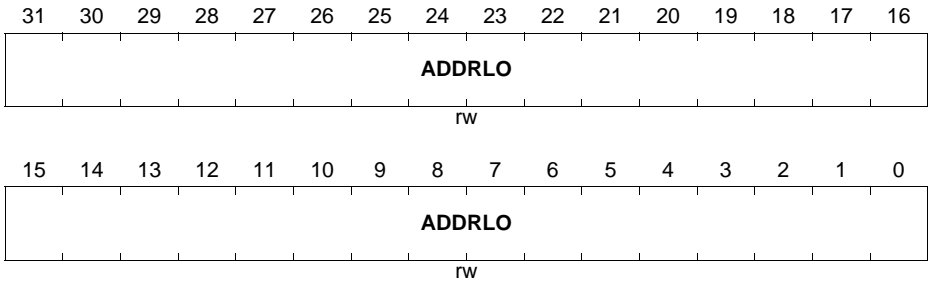
The MAC Address2 Low register holds the lower 32 bits of the third 6-byte MAC address of the station.

ETH0_MAC_ADDRESS2_LOW

MAC Address2 Low Register

(54_H)

Reset Value: FFFF FFFF_H



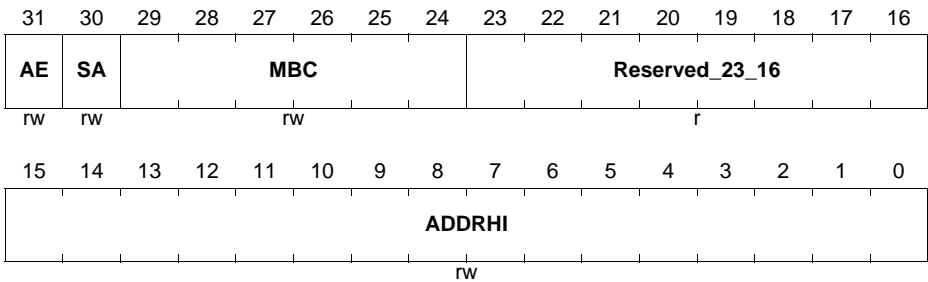
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	MAC Address2 [31:0] This field contains the lower 32 bits of the third 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

MAC_ADDRESS3_HIGH

The MAC Address3 High register holds the upper 16 bits of the fourth 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the MII clock domains, then the synchronization is triggered only when Bits[31:24] of the MAC Address3 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address3 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH0_MAC_ADDRESS3_HIGH

MAC Address3 High Register (58_H) **Reset Value: 0000 FFFF_H**



Field	Bits	Type	Description
ADDRHI	[15:0]	rw	MAC Address3 [47:32] This field contains the upper 16 bits (47:32) of the fourth 6-byte MAC address.
Reserved_23_16	[23:16]	r	Reserved
MBC	[29:24]	rw	Mask Byte Control These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address3 registers. Each bit controls the masking of the bytes as follows: * Bit 29: MAC_ADDRESS1_HIGH [15:8] * Bit 28: MAC_ADDRESS1_HIGH [7:0] * Bit 27: MAC_ADDRESS1_HIGH [31:24] * ... * Bit 24: MAC_ADDRESS1_HIGH [7:0]

Field	Bits	Type	Description
SA	30	rw	<p>Source Address</p> <p>When this bit is set, the MAC Address3[47:0] is used to compare with the SA fields of the received frame.</p> <p>When this bit is reset, the MAC Address3[47:0] is used to compare with the DA fields of the received frame.</p>
AE	31	rw	<p>Address Enable</p> <p>When this bit is set, the address filter module uses the fourth MAC address for perfect filtering.</p> <p>When this bit is reset, the address filter module ignores the address for filtering.</p>

MAC_ADDRESS3_LOW

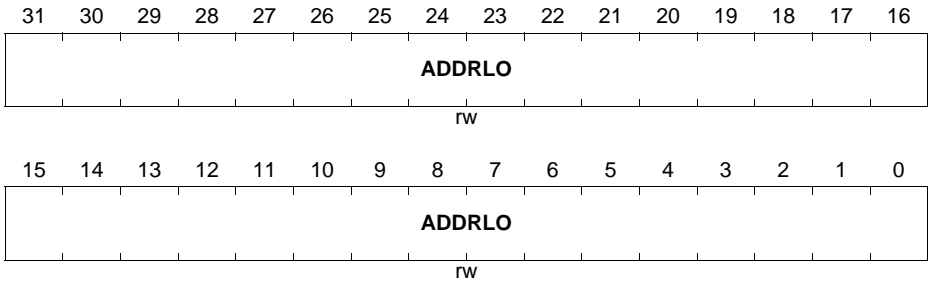
The MAC Address3 Low register holds the lower 32 bits of the fourth 6-byte MAC address of the station.

ETH0_MAC_ADDRESS3_LOW

MAC Address3 Low Register

(5C_H)

Reset Value: FFFF FFFF_H



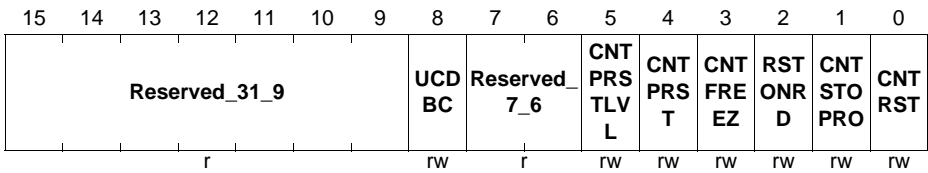
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	MAC Address3 [31:0] This field contains the lower 32 bits of the fourth 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

MMC_CONTROL

The MMC Control register establishes the operating mode of the management counters. Note: The bit 0 (Counters Reset) has higher priority than bit 4 (Counter Preset). Therefore, when the Software tries to set both bits in the same write cycle, all counters are cleared and the bit 4 is not set.

ETH0_MMC_CONTROL

MMC Control Register (100_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
CNTRST	0	rw	Counters Reset When this bit is set, all counters are reset. This bit is cleared automatically after one clock cycle.
CNTSTOP RO	1	rw	Counters Stop Rollover When this bit is set, after reaching maximum value, the counter does not roll over to zero.
RSTONRD	2	rw	Reset on Read When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (bits[7:0]) is read.
CNTFREE Z	3	rw	MMC Counter Freeze When this bit is set, it freezes all MMC counters to their current value. Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received frame. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.

Field	Bits	Type	Description
CNTPRST	4	rw	Counters Preset When this bit is set, all counters are initialized or preset to almost full or almost half according to bit 5. This bit is cleared automatically after 1 clock cycle. This bit, along with bit 5, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.
CNTPRST LVL	5	rw	Full-Half Preset When low and bit 4 is set, all MMC counters get preset to almost-half value. All octet counters get preset to 7FFF F800H (half - 2KBytes) and all frame-counters gets preset to 7FFF FFF0H (half - 16). When this bit is high and bit 4 is set, all MMC counters get preset to almost-full value. All octet counters get preset to FFFF F800H (full - 2KBytes) and all frame-counters gets preset to FFFF FFF0H (full - 16). For 16-bit counters, the almost-half preset values are 7800H and 7FF0H for the respective octet and frame counters. Similarly, the almost-full preset values for the 16-bit counters are F800H and FFF0H.
Reserved_7_6	[7:6]	r	Reserved
UCDBC	8	rw	Update MMC Counters for Dropped Broadcast Frames When set, this bit enables MAC to update all the related MMC Counters for Broadcast frames dropped due to setting of MAC_Filter.DBF bit. When reset, MMC Counters are not updated for dropped Broadcast frames.
Reserved_31_9	[31:9]	r	Reserved

Field	Bits	Type	Description
RXGOCTIS	2	r	MMC Receive Good Octet Counter Interrupt Status. This bit is set when the RX_OCTET_COUNT_GOOD counter reaches half of the maximum value or the maximum value.
RXBCGFIS	3	r	MMC Receive Broadcast Good Frame Counter Interrupt Status. This bit is set when the RX_BROADCAST_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.
RXMGFIS	4	r	MMC Receive Multicast Good Frame Counter Interrupt Status This bit is set when the RX_MULTICAST_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.
RXRCERFIS	5	r	MMC Receive CRC Error Frame Counter Interrupt Status This bit is set when the RX_CRC_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXALGNERFIS	6	r	MMC Receive Alignment Error Frame Counter Interrupt Status This bit is set when the RX_ALIGNMENT_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXRUNTFIS	7	r	MMC Receive Runt Frame Counter Interrupt Status This bit is set when the RX_RUNT_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXJABERFIS	8	r	MMC Receive Jabber Error Frame Counter Interrupt Status This bit is set when the RX_JABBER_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXUSIZEGFIS	9	r	MMC Receive Undersize Good Frame Counter Interrupt Status This bit is set when the RX_UNDERSIZE_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXOSIZE GFIS	10	r	MMC Receive Oversize Good Frame Counter Interrupt Status This bit is set when the RX_OVERSIZE_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.
RX64OCT GBFIS	11	r	MMC Receive 64 Octet Good Bad Frame Counter Interrupt Status This bit is set when the RX_64OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RX65T127 OCTGBFIS	12	r	MMC Receive 65 to 127 Octet Good Bad Frame Counter Interrupt Status This is set when the RX_65TO127OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RX128T255 OCTGBFIS	13	r	MMC Receive 128 to 255 Octet Good Bad Frame Counter Interrupt Status This bit is set when the RX_128TO255OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RX256T511 OCTGBFIS	14	r	MMC Receive 256 to 511 Octet Good Bad Frame Counter Interrupt Status This bit is set when the RX_256TO511OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RX512T1023 OCTGBFIS	15	r	MMC Receive 512 to 1023 Octet Good Bad Frame Counter Interrupt Status This bit is set when the RX_512TO1023OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RX1024TM AXOCTGB FIS	16	r	MMC Receive 1024 to Maximum Octet Good Bad Frame Counter Interrupt Status This bit is set when the RX_1024TOMAXOCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RXUCGFIS	17	r	MMC Receive Unicast Good Frame Counter Interrupt Status This bit is set when the rxunicastframes_gb counter reaches half of the maximum value or the maximum value.
RXLENER FIS	18	r	MMC Receive Length Error Frame Counter Interrupt Status This bit is set when the RX_LENGTH_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXORAN GEFIS	19	r	MMC Receive Out Of Range Error Frame Counter Interrupt Status This bit is set when the RX_OUT_OF_RANGE_TYPE_FRAMES counter reaches half of the maximum value or the maximum value.
RXPAUSFIS	20	r	MMC Receive Pause Frame Counter Interrupt Status This bit is set when the rxpauseframe counter reaches half of the maximum value or the maximum value.
RXFOVFIS	21	r	MMC Receive FIFO Overflow Frame Counter Interrupt Status This bit is set when the RX_FIFO_OVERFLOW_FRAMES counter reaches half of the maximum value or the maximum value.
RXVLANG BFIS	22	r	MMC Receive VLAN Good Bad Frame Counter Interrupt Status This bit is set when the RX_VLAN_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXWDOG FIS	23	r	MMC Receive Watchdog Error Frame Counter Interrupt Status This bit is set when the RX_WATCHDOG_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXRCVER RFIS	24	r	MMC Receive Error Frame Counter Interrupt Status This bit is set when the rxrcverror counter reaches half of the maximum value or the maximum value.
RXCTRLFIS	25	r	MMC Receive Control Frame Counter Interrupt Status This bit is set when the rxctrlframes_g counter reaches half of the maximum value or the maximum value.
Reserved_31_26	[31:26]	r	Reserved

MMC_TRANSMIT_INTERRUPT

The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half of their maximum values (8000 0000H for 32-bit counter and 8000H for 16-bit counter), and the maximum values (FFFF FFFFH for 32-bit counter and FFFFH for 16-bit counter). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Transmit Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

ETH0_MMC_TRANSMIT_INTERRUPT

MMC Transmit Interrupt Register (108_H) **Reset Value: 0000 0000_H**

Reserved_31_26						TXO SIZE GFIS	TXV LAN GFIS	TXP AUS FIS	TXE XDE FFIS	TXG FRMI S	TXG OCTI S	TXC ARE RFIS	TXE XCO LFIS	TXL ATC OLFI S	TXD EFFI S
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Reserved_15_10						TX10 24T MAX OCT GBFI	TX51 2T10 23O CTG BFIS	TX25 6T51 1OC TGB FIS	TX12 8T25 5OC TGB FIS	TX65 T127 OCT GBFI S	TX64 OCT GBFI S	TXM CGFI S	TXB CGFI S	TXG BFR MIS	TXG BOC TIS
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
TXGBOCTIS	0	r	MMC Transmit Good Bad Octet Counter Interrupt Status This bit is set when the TX_OCTET_COUNT_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TXGBFRMIS	1	r	MMC Transmit Good Bad Frame Counter Interrupt Status This bit is set when the TX_FRAME_COUNT_GOOD_BAD counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TXBCGFIS	2	r	MMC Transmit Broadcast Good Frame Counter Interrupt Status This bit is set when the TX_BROADCAST_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.
TXMCGFIS	3	r	MMC Transmit Multicast Good Frame Counter Interrupt Status This bit is set when the TX_MULTICAST_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.
TX64OCTGBFIS	4	r	MMC Transmit 64 Octet Good Bad Frame Counter Interrupt Status. This bit is set when the TX_64OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TX65T127OCTGBFIS	5	r	MMC Transmit 65 to 127 Octet Good Bad Frame Counter Interrupt Status This bit is set when the TX_65TO127OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TX128T255OCTGBFIS	6	r	MMC Transmit 128 to 255 Octet Good Bad Frame Counter Interrupt Status This bit is set when the TX_128TO255OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TX256T511OCTGBFIS	7	r	MMC Transmit 256 to 511 Octet Good Bad Frame Counter Interrupt Status This bit is set when the TX_256TO511OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TX512T10 23OCTGB FIS	8	r	MMC Transmit 512 to 1023 Octet Good Bad Frame Counter Interrupt Status This bit is set when the TX_512TO1023OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TX1024TM AXOCTGB FIS	9	r	MMC Transmit 1024 to Maximum Octet Good Bad Frame Counter Interrupt Status This bit is set when the TX_1024TOMAXOCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TXUCGBF IS	10	r	MMC Transmit Unicast Good Bad Frame Counter Interrupt Status This bit is set when the TX_UNICAST_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TXMCGBF IS	11	r	MMC Transmit Multicast Good Bad Frame Counter Interrupt Status This bit is set when the TX_MULTICAST_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TXBCGBF IS	12	r	MMC Transmit Broadcast Good Bad Frame Counter Interrupt Status This bit is set when the TX_BROADCAST_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TXUFLOW ERFIS	13	r	MMC Transmit Underflow Error Frame Counter Interrupt Status This bit is set when the TX_UNDERFLOW_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TXSCOLG FIS	14	r	MMC Transmit Single Collision Good Frame Counter Interrupt Status This bit is set when the TX_SINGLE_COLLISION_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.
TXMCOLG FIS	15	r	MMC Transmit Multiple Collision Good Frame Counter Interrupt Status This bit is set when the TX_MULTIPLE_COLLISION_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.
TXDEFFIS	16	r	MMC Transmit Deferred Frame Counter Interrupt Status This bit is set when the TX_DEFERRED_FRAMES counter reaches half of the maximum value or the maximum value.
TXLATCO LFIS	17	r	MMC Transmit Late Collision Frame Counter Interrupt Status This bit is set when the TX_LATE_COLLISION_FRAMES counter reaches half of the maximum value or the maximum value.
TXEXCOL FIS	18	r	MMC Transmit Excessive Collision Frame Counter Interrupt Status This bit is set when the txexcesscol counter reaches half of the maximum value or the maximum value.
TXCARER FIS	19	r	MMC Transmit Carrier Error Frame Counter Interrupt Status This bit is set when the TX_CARRIER_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
TXGOCTI S	20	r	MMC Transmit Good Octet Counter Interrupt Status This bit is set when the TX_OCTET_COUNT_GOOD counter reaches half of the maximum value or the maximum value.
TXGFRMI S	21	r	MMC Transmit Good Frame Counter Interrupt Status This bit is set when the TX_FRAME_COUNT_GOOD counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TXEXDEF FIS	22	r	MMC Transmit Excessive Deferral Frame Counter Interrupt Status This bit is set when the TX_EXCESSIVE_DEFERRAL_ERROR counter reaches half of the maximum value or the maximum value.
TXPAUSFIS	23	r	MMC Transmit Pause Frame Counter Interrupt Status This bit is set when the txpauseframeserror counter reaches half of the maximum value or the maximum value.
TXVLANG FIS	24	r	MMC Transmit VLAN Good Frame Counter Interrupt Status This bit is set when the TX_VLAN_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.
TXOSIZEG FIS	25	r	MMC Transmit Oversize Good Frame Counter Interrupt Status This bit is set when the txoversize_g counter reaches half of the maximum value or the maximum value.
Reserved_31_26	[31:26]	r	Reserved

MMC_RECEIVE_INTERRUPT_MASK

ETH0_MMC_RECEIVE_INTERRUPT_MASK

MMC Receive Interrupt Mask Register (10C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved_31_26						RXC TRL FIM	RXR CVE RRFI M	RXW DOG FIM	RXV LAN GBFI M	RXF OVFI M	RXP AUS FIM	RXO RAN GEFI M	RXL ENE RFIM	RXU CGFI M	RX1 024T MAX OCT GBFI
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX5 12T1 023O CTG BFIM	RX2 56T5 11O CTG BFIM	RX1 28T2 55O CTG BFIM	RX6 5T12 7OC TGB FIM	RX6 4OC TGB FIM	RXO SIZE GFI M	RXU SIZE GFI M	RXJ ABE RFIM	RXR UNT FIM	RXA LGN ERFI M	RXC RCE RFIM	RXM CGFI M	RXB CGFI M	RXG OCTI M	RXG BOC TIM	RXG BFR MIM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
RXGBFRM IM	0	rw	MMC Receive Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_FRAMES_COUNT_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RXGBOCT IM	1	rw	MMC Receive Good Bad Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RX_OCTET_COUNT_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RXGOCTI M	2	rw	MMC Receive Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RX_OCTET_COUNT_GOOD counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXBCGFI M	3	rw	MMC Receive Broadcast Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_BROADCAST_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.
RXMGFI M	4	rw	MMC Receive Multicast Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_MULTICAST_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.
RXRCRER FIM	5	rw	MMC Receive CRC Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_CRC_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXALGNE RFIM	6	rw	MMC Receive Alignment Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_ALIGNMENT_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXRUNTFI M	7	rw	MMC Receive Runt Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_RUNT_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXJABER FIM	8	rw	MMC Receive Jabber Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_JABBER_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXUSIZE GFIM	9	rw	MMC Receive Undersize Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_UNDERSIZE_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.
RXOSIZE GFIM	10	rw	MMC Receive Oversize Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_OVERSIZE_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RX64OCTGBFIM	11	rw	MMC Receive 64 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_64OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RX65T127OCTGBFIM	12	rw	MMC Receive 65 to 127 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_65TO127OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RX128T255OCTGBFIM	13	rw	MMC Receive 128 to 255 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_128TO255OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RX256T511OCTGBFIM	14	rw	MMC Receive 256 to 511 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_256TO511OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RX512T1023OCTGBFIM	15	rw	MMC Receive 512 to 1023 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_512TO1023OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RX1024TMAXOCTGBFIM	16	rw	MMC Receive 1024 to Maximum Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_1024TOMAXOCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXUCGFIM	17	rw	MMC Receive Unicast Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_UNICAST_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.
RXLENERFIM	18	rw	MMC Receive Length Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_LENGTH_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXORAN GEFIM	19	rw	MMC Receive Out Of Range Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_OUT_OF_RANGE_TYPE_FRAMES counter reaches half of the maximum value or the maximum value.
RXPAUSFIM	20	rw	MMC Receive Pause Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_PAUSE_FRAMES counter reaches half of the maximum value or the maximum value.
RXFOVFI M	21	rw	MMC Receive FIFO Overflow Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_FIFO_OVERFLOW_FRAMES counter reaches half of the maximum value or the maximum value.
RXVLANG BFIM	22	rw	MMC Receive VLAN Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RX_VLAN_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
RXWDOG FIM	23	rw	MMC Receive Watchdog Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxwatchdog counter reaches half of the maximum value or the maximum value.
RXRCVER RFIM	24	rw	MMC Receive Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxrcverror error counter reaches half the maximum value, and also when it reaches the maximum value.

Field	Bits	Type	Description
RXCTRLFI M	25	rw	MMC Receive Control Frame Counter Interrupt Mask Setting this bit masks the interrupt when the rxctrlframes counter reaches half the maximum value, and also when it reaches the maximum value.
Reserved_ 31_26	[31:26]	r	Reserved

MMC_TRANSMIT_INTERRUPT_MASK

The MMC Transmit Interrupt Mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or maximum value. This register is 32-bits wide.

ETH0_MMC_TRANSMIT_INTERRUPT_MASK

MMC Transmit Interrupt Mask Register (110_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved_31_26						TXO SIZE GFI M	TXV LAN GFI M	TXP AUS FIM	TXE XDE FFIM	TXG FRMI M	TXG OCTI M	TXC ARE RFIM	TXE XCO LFIM	TXL ATC OLFI M	TXD EFFI M
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXM COL GFI M	TXS COL GFI M	TXU FLO WER FIM	TXB CGB FIM	TXM CGB FIM	TXU CGB FIM	TX10 24T MAX OCT GBFI	TX51 2T10 23O CTG BFIM	TX25 6T51 10C TGB FIM	TX12 8T25 50C TGB FIM	TX65 T127 OCT GBFI M	TXM CGFI M	TXB CGFI M	TXG BFR MIM	TXG BOC TIM	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TXGBOCTIM	0	rw	MMC Transmit Good Bad Octet Counter Interrupt Mask Setting this bit masks the interrupt when the TX_OCTET_COUNT_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TXGBFRMIM	1	rw	MMC Transmit Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_FRAME_COUNT_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TXBCGFI M	2	rw	MMC Transmit Broadcast Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_BROADCAST_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TXMCGFIM	3	rw	MMC Transmit Multicast Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_MULTICAST_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.
TX64OCTGBFIM	4	rw	MMC Transmit 64 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_64OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TX65T127OCTGBFIM	5	rw	MMC Transmit 65 to 127 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_65TO127OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TX128T255OCTGBFIM	6	rw	MMC Transmit 128 to 255 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_128TO255OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TX256T511OCTGBFIM	7	rw	MMC Transmit 256 to 511 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_256TO511OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TX512T1023OCTGBFIM	8	rw	MMC Transmit 512 to 1023 Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_512TO1023OCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TX1024TM AXOCTGB FIM	9	rw	MMC Transmit 1024 to Maximum Octet Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_1024TOMAXOCTETS_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TXUCGBF IM	10	rw	MMC Transmit Unicast Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_UNICAST_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TXMCGBF IM	11	rw	MMC Transmit Multicast Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_MULTICAST_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TXBCGBF IM	12	rw	MMC Transmit Broadcast Good Bad Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_BROADCAST_FRAMES_GOOD_BAD counter reaches half of the maximum value or the maximum value.
TXUFLOW ERFIM	13	rw	MMC Transmit Underflow Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_UNDERFLOW_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
TXSCOLG FIM	14	rw	MMC Transmit Single Collision Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_SINGLE_COLLISION_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TXMCOLG FIM	15	rw	MMC Transmit Multiple Collision Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_MULTIPLE_COLLISION_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.
TXDEFFIM	16	rw	MMC Transmit Deferred Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_DEFERRED_FRAMES counter reaches half of the maximum value or the maximum value.
TXLATCO L FIM	17	rw	MMC Transmit Late Collision Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_LATE_COLLISION_FRAMES counter reaches half of the maximum value or the maximum value.
TXEXCOL FIM	18	rw	MMC Transmit Excessive Collision Frame Counter Interrupt Mask Setting this bit masks the interrupt when the txexcesscol counter reaches half of the maximum value or the maximum value.
TXCARER FIM	19	rw	MMC Transmit Carrier Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_CARRIER_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
TXGOCTI M	20	rw	MMC Transmit Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the TX_OCTET_COUNT_GOOD counter reaches half of the maximum value or the maximum value.
TXGFRMI M	21	rw	MMC Transmit Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_FRAME_COUNT_GOOD counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TXEXDEF FIM	22	rw	MMC Transmit Excessive Deferral Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_EXCESSIVE_DEFERRAL_ERROR counter reaches half of the maximum value or the maximum value.
TXPAUSFI M	23	rw	MMC Transmit Pause Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_PAUSE_FRAMES counter reaches half of the maximum value or the maximum value.
TXVLANG FIM	24	rw	MMC Transmit VLAN Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the TX_VLAN_FRAMES_GOOD counter reaches half of the maximum value or the maximum value.
TXOSIZEG FIM	25	rw	MMC Transmit Oversize Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the txoversize_g counter reaches half of the maximum value or the maximum value.
Reserved_31_26	[31:26]	r	Reserved

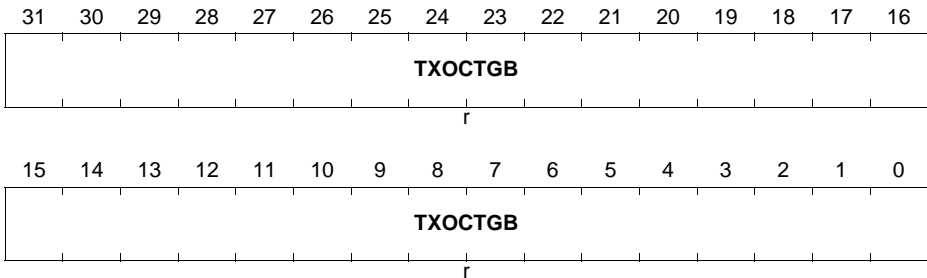
TX_OCTET_COUNT_GOOD_BAD

This register maintains the number of bytes transmitted in good and bad frames exclusive of preamble and retried bytes.

ETH0_TX_OCTET_COUNT_GOOD_BAD

Transmit Octet Count for Good and Bad Frames Register (114_H)
0000 0000_H

Reset Value:



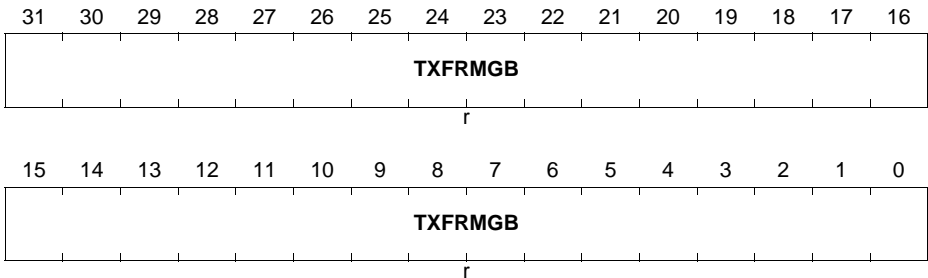
Field	Bits	Type	Description
TXOCTGB	[31:0]	r	This field indicates the number of bytes transmitted in good and bad frames exclusive of preamble and retried bytes.

TX_FRAME_COUNT_GOOD_BAD

This register maintains the number of good and bad frames transmitted, exclusive of retried frames.

ETH0_TX_FRAME_COUNT_GOOD_BAD

Transmit Frame Count for Good and Bad Frames Register (118_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TXFRMGB	[31:0]	r	This field indicates the number of good and bad frames transmitted, exclusive of retried frames

TX_BROADCAST_FRAMES_GOOD

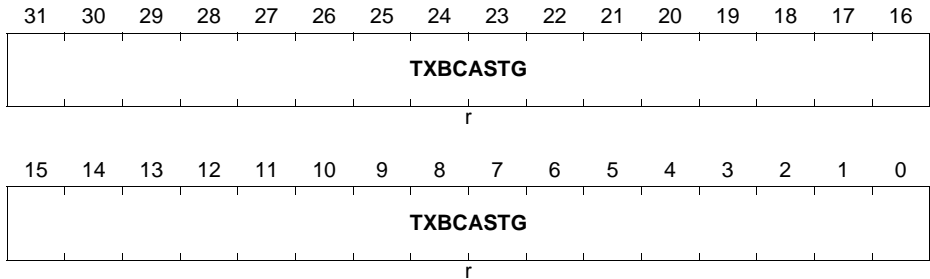
This register maintains the number of transmitted good broadcast frames.

ETH0_TX_BROADCAST_FRAMES_GOOD

Transmit Frame Count for Good Broadcast Frames (11C_H)

Reset Value: 0000

0000_H



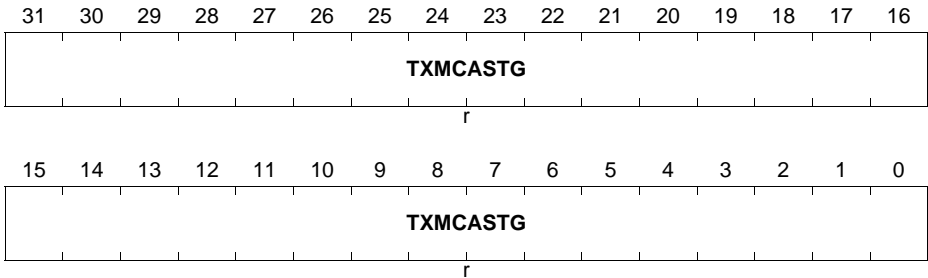
Field	Bits	Type	Description
TXBCASTG	[31:0]	r	This field indicates the number of transmitted good broadcast frames.

TX_MULTICAST_FRAMES_GOOD

This register maintains the number of transmitted good multicast frames.

ETH0_TX_MULTICAST_FRAMES_GOOD

Transmit Frame Count for Good Multicast Frames (120_H) Reset Value: 0000 0000_H



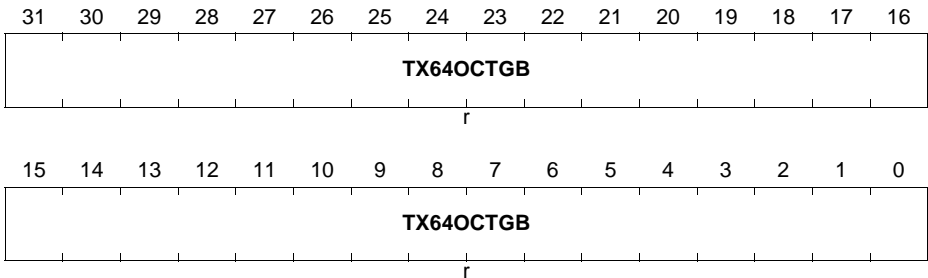
Field	Bits	Type	Description
TXMCASTG	[31:0]	r	This field indicates the number of transmitted good multicast frames.

TX_64OCTETS_FRAMES_GOOD_BAD

This register maintains the number of transmitted good and bad frames with length of 64 bytes, exclusive of preamble and retried frames.

ETH0_TX_64OCTETS_FRAMES_GOOD_BAD

Transmit Octet Count for Good and Bad 64 Byte Frames (124_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
TX64OCTGB	[31:0]	r	This field indicates the number of transmitted good and bad frames with length of 64 bytes, exclusive of preamble and retried frames.

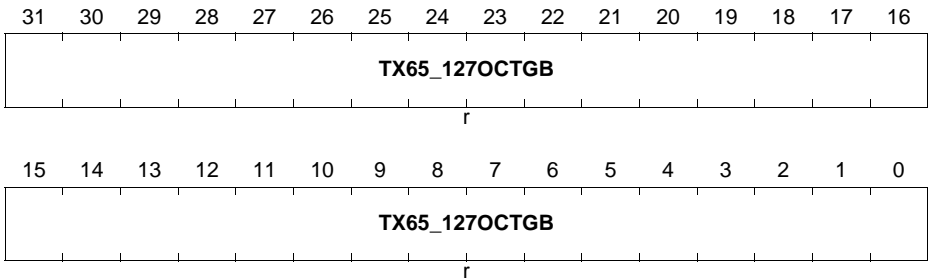
TX_65TO127OCTETS_FRAMES_GOOD_BAD

This register maintains the number of transmitted good and bad frames with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.

ETH0_TX_65TO127OCTETS_FRAMES_GOOD_BAD

Transmit Octet Count for Good and Bad 65 to 127 Bytes Frames (128_H) Reset

Value: 0000 0000_H



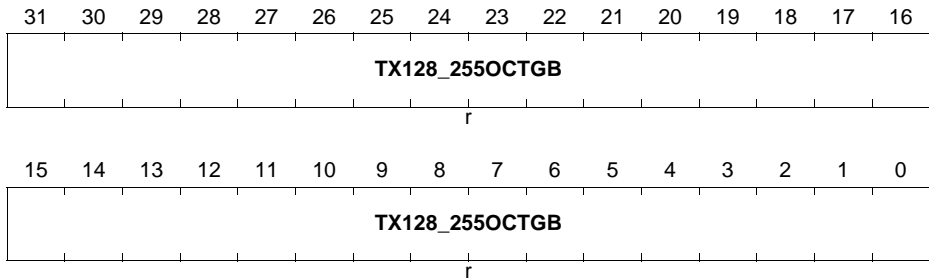
Field	Bits	Type	Description
TX65_127 OCTGB	[31:0]	r	This field indicates the number of transmitted good and bad frames with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.

TX_128TO255OCTETS_FRAMES_GOOD_BAD

This register maintains the number of transmitted good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames.

ETH0_TX_128TO255OCTETS_FRAMES_GOOD_BAD

Transmit Octet Count for Good and Bad 128 to 255 Bytes Frames (12C_H) **Reset Value: 0000 0000_H**



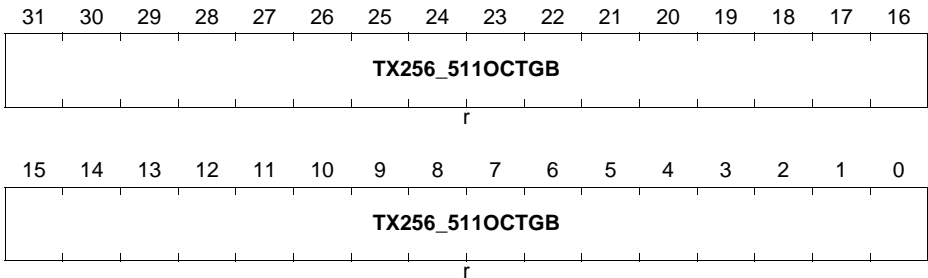
Field	Bits	Type	Description
TX128_255OCTGB	[31:0]	r	This field indicates the number of transmitted good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames.

TX_256TO511OCTETS_FRAMES_GOOD_BAD

This register maintains the number of transmitted good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.

ETH0_TX_256TO511OCTETS_FRAMES_GOOD_BAD

Transmit Octet Count for Good and Bad 256 to 511 Bytes Frames(130_H) Reset Value: 0000 0000_H



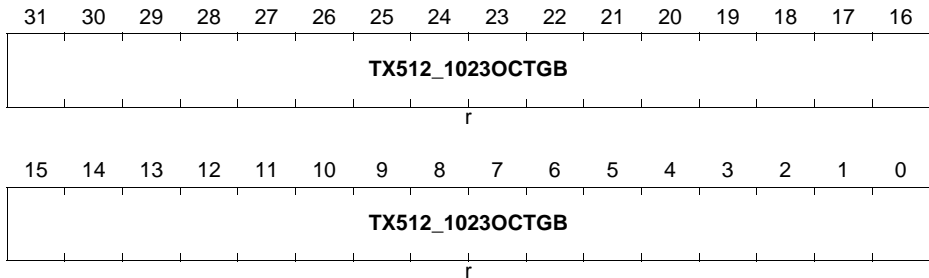
Field	Bits	Type	Description
TX256_511OCTGB	[31:0]	r	This field indicates the number of transmitted good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.

TX_512TO1023OCTETS_FRAMES_GOOD_BAD

This register maintains the number of transmitted good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble and retried frames.

ETH0_TX_512TO1023OCTETS_FRAMES_GOOD_BAD

Transmit Octet Count for Good and Bad 512 to 1023 Bytes Frames(134_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
TX512_1023OCTGB	[31:0]	r	This field indicates the number of transmitted good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble and retried frames.

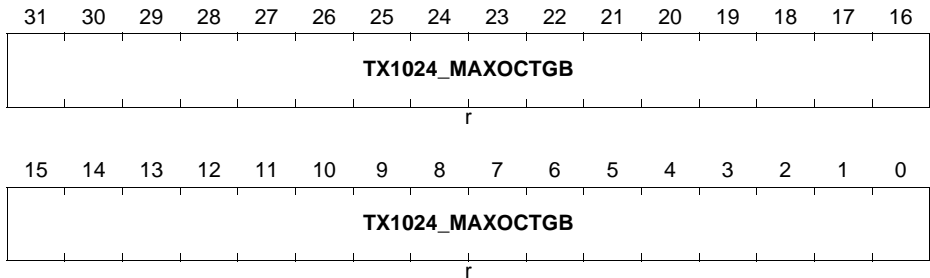
TX_1024TOMAXOCTETS_FRAMES_GOOD_BAD

This register maintains the number of transmitted good and bad frames with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.

ETH0_TX_1024TOMAXOCTETS_FRAMES_GOOD_BAD

Transmit Octet Count for Good and Bad 1024 to Maxsize Bytes Frames(138_H)

Reset Value: 0000 0000_H



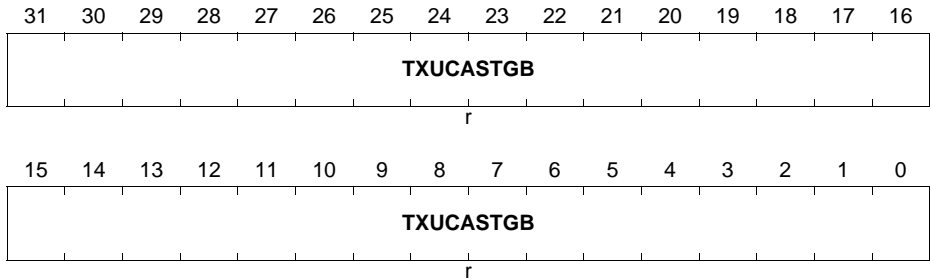
Field	Bits	Type	Description
TX1024_M AXOCTGB	[31:0]	r	This field indicates the number of good and bad frames transmitted with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.

TX_UNICAST_FRAMES_GOOD_BAD

This register maintains the number of transmitted good and bad unicast frames.

ETH0_TX_UNICAST_FRAMES_GOOD_BAD

Transmit Frame Count for Good and Bad Unicast Frames (13C_H) **Reset Value: 0000 0000_H**



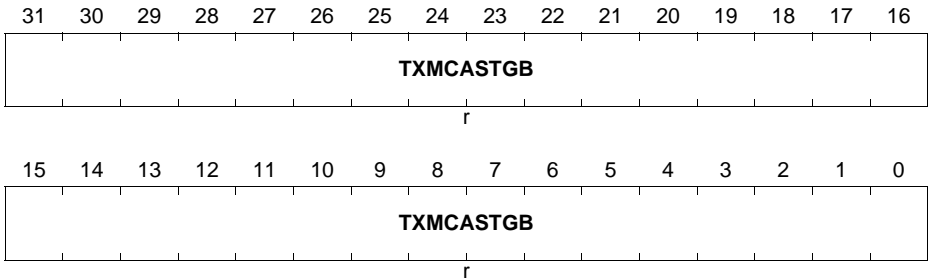
Field	Bits	Type	Description
TXUCAST GB	[31:0]	r	This field indicates the number of transmitted good and bad unicast frames.

TX_MULTICAST_FRAMES_GOOD_BAD

This register maintains the number of transmitted good and bad multicast frames.

ETH0_TX_MULTICAST_FRAMES_GOOD_BAD

Transmit Frame Count for Good and Bad Multicast Frames(140_H) **Reset Value: 0000 0000_H**



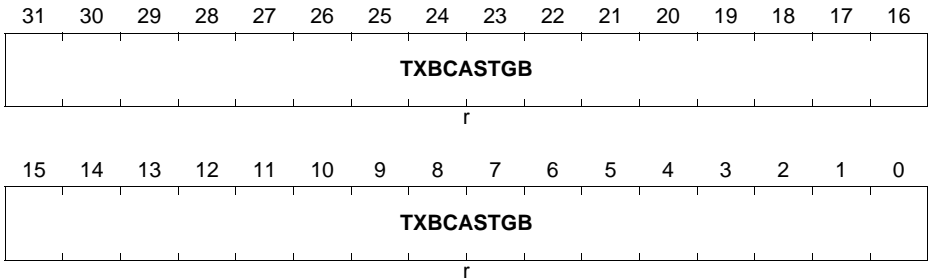
Field	Bits	Type	Description
TXMCAST GB	[31:0]	r	This field indicates the number of transmitted good and bad multicast frames.

TX_BROADCAST_FRAMES_GOOD_BAD

This register maintains the number of transmitted good and bad broadcast frames.

ETH0_TX_BROADCAST_FRAMES_GOOD_BAD

Transmit Frame Count for Good and Bad Broadcast Frames(144_H) **Reset Value:**
0000 0000_H



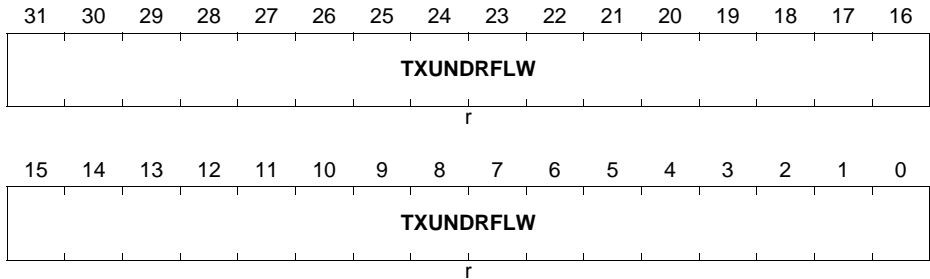
Field	Bits	Type	Description
TXBCAST GB	[31:0]	r	This field indicates the number of transmitted good and bad broadcast frames.

TX_UNDERFLOW_ERROR_FRAMES

This register maintains the number of frames aborted because of frame underflow error.

ETH0_TX_UNDERFLOW_ERROR_FRAMES

Transmit Frame Count for Underflow Error Frames (148_H) Reset Value: 0000 0000_H



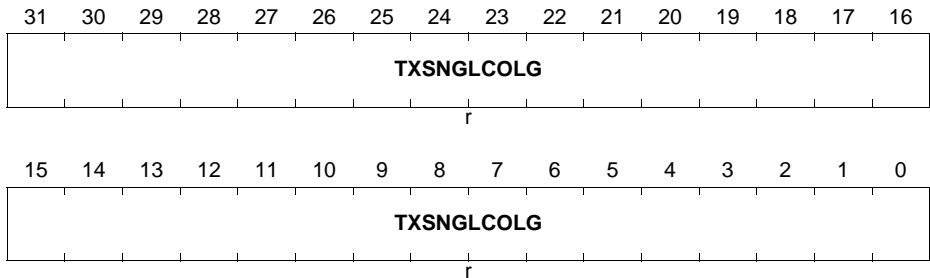
Field	Bits	Type	Description
TXUNDRFLW	[31:0]	r	This field indicates the number of frames aborted because of frame underflow error.

TX_SINGLE_COLLISION_GOOD_FRAMES

This register maintains the number of successfully transmitted frames after a single collision in the half-duplex mode.

ETH0_TX_SINGLE_COLLISION_GOOD_FRAMES

Transmit Frame Count for Frames Transmitted after Single Collision(14C_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
TXSNGLCOLG	[31:0]	r	This field indicates the number of successfully transmitted frames after a single collision in the half-duplex mode.

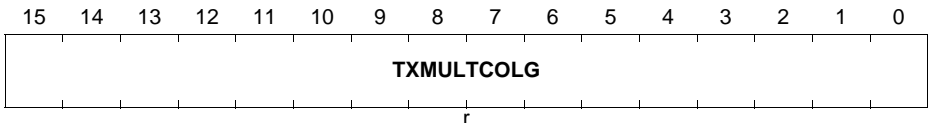
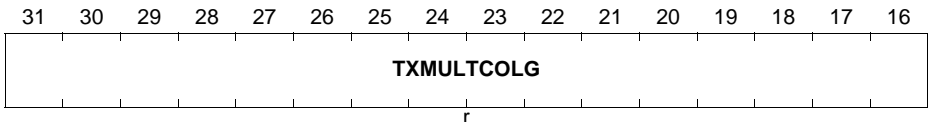
TX_MULTIPLE_COLLISION_GOOD_FRAMES

This register maintains the number of successfully transmitted frames after multiple collisions in the half-duplex mode.

ETH0_TX_MULTIPLE_COLLISION_GOOD_FRAMES

Transmit Frame Count for Frames Transmitted after Multiple Collision(150_H)

Reset Value: 0000 0000_H



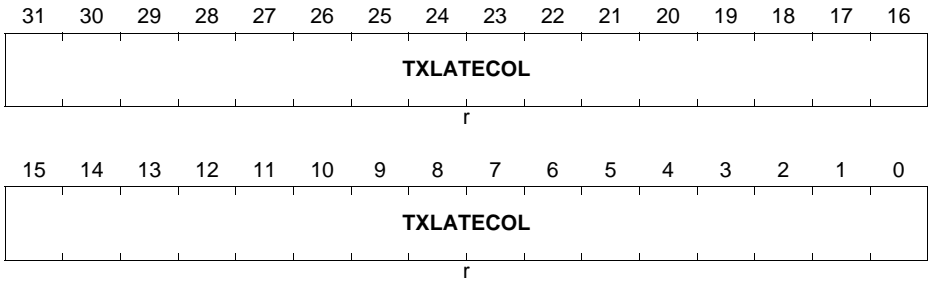
Field	Bits	Type	Description
TXMULTCOLG	[31:0]	r	This field indicates the number of successfully transmitted frames after multiple collisions in the half-duplex mode.

TX_LATE_COLLISION_FRAMES

This register maintains the number of frames aborted because of late collision error.

ETH0_TX_LATE_COLLISION_FRAMES

Transmit Frame Count for Late Collision Error Frames(158_H) Reset Value: 0000 0000_H



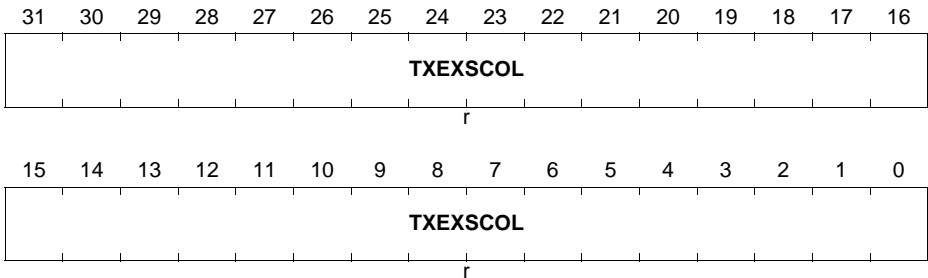
Field	Bits	Type	Description
TXLATECOL	[31:0]	r	This field indicates the number of frames aborted because of late collision error.

TX_EXCESSIVE_COLLISION_FRAMES

This register maintains the number of frames aborted because of excessive (16) collision error.

ETH0_TX_EXCESSIVE_COLLISION_FRAMES

Transmit Frame Count for Excessive Collision Error Frames(15C_H) Reset Value: 0000 0000_H



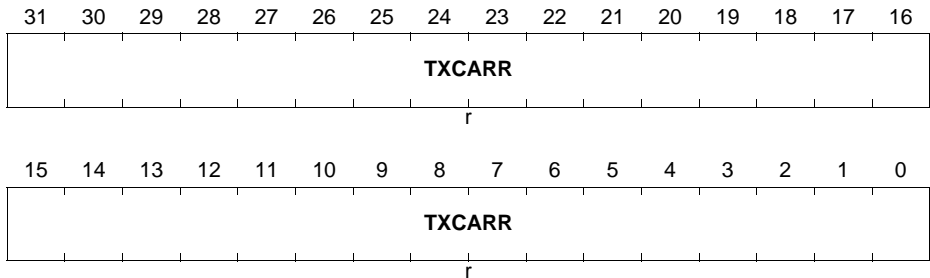
Field	Bits	Type	Description
TXEXSCOL	[31:0]	r	This field indicates the number of frames aborted because of excessive (16) collision error.

TX_CARRIER_ERROR_FRAMES

This register maintains the number of frames aborted because of carrier sense error (no carrier or loss of carrier).

ETH0_TX_CARRIER_ERROR_FRAMES

Transmit Frame Count for Carrier Sense Error Frames(160_H) Reset Value: 0000 0000_H



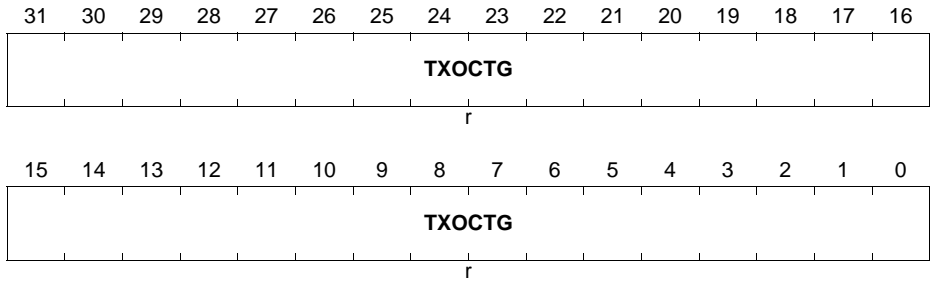
Field	Bits	Type	Description
TXCARR	[31:0]	r	This field indicates the number of frames aborted because of carrier sense error (no carrier or loss of carrier).

TX_OCTET_COUNT_GOOD

This register maintains the number of bytes transmitted, exclusive of preamble, in good frames.

ETH0_TX_OCTET_COUNT_GOOD

Tx Octet Count Good Register (164_H) **Reset Value: 0000 0000_H**



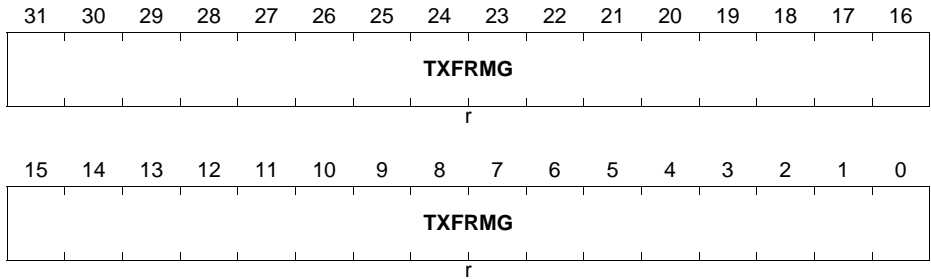
Field	Bits	Type	Description
TXOCTG	[31:0]	r	This field indicates the number of bytes transmitted, exclusive of preamble, in good frames.

TX_FRAME_COUNT_GOOD

This register maintains the number of transmitted good frames, exclusive of preamble.

ETH0_TX_FRAME_COUNT_GOOD

Tx Frame Count Good Register (168_H) **Reset Value: 0000 0000_H**



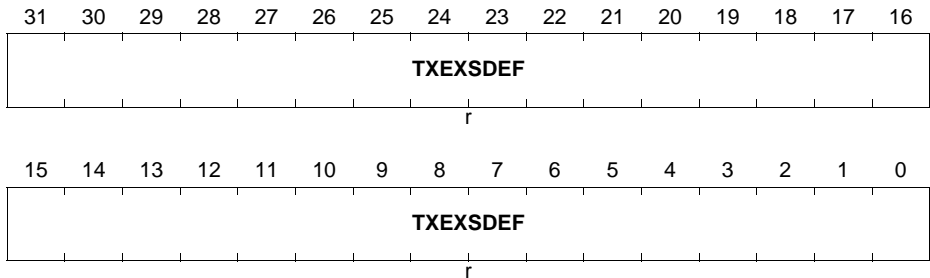
Field	Bits	Type	Description
TXFRMG	[31:0]	r	This field indicates the number of transmitted good frames, exclusive of preamble.

TX_EXCESSIVE_DEFERRAL_ERROR

This register maintains the number of frames aborted because of excessive deferral error, that is, frames deferred for more than two max-sized frame times.

ETH0_TX_EXCESSIVE_DEFERRAL_ERROR

Transmit Frame Count for Excessive Deferral Error Frames(16C_H) Reset Value: 0000 0000_H



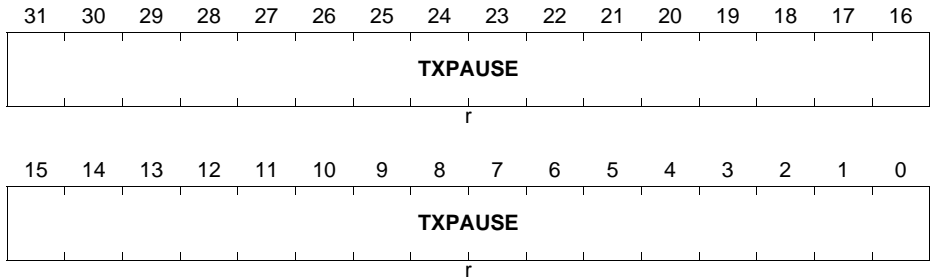
Field	Bits	Type	Description
TXEXSDEF	[31:0]	r	This field indicates the number of frames aborted because of excessive deferral error, that is, frames deferred for more than two max-sized frame times.

TX_PAUSE_FRAMES

This register maintains the number of transmitted good PAUSE frames.

ETH0_TX_PAUSE_FRAMES

Transmit Frame Count for Good PAUSE Frames(170_H) Reset Value: 0000 0000_H



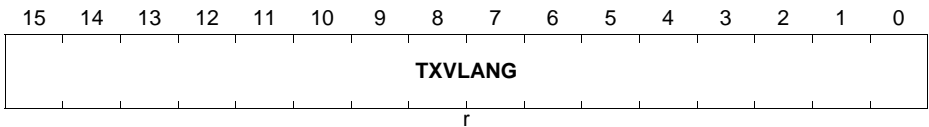
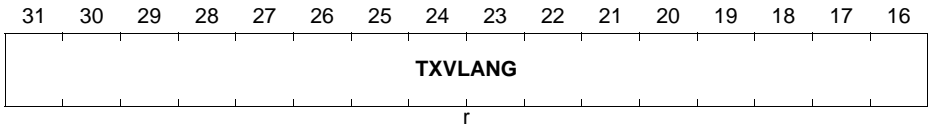
Field	Bits	Type	Description
TXPAUSE	[31:0]	r	This field indicates the number of transmitted good PAUSE frames.

TX_VLAN_FRAMES_GOOD

This register maintains the number of transmitted good VLAN frames, exclusive of retried frames.

ETH0_TX_VLAN_FRAMES_GOOD

Transmit Frame Count for Good VLAN Frames(174_H) Reset Value: 0000 0000_H



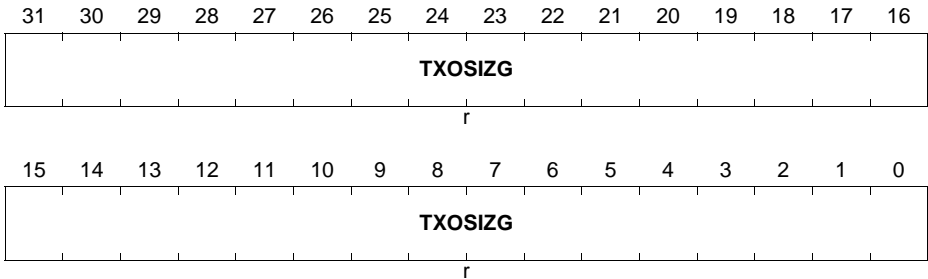
Field	Bits	Type	Description
TXVLANG	[31:0]	r	This register maintains the number of transmitted good VLAN frames, exclusive of retried frames.

TX_OSIZE_FRAMES_GOOD

This register maintains the number of transmitted good Oversize frames, exclusive of retried frames.

ETH0_TX_OSIZE_FRAMES_GOOD

Transmit Frame Count for Good Oversize Frames(178_H) Reset Value: 0000 0000_H



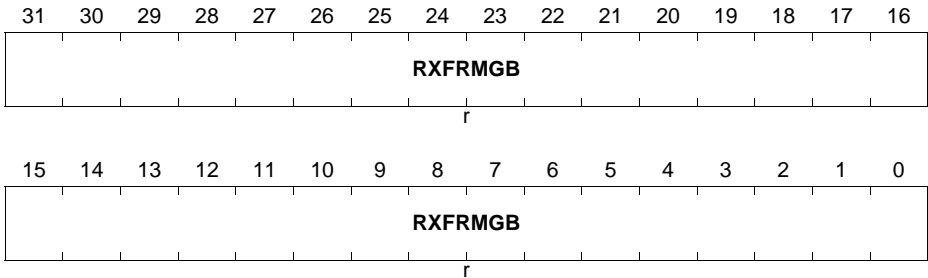
Field	Bits	Type	Description
TXOSIZG	[31:0]	r	This field indicates the number of frames transmitted without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged frames; 2000 bytes if enabled by setting MAC Configuration.TWOKP).

RX_FRAMES_COUNT_GOOD_BAD

This register maintains the number of received good and bad frames.

ETH0_RX_FRAMES_COUNT_GOOD_BAD

Receive Frame Count for Good and Bad Frames(180_H) Reset Value: 0000 0000_H



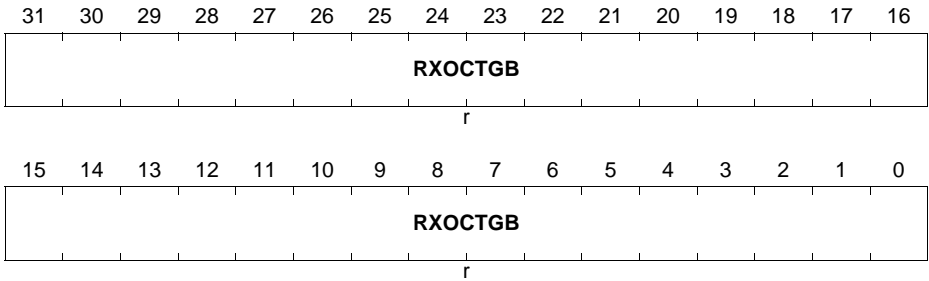
Field	Bits	Type	Description
RXFRMGB	[31:0]	r	This field indicates the number of received good and bad frames.

RX_OCTET_COUNT_GOOD_BAD

This register maintains the number of bytes received, exclusive of preamble, in good and bad frames.

ETH0_RX_OCTET_COUNT_GOOD_BAD

Receive Octet Count for Good and Bad Frames(184_H) Reset Value: 0000 0000_H



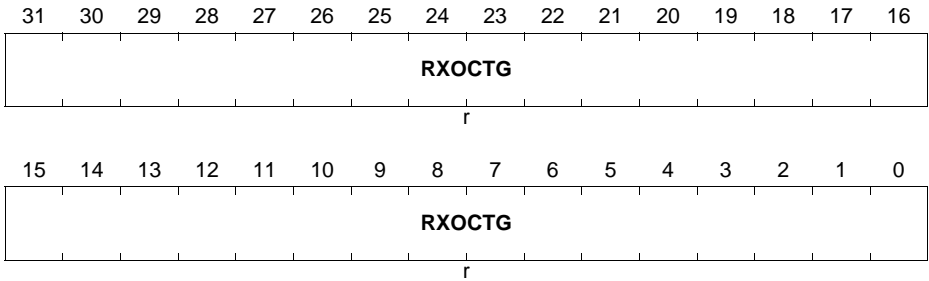
Field	Bits	Type	Description
RXOCTGB	[31:0]	r	This field indicates the number of bytes received, exclusive of preamble, in good and bad frames.

RX_OCTET_COUNT_GOOD

This register maintains the number of bytes received, exclusive of preamble, only in good frames.

ETH0_RX_OCTET_COUNT_GOOD

Rx Octet Count Good Register (188_H) **Reset Value: 0000 0000_H**



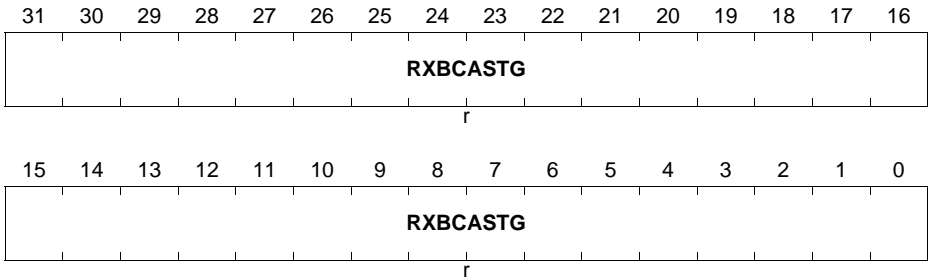
Field	Bits	Type	Description
RXOCTG	[31:0]	r	This field indicates the number of bytes received, exclusive of preamble, only in good frames.

RX_BROADCAST_FRAMES_GOOD

This register maintains the number of received good broadcast frames.

ETH0_RX_BROADCAST_FRAMES_GOOD

Receive Frame Count for Good Broadcast Frames(18C_H) Reset Value: 0000 0000_H



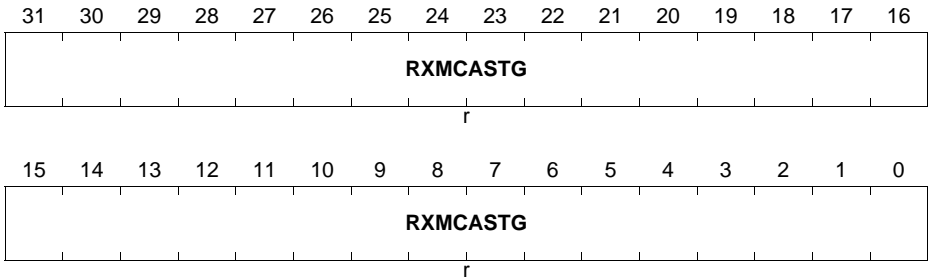
Field	Bits	Type	Description
RXBCASTG	[31:0]	r	This field indicates the number of received good broadcast frames.

RX_MULTICAST_FRAMES_GOOD

This register maintains the number of received good multicast frames.

ETH0_RX_MULTICAST_FRAMES_GOOD

Receive Frame Count for Good Multicast Frames(190_H) Reset Value: 0000 0000_H



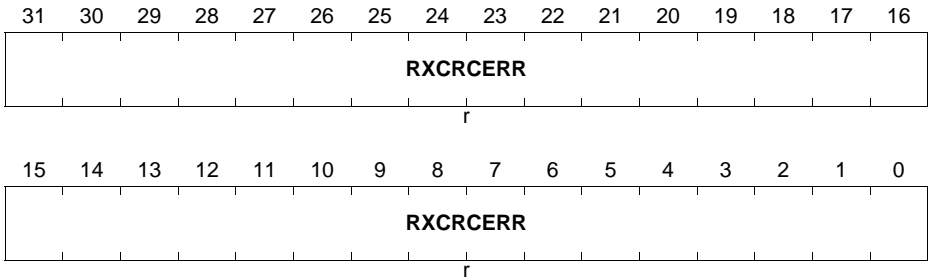
Field	Bits	Type	Description
RXCASTG	[31:0]	r	This field indicates the number of received good multicast frames.

RX_CRC_ERROR_FRAMES

This register maintains the number of frames received with CRC error.

ETH0_RX_CRC_ERROR_FRAMES

Receive Frame Count for CRC Error Frames(194_H) **Reset Value: 0000 0000_H**



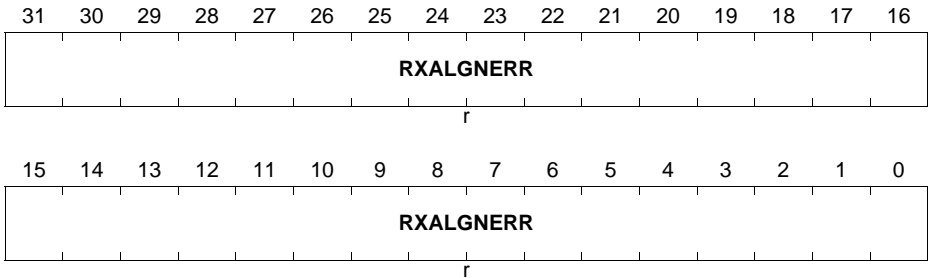
Field	Bits	Type	Description
RXCRCER R	[31:0]	r	This field indicates the number of frames received with CRC error.

RX_ALIGNMENT_ERROR_FRAMES

This register maintains the number of frames received with alignment (dribble) error.

ETH0_RX_ALIGNMENT_ERROR_FRAMES

Receive Frame Count for Alignment Error Frames(198_H) Reset Value: 0000 0000_H



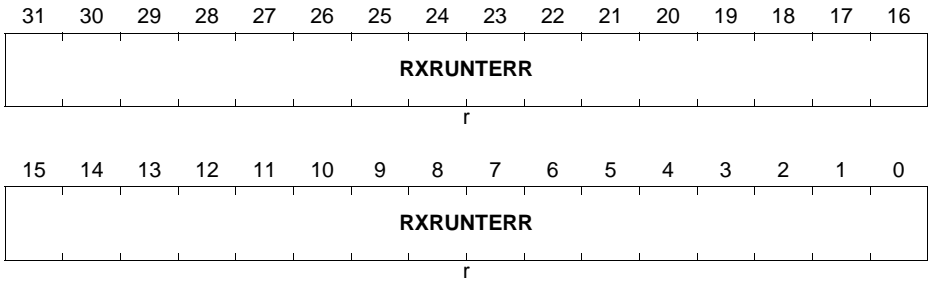
Field	Bits	Type	Description
RXALGNERR	[31:0]	r	This field indicates the number of frames received with alignment (dribble) error.

RX_RUNT_ERROR_FRAMES

This register maintains the number of frames received with runt error(<64 bytes and CRC error).

ETH0_RX_RUNT_ERROR_FRAMES

Receive Frame Count for Runt Error Frames(19C_H) Reset Value: 0000 0000_H



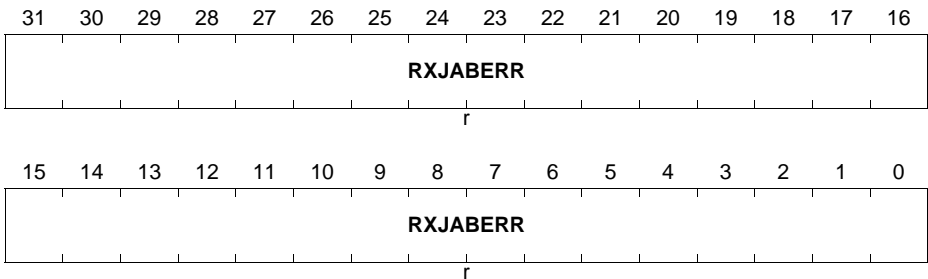
Field	Bits	Type	Description
RXRUNTE RR	[31:0]	r	This field indicates the number of frames received with runt error(<64 bytes and CRC error).

RX_JABBER_ERROR_FRAMES

This register maintains the number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames.

ETH0_RX_JABBER_ERROR_FRAMES

Receive Frame Count for Jabber Error Frames(1A0_H) Reset Value: 0000 0000_H



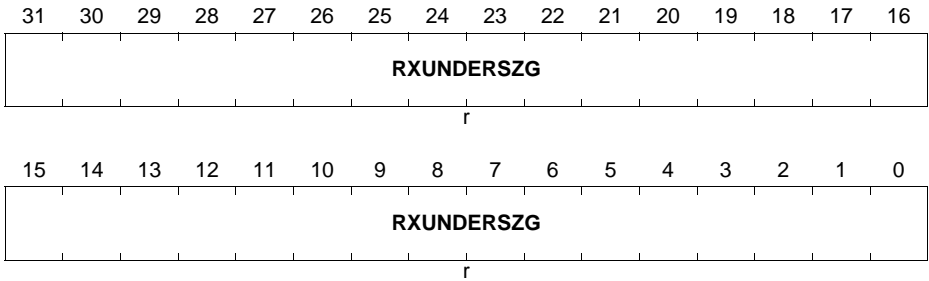
Field	Bits	Type	Description
RXJABERR	[31:0]	r	This field indicates the number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames.

RX_UNDERSIZE_FRAMES_GOOD

This register maintains the number of frames received with length less than 64 bytes and without errors.

ETH0_RX_UNDERSIZE_FRAMES_GOOD

Receive Frame Count for Undersize Frames(1A4_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RXUNDER SZG	[31:0]	r	This field indicates the number of frames received with length less than 64 bytes and without errors.

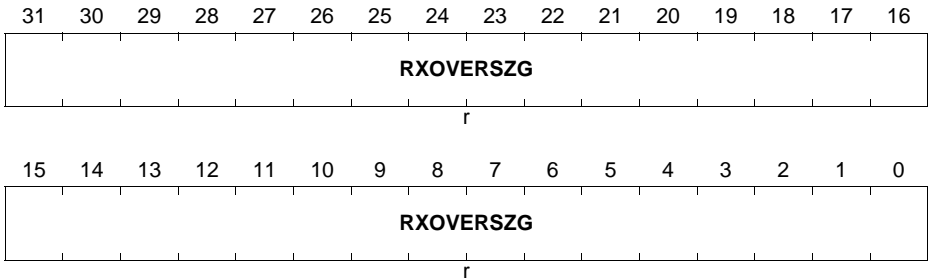
RX_OVERSIZE_FRAMES_GOOD

This register maintains the number of frames received with length greater than the maxsize (1,518 or 1,522 for VLAN tagged frames) and without errors.

ETH0_RX_OVERSIZE_FRAMES_GOOD

Rx Oversize Frames Good Register (1A8_H)

Reset Value: 0000 0000_H



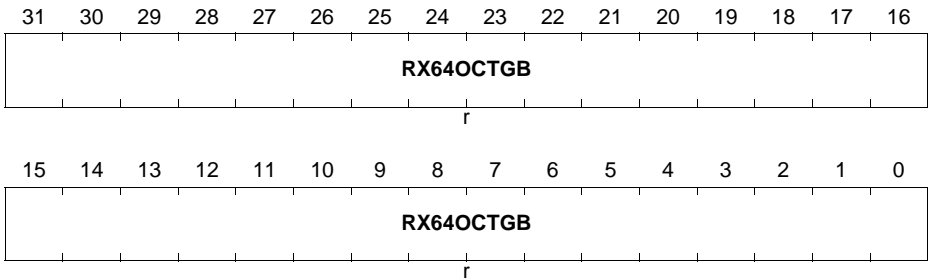
Field	Bits	Type	Description
RXOVERSZG	[31:0]	r	This field indicates the number of frames received without errors, with length greater than the maxsize (1,518 or 1,522 for VLAN tagged frames; 2,000 bytes if enabled by setting MAC Configuration.TWOKPE).

RX_64OCTETS_FRAMES_GOOD_BAD

This register maintains the number of received good and bad frames with length 64 bytes, exclusive of preamble.

ETH0_RX_64OCTETS_FRAMES_GOOD_BAD

Receive Frame Count for Good and Bad 64 Byte Frames(1AC_H) Reset Value: 0000 0000_H



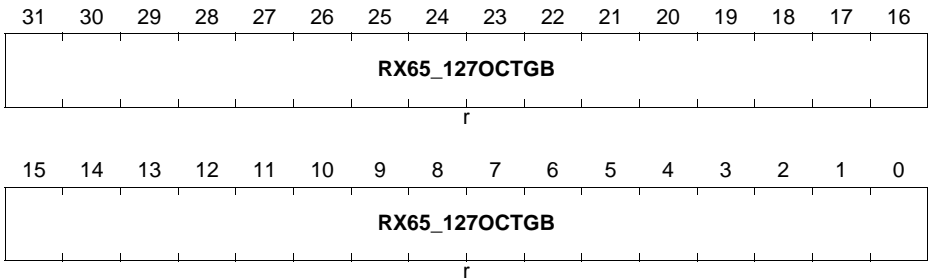
Field	Bits	Type	Description
RX64OCT GB	[31:0]	r	This field indicates the number of received good and bad frames with length 64 bytes, exclusive of preamble.

RX_65TO127OCTETS_FRAMES_GOOD_BAD

This register maintains the number of received good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble.

ETH0_RX_65TO127OCTETS_FRAMES_GOOD_BAD

Receive Frame Count for Good and Bad 65 to 127 Bytes Frames(1B0_H) **Reset Value: 0000 0000_H**



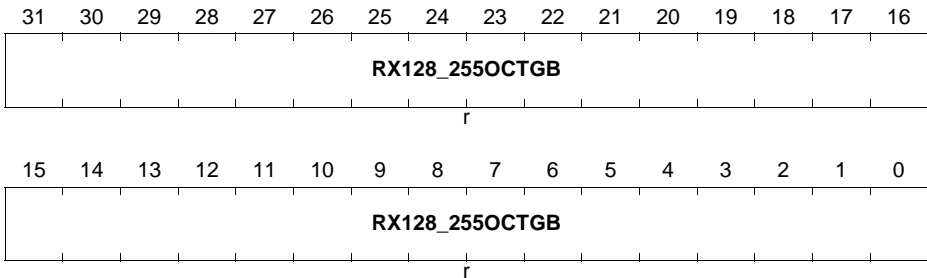
Field	Bits	Type	Description
RX65_127 OCTGB	[31:0]	r	This field indicates the number of received good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble.

RX_128TO255OCTETS_FRAMES_GOOD_BAD

This register maintains the number of received good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble.

ETH0_RX_128TO255OCTETS_FRAMES_GOOD_BAD

Receive Frame Count for Good and Bad 128 to 255 Bytes Frames(1B4_μ) **Reset Value: 0000 0000_H**



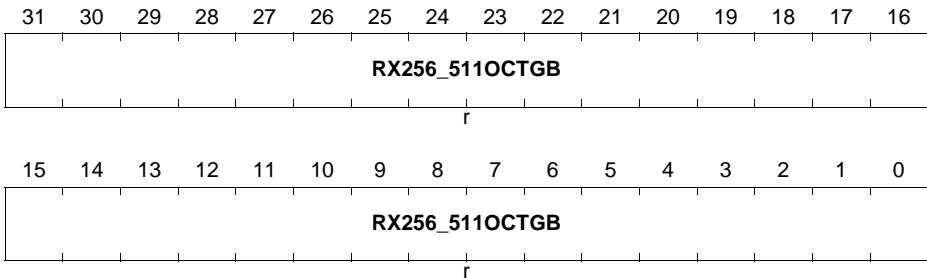
Field	Bits	Type	Description
RX128_255OCTGB	[31:0]	r	This field indicates the number of received good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble.

RX_256TO511OCTETS_FRAMES_GOOD_BAD

This register maintains the number of received good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble.

ETH0_RX_256TO511OCTETS_FRAMES_GOOD_BAD

Receive Frame Count for Good and Bad 256 to 511 Bytes Frames(1B8_H) Reset Value: 0000 0000_H



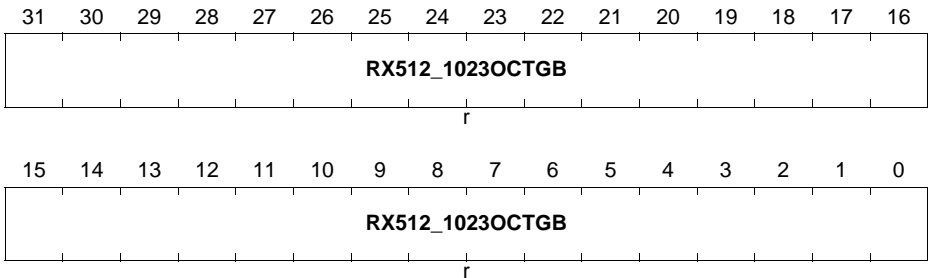
Field	Bits	Type	Description
RX256_511OCTGB	[31:0]	r	This field indicates the number of received good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble.

RX_512TO1023OCTETS_FRAMES_GOOD_BAD

This register maintains the number of received good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble.

ETH0_RX_512TO1023OCTETS_FRAMES_GOOD_BAD

Receive Frame Count for Good and Bad 512 to 1,023 Bytes Frames(1BC_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
RX512_1023OCTGB	[31:0]	r	This field indicates the number of received good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble.

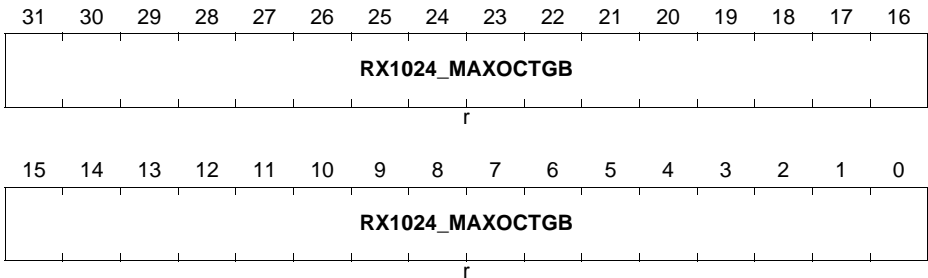
RX_1024TOMAXOCTETS_FRAMES_GOOD_BAD

This register maintains the number of received good and bad frames with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble.

ETH0_RX_1024TOMAXOCTETS_FRAMES_GOOD_BAD

Receive Frame Count for Good and Bad 1,024 to Maxsize Bytes Frames(1C0_H)

Reset Value: 0000 0000_H



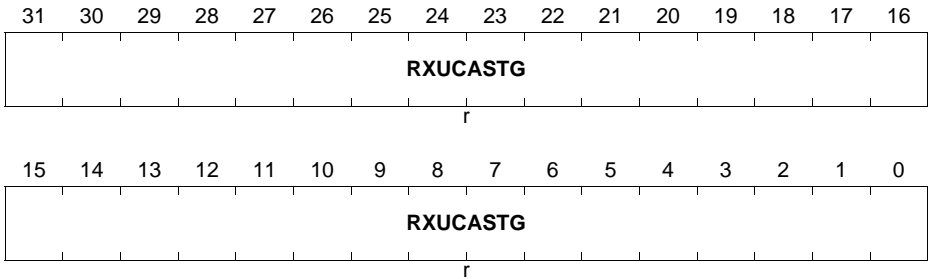
Field	Bits	Type	Description
RX1024_M AXOCTGB	[31:0]	r	This field indicates the number of received good and bad frames with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.

RX_UNICAST_FRAMES_GOOD

This register maintains the number of received good unicast frames.

ETH0_RX_UNICAST_FRAMES_GOOD

Receive Frame Count for Good Unicast Frames(1C4_H) **Reset Value: 0000 0000_H**



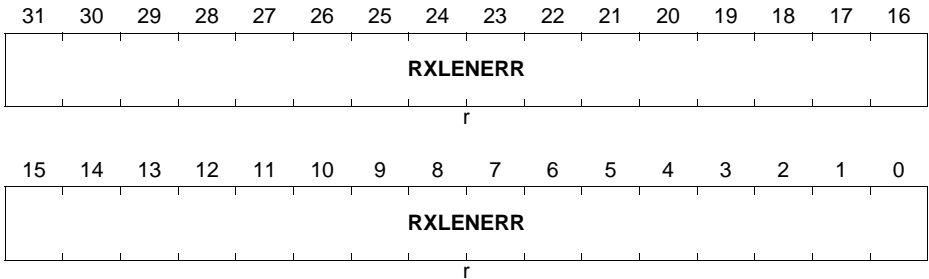
Field	Bits	Type	Description
RXUCASTG	[31:0]	r	This field indicates the number of received good unicast frames.

RX_LENGTH_ERROR_FRAMES

This register maintains the number of frames received with length error (Length type field not equal to frame size) for all frames with valid length field.

ETH0_RX_LENGTH_ERROR_FRAMES

Receive Frame Count for Length Error Frames(1C8_H) Reset Value: 0000 0000_H



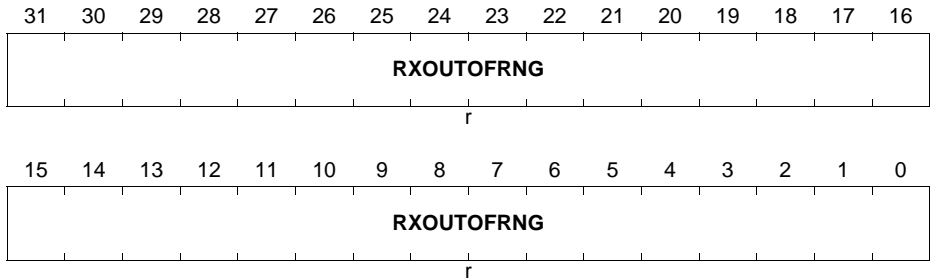
Field	Bits	Type	Description
RXLENER R	[31:0]	r	This field indicates the number of frames received with length error (Length type field not equal to frame size) for all frames with valid length field.

RX_OUT_OF_RANGE_TYPE_FRAMES

This register maintains the number of received frames with length field not equal to the valid frame size (greater than 1,500 but less than 1,536).

ETH0_RX_OUT_OF_RANGE_TYPE_FRAMES

Receive Frame Count for Out of Range Frames(1CC_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RXOUTOFRNG	[31:0]	r	This field indicates the number of received frames with length field not equal to the valid frame size (greater than 1,500 but less than 1,536).

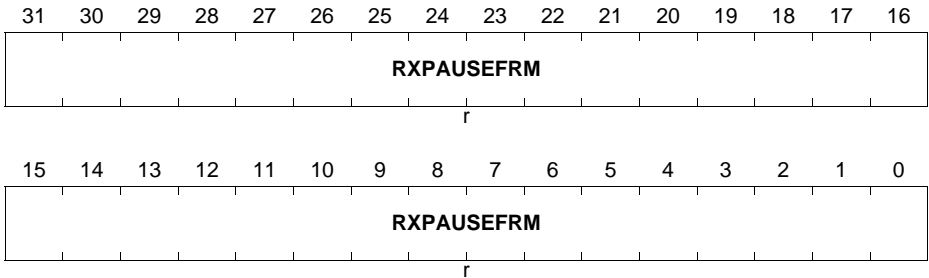
RX_PAUSE_FRAMES

This register maintains the number of received good and valid PAUSE frames.

ETH0_RX_PAUSE_FRAMES

Receive Frame Count for PAUSE Frames(1D0_H)

Reset Value: 0000 0000_H



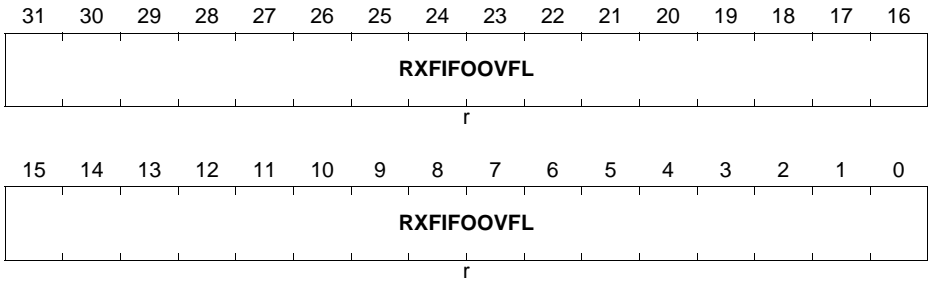
Field	Bits	Type	Description
RXPAUSE FRM	[31:0]	r	This field indicates the number of received good and valid PAUSE frames.

RX_FIFO_OVERFLOW_FRAMES

This register maintains the number of received frames missed because of FIFO overflow.

ETH0_RX_FIFO_OVERFLOW_FRAMES

Receive Frame Count for FIFO Overflow Frames(1D4_H) Reset Value: 0000 0000_H



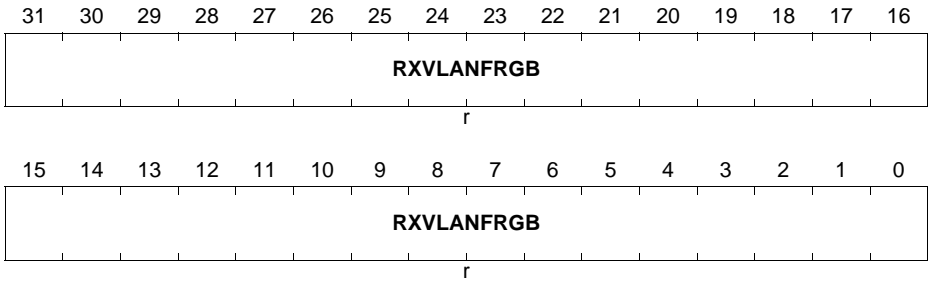
Field	Bits	Type	Description
RXFIFO VFL	[31:0]	r	This field indicates the number of received frames missed because of FIFO overflow.

RX_VLAN_FRAMES_GOOD_BAD

This register maintains the number of received good and bad VLAN frames.

ETH0_RX_VLAN_FRAMES_GOOD_BAD

Receive Frame Count for Good and Bad VLAN Frames(1D8_H) **Reset Value: 0000 0000_H**



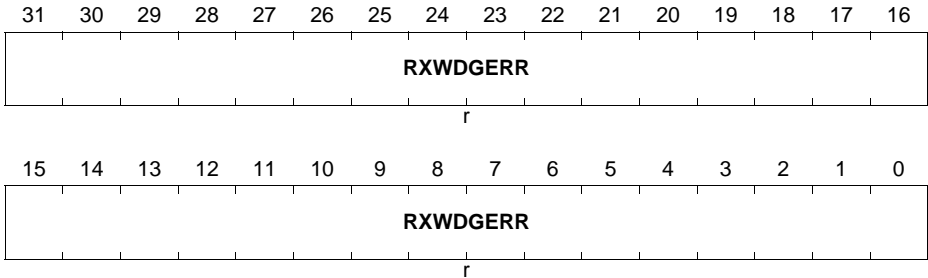
Field	Bits	Type	Description
RXVLANFRGB	[31:0]	r	This field indicates the number of received good and bad VLAN frames.

RX_WATCHDOG_ERROR_FRAMES

This register maintains the number of frames received with error because of the watchdog timeout error (frames with more than 2,048 bytes data load).

ETH0_RX_WATCHDOG_ERROR_FRAMES

Receive Frame Count for Watchdog Error Frames(1DC_H) Reset Value: 0000 0000_H



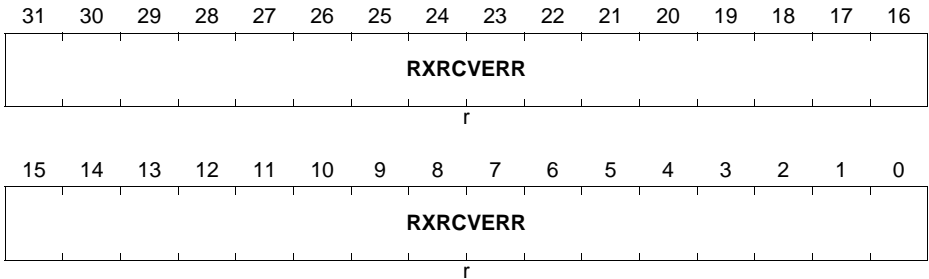
Field	Bits	Type	Description
RXWDGERR	[31:0]	r	This field indicates the number of frames received with error because of the watchdog timeout error (frames with more than 2,048 bytes data load).

RX_RECEIVE_ERROR_FRAMES

This register maintains the number of frames received with error because of the MII RXER error.

ETH0_RX_RECEIVE_ERROR_FRAMES

Receive Frame Count for Receive Error Frames(1E0_H) Reset Value: 0000 0000_H



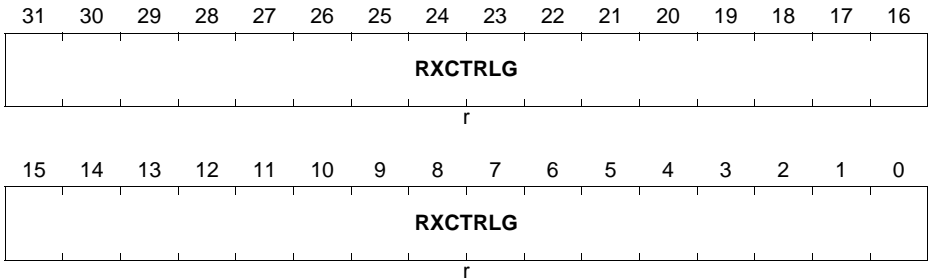
Field	Bits	Type	Description
RXRCVERR	[31:0]	r	This field indicates the number of frames received with error because of the watchdog timeout error (frames with more than 2,048 bytes data load).

RX_CONTROL_FRAMES_GOOD

This register maintains the number of good control frames received.

ETH0_RX_CONTROL_FRAMES_GOOD

Receive Frame Count for Good Control Frames (1E4_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXCTRLG	[31:0]	r	This field indicates the number of frames received with error because of the watchdog timeout error (frames with more than 2,048 bytes data load).

MMC_IPC_RECEIVE_INTERRUPT_MASK

This register maintains the mask for the interrupt generated from the receive IPC statistic counters. This register is 32-bits wide.

ETH0_MMC_IPC_RECEIVE_INTERRUPT_MASK

MMC Receive Checksum Offload Interrupt Mask Register(200_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved_31_30	RXIC MPE ROI M	RXIC MPG OIM	RXT CPE ROI M	RXT CPG OIM	RXU DPE ROI M	RXU DPG OIM	RXIP V6N OPA YOIM	RXIP V6H EROI M	RXIP V6G OIM	RXIP V4U DSB LOIM	RXIP V4F RAG OIM	RXIP V4N OPA YOIM	RXIP V4H EROI M	RXIP V4G OIM	
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved_15_14	RXIC MPE RFIM	RXIC MPG FIM	RXT CPE RFIM	RXT CPG FIM	RXU DPE RFIM	RXU DPG FIM	RXIP V6N OPA YFIM	RXIP V6H ERFIM	RXIP V6G FIM	RXIP V4U DSB LFIM	RXIP V4F RAG FIM	RXIP V4N OPA YFIM	RXIP V4H ERFIM	RXIP V4G FIM	
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
RXIPV4GFI M	0	rw	MMC Receive IPV4 Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV4_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.
RXIPV4HE RFIM	1	rw	MMC Receive IPV4 Header Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV4_HEADER_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXIPV4NO PAYFIM	2	rw	MMC Receive IPV4 No Payload Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV4_NO_PAYLOAD_FRAMES counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXIPV4FR AGFIM	3	rw	MMC Receive IPV4 Fragmented Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV4_FRAGMENTED_FRAMES counter reaches half of the maximum value or the maximum value.
RXIPV4UD SBLFIM	4	rw	MMC Receive IPV4 UDP Checksum Disabled Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV4_UDP_CHECKSUM_DISABLED_FRAMES counter reaches half of the maximum value or the maximum value.
RXIPV6GF IM	5	rw	MMC Receive IPV6 Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV6_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.
RXIPV6HE RFIM	6	rw	MMC Receive IPV6 Header Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV6_HEADER_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXIPV6NO PAYFIM	7	rw	MMC Receive IPV6 No Payload Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV6_NO_PAYLOAD_FRAMES counter reaches half of the maximum value or the maximum value.
RXUDPGF IM	8	rw	MMC Receive UDP Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXUDP_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.
RXUDPER FIM	9	rw	MMC Receive UDP Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXUDP_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXTCPGFI M	10	rw	MMC Receive TCP Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXTCP_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.
RXTCPER FIM	11	rw	MMC Receive TCP Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXTCP_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXICMPG FIM	12	rw	MMC Receive ICMP Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXICMP_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.
RXICMPE RFIM	13	rw	MMC Receive ICMP Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the RXICMP_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
Reserved_ 15_14	[15:14]	r	Reserved
RXIPV4G OIM	16	rw	MMC Receive IPv4 Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV4_GOOD_OCTETS counter reaches half of the maximum value or the maximum value.
RXIPV4HE ROIM	17	rw	MMC Receive IPv4 Header Error Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV4_HEADER_ERROR_OCTETS counter reaches half of the maximum value or the maximum value.
RXIPV4NO PAYOIM	18	rw	MMC Receive IPv4 No Payload Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV4_NO_PAYLOAD_OCTETS counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXIPV4FR AGOIM	19	rw	MMC Receive IPV4 Fragmented Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV4_FRAGMENTED_OCTETS counter reaches half of the maximum value or the maximum value.
RXIPV4UD SBLOIM	20	rw	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS counter reaches half of the maximum value or the maximum value.
RXIPV6G OIM	21	rw	MMC Receive IPV6 Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV6_GOOD_OCTETS counter reaches half of the maximum value or the maximum value.
RXIPV6HE ROIM	22	rw	MMC Receive IPV6 Header Error Octet Counter Interrupt Mask Setting this bit masks interrupt when the RXIPV6_HEADER_ERROR_OCTETS counter reaches half of the maximum value or the maximum value.
RXIPV6NO PAYOIM	23	rw	MMC Receive IPV6 No Payload Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXIPV6_NO_PAYLOAD_OCTETS counter reaches half of the maximum value or the maximum value.
RXUDPGO IM	24	rw	MMC Receive UDP Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXUDP_GOOD_OCTETS counter reaches half of the maximum value or the maximum value.
RXUDPER OIM	25	rw	MMC Receive UDP Error Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXUDP_ERROR_OCTETS counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXTCPGO IM	26	rw	MMC Receive TCP Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXTCP_GOOD_OCTETS counter reaches half of the maximum value or the maximum value.
RXTCPER OIM	27	rw	MMC Receive TCP Error Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXTCP_ERROR_OCTETS counter reaches half of the maximum value or the maximum value.
RXICMPG OIM	28	rw	MMC Receive ICMP Good Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXICMP_GOOD_OCTETS counter reaches half of the maximum value or the maximum value.
RXICMPE ROIM	29	rw	MMC Receive ICMP Error Octet Counter Interrupt Mask Setting this bit masks the interrupt when the RXICMP_ERROR_OCTETS counter reaches half of the maximum value or the maximum value.
Reserved_ 31_30	[31:30]	r	Reserved

MMC_IPC_RECEIVE_INTERRUPT

This register maintains the interrupt that the receive IPC statistic counters generate.

ETH0_MMC_IPC_RECEIVE_INTERRUPT

MMC Receive Checksum Offload Interrupt Register(208_H)Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved_31_30	RXIC MPE ROIS	RXIC MPG OIS	RXT CPE ROIS	RXT CPG OIS	RXU DPE ROIS	RXU DPG OIS	RXIP V6N OPA YOIS	RXIP V6H EROI S	RXIP V6G OIS	RXIP V4U DSB LOIS	RXIP V4F RAG OIS	RXIP V4N OPA YOIS	RXIP V4H EROI S	RXIP V4G OIS	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved_15_14	RXIC MPE RFIS	RXIC MPG FIS	RXT CPE RFIS	RXT CPG FIS	RXU DPE RFIS	RXU DPG FIS	RXIP V6N OPA YFIS	RXIP V6H ERFI S	RXIP V6G FIS	RXIP V4U DSB LFIS	RXIP V4F RAG FIS	RXIP V4N OPA YFIS	RXIP V4H ERFI S	RXIP V4G FIS	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
RXIPV4GFIS	0	r	MMC Receive IPv4 Good Frame Counter Interrupt Status This bit is set when the RXIPV4_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.
RXIPV4HERFIS	1	r	MMC Receive IPv4 Header Error Frame Counter Interrupt Status This bit is set when the RXIPV4_HEADER_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXIPV4NO PAYFIS	2	r	MMC Receive IPv4 No Payload Frame Counter Interrupt Status This bit is set when the RXIPV4_NO_PAYLOAD_FRAMES counter reaches half of the maximum value or the maximum value.
RXIPV4FRAGFIS	3	r	MMC Receive IPv4 Fragmented Frame Counter Interrupt Status This bit is set when the RXIPV4_FRAGMENTED_FRAMES counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXIPV4UD SBLFIS	4	r	MMC Receive IPV4 UDP Checksum Disabled Frame Counter Interrupt Status This bit is set when the RXIPV4_UDP_CHECKSUM_DISABLED_FRAMES counter reaches half of the maximum value or the maximum value.
RXIPV6GF IS	5	r	MMC Receive IPV6 Good Frame Counter Interrupt Status This bit is set when the RXIPV6_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.
RXIPV6HE RFIS	6	r	MMC Receive IPV6 Header Error Frame Counter Interrupt Status This bit is set when the RXIPV6_HEADER_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXIPV6NO PAYFIS	7	r	MMC Receive IPV6 No Payload Frame Counter Interrupt Status This bit is set when the RXIPV6_NO_PAYLOAD_FRAMES counter reaches half of the maximum value or the maximum value.
RXUDPGF IS	8	r	MMC Receive UDP Good Frame Counter Interrupt Status This bit is set when the RXUDP_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.
RXUDPER FIS	9	r	MMC Receive UDP Error Frame Counter Interrupt Status This bit is set when the RXUDP_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXTCPGFI S	10	r	MMC Receive TCP Good Frame Counter Interrupt Status This bit is set when the RXTCP_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXTCPER FIS	11	r	MMC Receive TCP Error Frame Counter Interrupt Status This bit is set when the RXTCP_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
RXICMPG FIS	12	r	MMC Receive ICMP Good Frame Counter Interrupt Status This bit is set when the RXICMP_GOOD_FRAMES counter reaches half of the maximum value or the maximum value.
RXICMPE RFIS	13	r	MMC Receive ICMP Error Frame Counter Interrupt Status This bit is set when the RXICMP_ERROR_FRAMES counter reaches half of the maximum value or the maximum value.
Reserved_15_14	[15:14]	r	Reserved
RXIPV4G OIS	16	r	MMC Receive IPv4 Good Octet Counter Interrupt Status This bit is set when the RXIPV4_GOOD_OCTETS counter reaches half of the maximum value or the maximum value.
RXIPV4HE ROIS	17	r	MMC Receive IPv4 Header Error Octet Counter Interrupt Status This bit is set when the RXIPV4_HEADER_ERROR_OCTETS counter reaches half of the maximum value or the maximum value.
RXIPV4NO PAYOIS	18	r	MMC Receive IPv4 No Payload Octet Counter Interrupt Status This bit is set when the RXIPV4_NO_PAYLOAD_OCTETS counter reaches half of the maximum value or the maximum value.
RXIPV4FR AGOIS	19	r	MMC Receive IPv4 Fragmented Octet Counter Interrupt Status This bit is set when the RXIPV4_FRAGMENTED_OCTETS counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXIPV4UD SBLOIS	20	r	MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Status This bit is set when the RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS counter reaches half of the maximum value or the maximum value.
RXIPV6G OIS	21	r	MMC Receive IPV6 Good Octet Counter Interrupt Status This bit is set when the RXIPV6_GOOD_OCTETS counter reaches half of the maximum value or the maximum value.
RXIPV6HE ROIS	22	r	MMC Receive IPV6 Header Error Octet Counter Interrupt Status This bit is set when the RXIPV6_HEADER_ERROR_OCTETS counter reaches half of the maximum value or the maximum value.
RXIPV6NO PAYOIS	23	r	MMC Receive IPV6 No Payload Octet Counter Interrupt Status This bit is set when the RXIPV6_NO_PAYLOAD_OCTETS counter reaches half of the maximum value or the maximum value.
RXUDPGO IS	24	r	MMC Receive UDP Good Octet Counter Interrupt Status This bit is set when the RXUDP_GOOD_OCTETS counter reaches half of the maximum value or the maximum value.
RXUDPER OIS	25	r	MMC Receive UDP Error Octet Counter Interrupt Status This bit is set when the RXUDP_ERROR_OCTETS counter reaches half the maximum value or the maximum value.
RXTCPGO IS	26	r	MMC Receive TCP Good Octet Counter Interrupt Status This bit is set when the RXTCP_GOOD_OCTETS counter reaches half the maximum value or the maximum value.

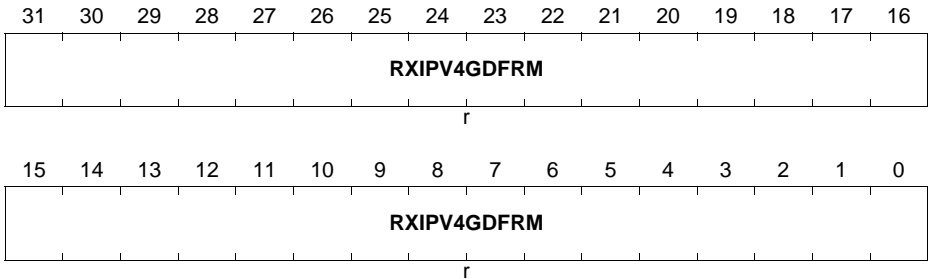
Field	Bits	Type	Description
RXTCPEROIS	27	r	MMC Receive TCP Error Octet Counter Interrupt Status This bit is set when the RXTCP_ERROR_OCTETS counter reaches half of the maximum value or the maximum value.
RXICMPOIS	28	r	MMC Receive ICMP Good Octet Counter Interrupt Status This bit is set when the RXICMP_GOOD_OCTETS counter reaches half of the maximum value or the maximum value.
RXICMPEOIS	29	r	MMC Receive ICMP Error Octet Counter Interrupt Status This bit is set when the RXICMP_ERROR_OCTETS counter reaches half of the maximum value or the maximum value.
Reserved_31_30	[31:30]	r	Reserved

RXIPV4_GOOD_FRAMES

This register maintains the number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload.

ETH0_RXIPV4_GOOD_FRAMES

RxIPv4 Good Frames Register (210_H) **Reset Value: 0000 0000_H**



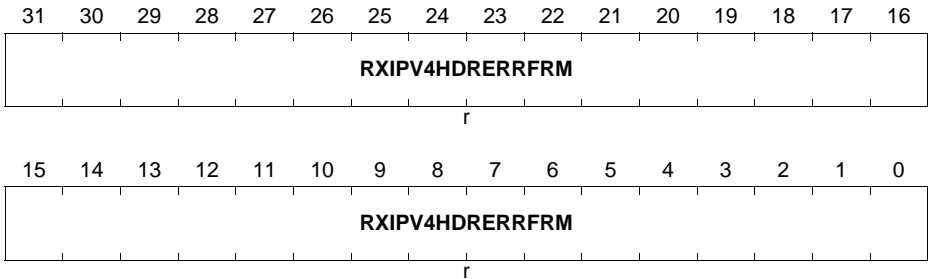
Field	Bits	Type	Description
RXIPV4GDFRM	[31:0]	r	This field indicates the number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload.

RXIPV4_HEADER_ERROR_FRAMES

This register maintains the number of IPv4 datagrams received with header errors (checksum, length, or version mismatch).

ETH0_RXIPV4_HEADER_ERROR_FRAMES

Receive IPV4 Header Error Frame Counter Register(214_H)Reset Value: 0000 0000_H



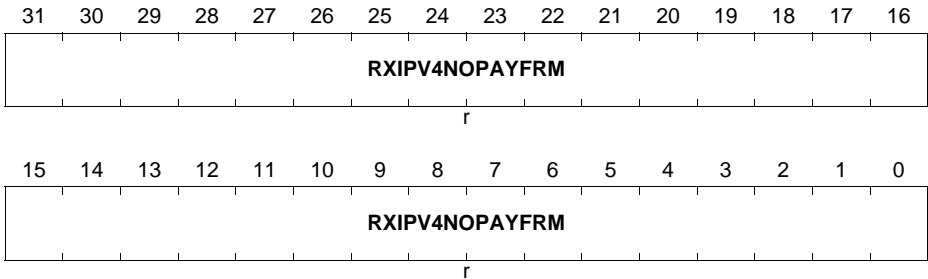
Field	Bits	Type	Description
RXIPV4HDRERRFRM	[31:0]	r	This field indicates the number of IPv4 datagrams received with header errors (checksum, length, or version mismatch).

RXIPV4_NO_PAYLOAD_FRAMES

This register maintains the number of received IPv4 datagram frames without a TCP, UDP, or ICMP payload processed by the Checksum engine.

ETH0_RXIPV4_NO_PAYLOAD_FRAMES

Receive IPV4 No Payload Frame Counter Register(218_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXIPV4NO PAYFRM	[31:0]	r	This field indicates the number of IPv4 datagram frames received that did not have a TCP, UDP, or ICMP payload processed by the Checksum engine.

RXIPV4_FRAGMENTED_FRAMES

This register maintains the number of good IPv4 datagrams received with fragmentation.

ETH0_RXIPV4_FRAGMENTED_FRAMES

Receive IPV4 Fragmented Frame Counter Register(21C_H) Reset Value: 0000 0000_H



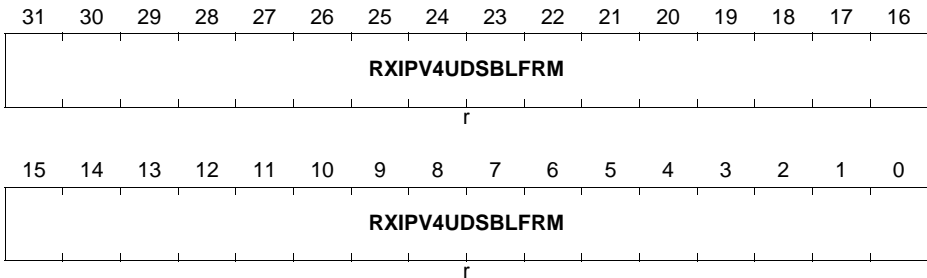
Field	Bits	Type	Description
RXIPV4FRAGFRM	[31:0]	r	This field indicates the number of good IPv4 datagrams received with fragmentation.

RXIPV4_UDP_CHECKSUM_DISABLED_FRAMES

This register maintains the number of received good IPv4 datagrams which have the UDP payload with checksum disabled.

ETH0_RXIPV4_UDP_CHECKSUM_DISABLED_FRAMES

Receive IPV4 UDP Checksum Disabled Frame Counter Register(220_H) **Reset Value: 0000 0000_H**



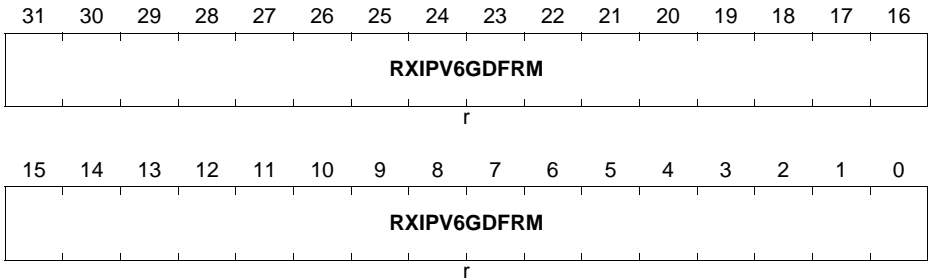
Field	Bits	Type	Description
RXIPV4UDSBLFRM	[31:0]	r	This field indicates the number of received good IPv4 datagrams which have the UDP payload with checksum disabled.

RXIPV6_GOOD_FRAMES

This register maintains the number of good IPv6 datagrams received with TCP, UDP, or ICMP payloads.

ETH0_RXIPV6_GOOD_FRAMES

RxIPv6 Good Frames Register (224_H) **Reset Value: 0000 0000_H**



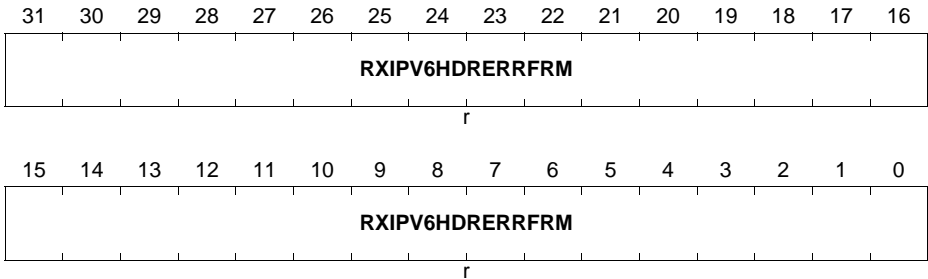
Field	Bits	Type	Description
RXIPV6GDFRM	[31:0]	r	This field indicates the number of good IPv6 datagrams received with TCP, UDP, or ICMP payloads.

RXIPV6_HEADER_ERROR_FRAMES

This register maintains the number of IPv6 datagrams received with header errors (length or version mismatch).

ETH0_RXIPV6_HEADER_ERROR_FRAMES

Receive IPV6 Header Error Frame Counter Register(228_H)Reset Value: 0000 0000_H



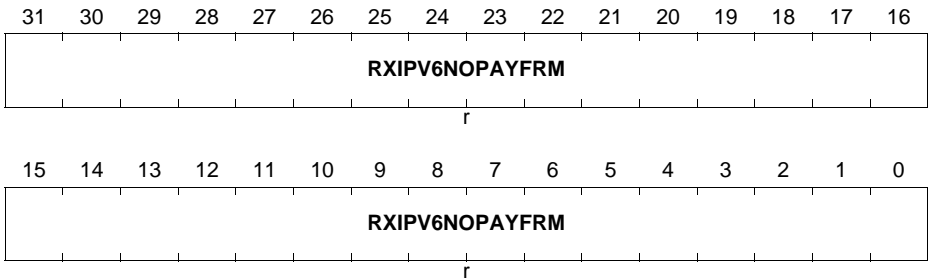
Field	Bits	Type	Description
RXIPV6HDRERRFRM	[31:0]	r	This field indicates the number of IPv6 datagrams received with header errors (length or version mismatch).

RXIPV6_NO_PAYLOAD_FRAMES

This register maintains the number of received IPv6 datagram frames without a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

ETH0_RXIPV6_NO_PAYLOAD_FRAMES

Receive IPV6 No Payload Frame Counter Register(22C_H) Reset Value: 0000 0000_H



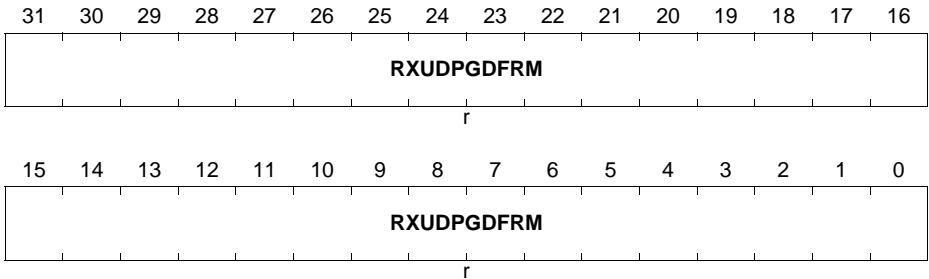
Field	Bits	Type	Description
RXIPV6NO PAYFRM	[31:0]	r	This field indicates the number of IPv6 datagram frames received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

RXUDP_GOOD_FRAMES

This register maintains the number of good IP datagrams with a good UDP payload. This counter is not updated when the counter is incremented.

ETH0_RXUDP_GOOD_FRAMES

RxUDP Good Frames Register (230_H) **Reset Value: 0000 0000_H**



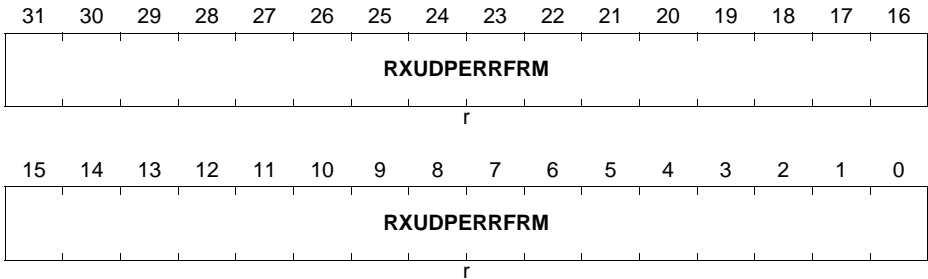
Field	Bits	Type	Description
RXUDPGDFRM	[31:0]	r	This field indicates the number of good IP datagrams with a good UDP payload. This counter is not updated when the counter is incremented.

RXUDP_ERROR_FRAMES

This register maintains the number of good IP datagrams whose UDP payload has a checksum error.

ETH0_RXUDP_ERROR_FRAMES

RxUDP Error Frames Register (234_H) Reset Value: 0000 0000_H



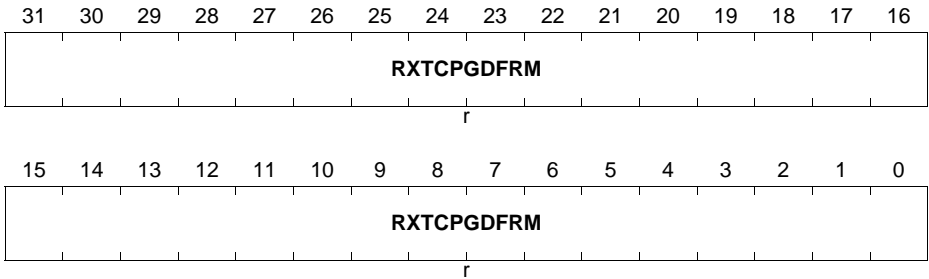
Field	Bits	Type	Description
RXUDPER RFRM	[31:0]	r	This field indicates the number of good IP datagrams whose UDP payload has a checksum error.

RXTCP_GOOD_FRAMES

This register maintains the number of good IP datagrams with a good TCP payload.

ETH0_RXTCP_GOOD_FRAMES

RxTCP Good Frames Register (238_H) **Reset Value: 0000 0000_H**



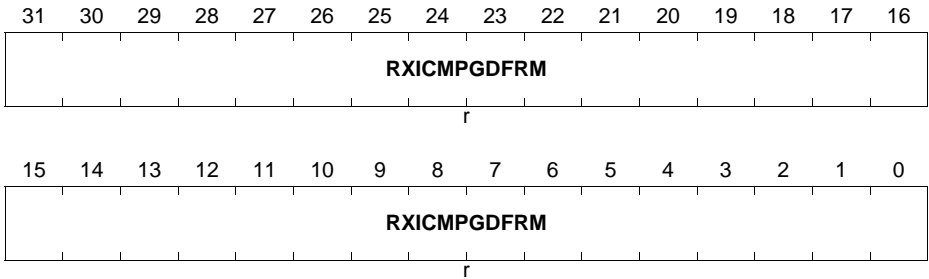
Field	Bits	Type	Description
RXTCPGD FRM	[31:0]	r	This field indicates the number of good IP datagrams with a good TCP payload.

RXICMP_GOOD_FRAMES

This register maintains the number of good IP datagrams with a good ICMP payload.

ETH0_RXICMP_GOOD_FRAMES

RxICMP Good Frames Register (240_H) **Reset Value: 0000 0000_H**



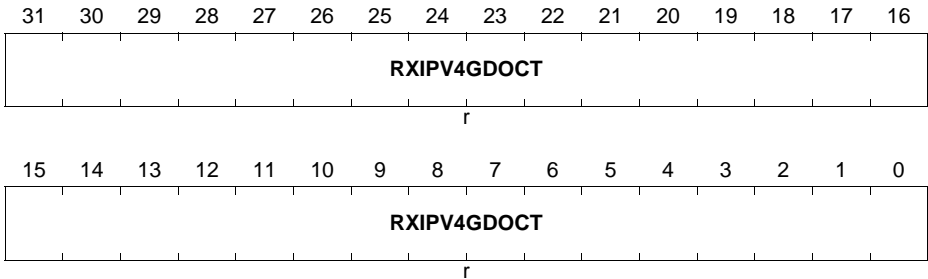
Field	Bits	Type	Description
RXICMPGDFRM	[31:0]	r	This field indicates the number of good IP datagrams with a good ICMP payload.

RXIPV4_GOOD_OCTETS

This register maintains the number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data.

ETH0_RXIPV4_GOOD_OCTETS

RxIPv4 Good Octets Register (250_H) Reset Value: 0000 0000_H



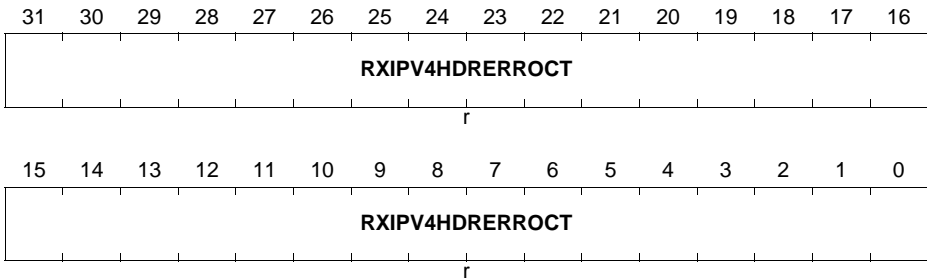
Field	Bits	Type	Description
RXIPV4GD OCT	[31:0]	r	This field indicates the number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV4_HEADER_ERROR_OCTETS

This register maintains the number of bytes received in IPv4 datagrams with header errors (checksum, length, or version mismatch). The value in the Length field of the IPv4 header is used to update this counter.

ETH0_RXIPV4_HEADER_ERROR_OCTETS

Receive IPV4 Header Error Octet Counter Register(254_H) Reset Value: 0000 0000_H



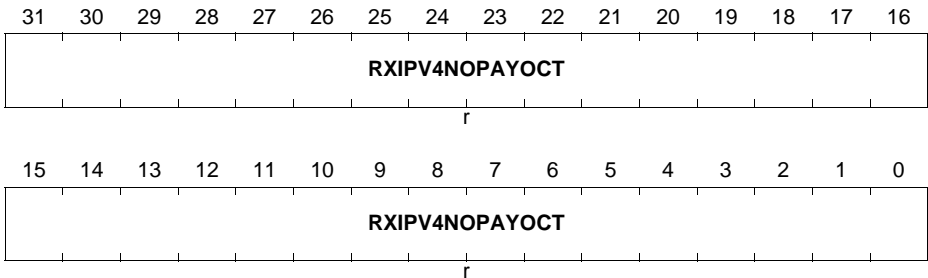
Field	Bits	Type	Description
RXIPV4HDRERROCT	[31:0]	r	This field indicates the number of bytes received in the IPv4 datagrams with header errors (checksum, length, or version mismatch). The value in the Length field of IPv4 header is used to update this counter. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV4_NO_PAYLOAD_OCTETS

This register maintains the number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv4 headers Length field is used to update this counter.

ETH0_RXIPV4_NO_PAYLOAD_OCTETS

Receive IPV4 No Payload Octet Counter Register(258_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXIPV4NO PAYOCT	[31:0]	r	This field indicates the number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv4 headers Length field is used to update this counter. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV4_FRAGMENTED_OCTETS

This register maintains the number of bytes received in fragmented IPv4 datagrams. The value in the IPv4 headers Length field is used to update this counter.

ETH0_RXIPV4_FRAGMENTED_OCTETS

Receive IPV4 Fragmented Octet Counter Register(25C_H) Reset Value: 0000 0000_H



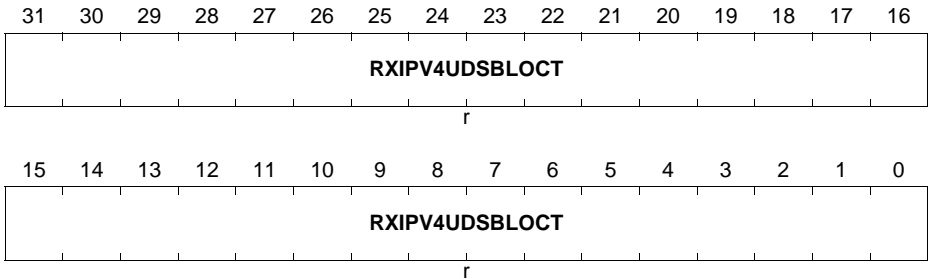
Field	Bits	Type	Description
RXIPV4FRAGOCT	[31:0]	r	This field indicates the number of bytes received in fragmented IPv4 datagrams. The value in the IPv4 headers Length field is used to update this counter. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS

This register maintains the number of bytes received in a UDP segment that had the UDP checksum disabled.

ETH0_RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS

Receive IPV4 Fragmented Octet Counter Register(260_H) Reset Value: 0000 0000_H



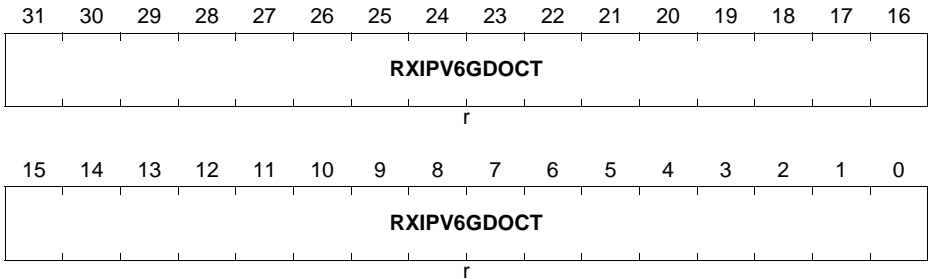
Field	Bits	Type	Description
RXIPV4UDSBLOCT	[31:0]	r	This field indicates the number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV6_GOOD_OCTETS

This register maintains the number of bytes received in good IPv6 datagrams encapsulating TCP, UDP or ICMPv6 data.

ETH0_RXIPV6_GOOD_OCTETS

RxIPv6 Good Octets Register (264_H) Reset Value: 0000 0000_H



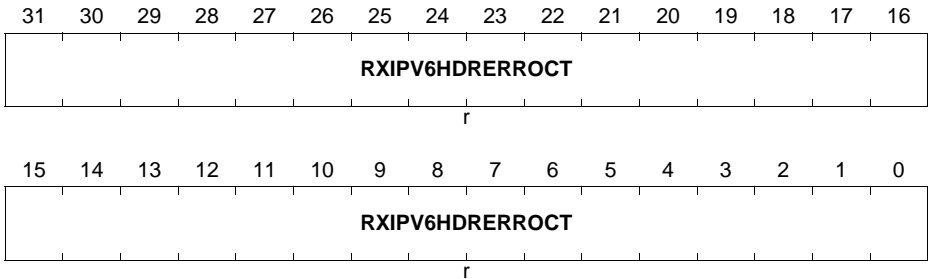
Field	Bits	Type	Description
RXIPV6GD OCT	[31:0]	r	This field indicates the number of bytes received in good IPv6 datagrams encapsulating TCP, UDP or ICMPv6 data. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV6_HEADER_ERROR_OCTETS

This register maintains the number of bytes received in IPv6 datagrams with header errors (length or version mismatch).

ETH0_RXIPV6_HEADER_ERROR_OCTETS

Receive IPV6 Header Error Octet Counter Register(268_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXIPV6HDRRROCT	[31:0]	r	This field indicates the number of bytes received in IPv6 datagrams with header errors (length or version mismatch). The value in the IPv6 headers Length field is used to update this counter. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV6_NO_PAYLOAD_OCTETS

This register maintains the number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload.

ETH0_RXIPV6_NO_PAYLOAD_OCTETS

Receive IPV6 No Payload Octet Counter Register(26C_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXIPV6NO PAYOCT	[31:0]	r	This field indicates the number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv6 headers Length field is used to update this counter. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

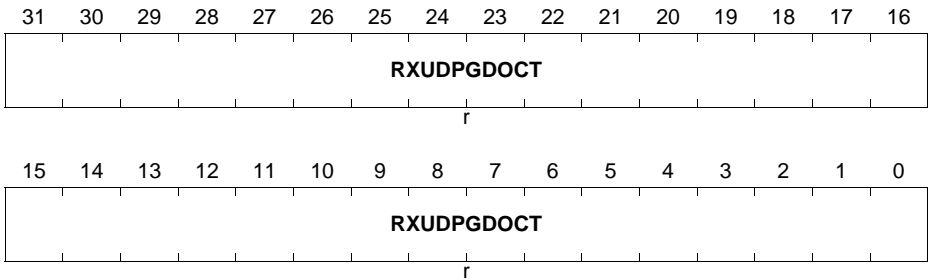
RXUDP_GOOD_OCTETS

This register maintains the number of bytes received in a good UDP segment.

ETH0_RXUDP_GOOD_OCTETS

Receive UDP Good Octets Register (270_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXUDPGD OCT	[31:0]	r	This field indicates the number of bytes received in a good UDP segment. This counter does not count IP header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

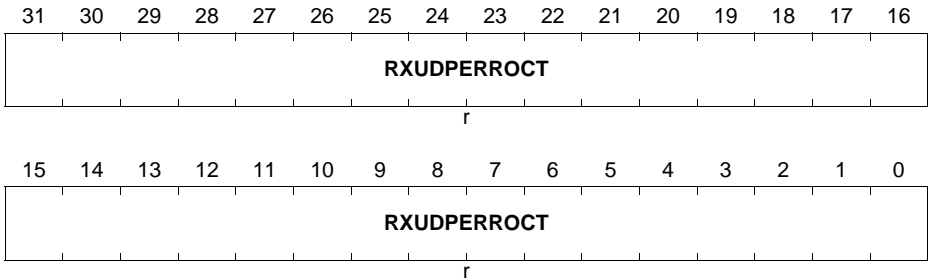
RXUDP_ERROR_OCTETS

This register maintains the number of bytes received in a UDP segment with checksum errors.

ETH0_RXUDP_ERROR_OCTETS

Receive UDP Error Octets Register (274_H)

Reset Value: 0000 0000_H



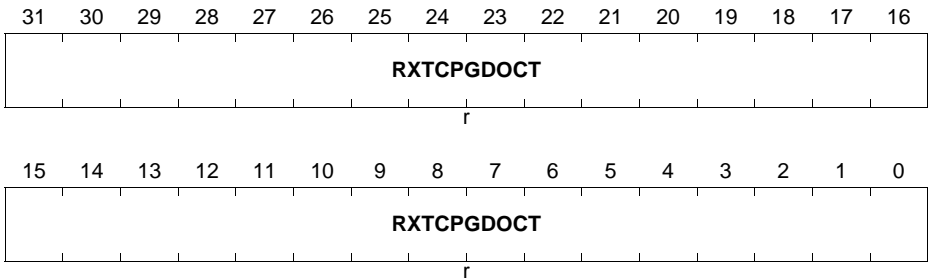
Field	Bits	Type	Description
RXUDPERROCT	[31:0]	r	This field indicates the number of bytes received in a UDP segment with checksum errors. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXTCP_GOOD_OCTETS

This register maintains the number of bytes received in a good TCP segment.

ETH0_RXTCP_GOOD_OCTETS

Receive TCP Good Octets Register (278_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RXTCPGD OCT	[31:0]	r	This field indicates the number of bytes received in a good TCP segment. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

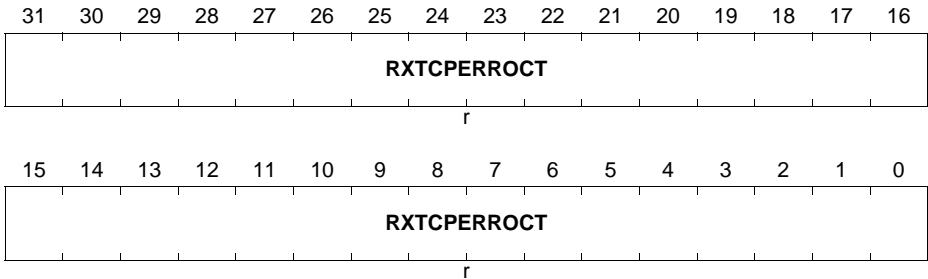
RXTCP_ERROR_OCTETS

This register maintains the number of bytes received in a TCP segment with checksum errors.

ETH0_RXTCP_ERROR_OCTETS

Receive TCP Error Octets Register (27C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXTCPERROCT	[31:0]	r	This field indicates the number of bytes received in a TCP segment with checksum errors. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

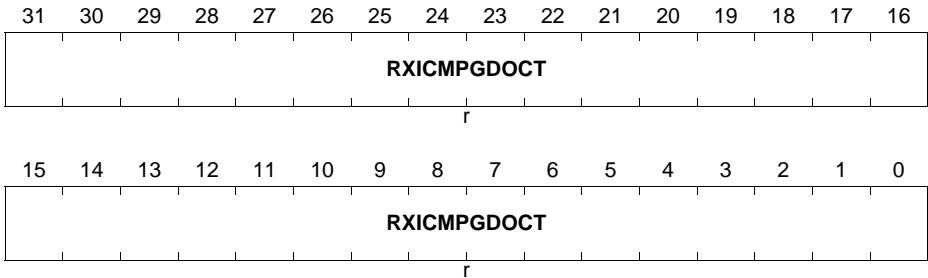
RXICMP_GOOD_OCTETS

This register maintains the number of bytes received in a good ICMP segment.

ETH0_RXICMP_GOOD_OCTETS

Receive ICMP Good Octets Register (280_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXICMPG DOCT	[31:0]	r	This field indicates the number of bytes received in a good ICMP segment. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

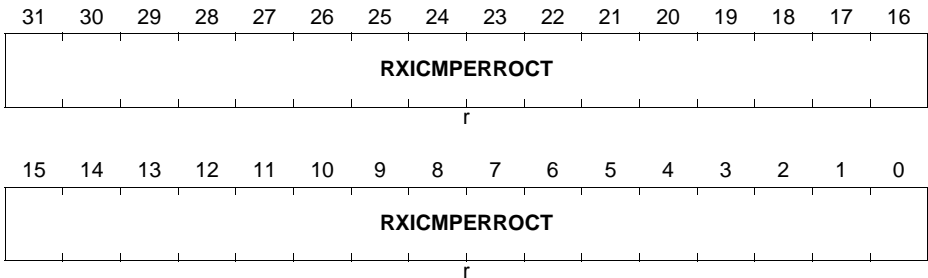
RXICMP_ERROR_OCTETS

This register maintains the number of bytes received in a ICMP segment with checksum errors. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

ETH0_RXICMP_ERROR_OCTETS

Receive ICMP Error Octets Register (284_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXICMPE RROCT	[31:0]	r	Number of bytes received in an ICMP segment with checksum errors

TIMESTAMP_CONTROL

This register controls the operation of the System Time generator and the processing of PTP packets for timestamping in the Receiver.

ETH0_TIMESTAMP_CONTROL

Timestamp Control Register

(700_H)

Reset Value: 0000 2000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved_23_19													TSE NMA CAD DR	SNAPTYP SEL	
r													rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSM STR ENA	TSE VNT ENA	TSIP V4E NA	TSIP V6E NA	TSIP ENA	TSV ER2 ENA	TSC TRL SSR	TSE NAL L	Reserved_ 7_6	TSA DDR EG	TST RIG	TSU PDT	TSIN IT	TSC FUP DT	TSE NA	
rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
TSENA	0	rw	<p>Timestamp Enable</p> <p>When set, the timestamp is added for the transmit and receive frames. When disabled, timestamp is not added for the transmit and receive frames and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode. On the receive side, the MAC processes the 1588 frames only if this bit is set.</p>
TSCFUPDT	1	rw	<p>Timestamp Fine or Coarse Update</p> <p>When set, this bit indicates that the system times update should be done using the fine update method. When reset, it indicates the system timestamp update should be done using the Coarse method.</p>
TSINIT	2	rw	<p>Timestamp Initialize</p> <p>When set, the system time is initialized (overwritten) with the value specified in the SYSTEM_TIME_SECONDS_UPDATE Register and SYSTEM_TIME_NANOSECONDS_UPDATE Register. This bit should be read zero before updating it. This bit is reset when the initialization is complete.</p>

Field	Bits	Type	Description
TSUPDT	3	rw	Timestamp Update When set, the system time is updated (added or subtracted) with the value specified in System Time_Seconds_Update Register and SYSTEM_TIME_NANOSECONDS_UPDATE Register. This bit should be read zero before updating it. This bit is reset when the update is completed in hardware.
TSTRIG	4	rw	Timestamp Interrupt Trigger Enable When set, the timestamp interrupt is generated when the System Time becomes greater than the value written in the Target Time register. This bit is reset after the generation of the Timestamp Trigger Interrupt.
TSADDRE G	5	rw	Addend Reg Update When set, the content of the TIMESTAMP_ADDEND register is updated in the PTP block for fine correction. This is cleared when the update is completed. This register bit should be zero before setting it.
Reserved_7_6	[7:6]	r	Reserved
TSENALL	8	rw	Enable Timestamp for All Frames When set, the timestamp snapshot is enabled for all frames received by the MAC.
TSCTRLS SR	9	rw	Timestamp Digital or Binary Rollover Control When set, the Timestamp Low register rolls over after 3B9A C9FFH value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When reset, the rollover value of sub-second register is 7FFF FFFFH. The sub-second increment has to be programmed correctly depending on the PTP reference clock frequency and the value of this bit.
TSVER2E NA	10	rw	Enable PTP packet Processing for Version 2 Format When set, the PTP packets are processed using the 1588 version 2 format. Otherwise, the PTP packets are processed using the version 1 format.
TSIPENA	11	rw	Enable Processing of PTP over Ethernet Frames When set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet frames. When this bit is clear, the MAC ignores the PTP over Ethernet packets.

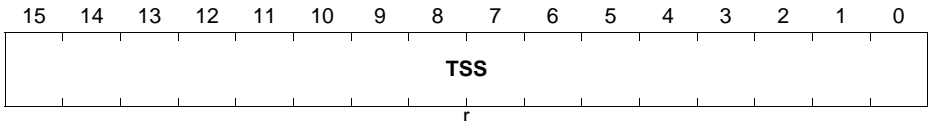
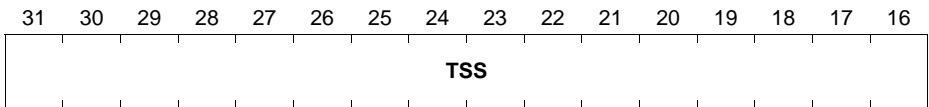
Field	Bits	Type	Description
TSIPV6EN A	12	rw	Enable Processing of PTP Frames Sent Over IPv6-UDP When set, the MAC receiver processes PTP packets encapsulated in UDP over IPv6 packets. When this bit is clear, the MAC ignores the PTP transported over UDP-IPv6 packets.
TSIPV4EN A	13	rw	Enable Processing of PTP Frames Sent over IPv4-UDP When set, the MAC receiver processes the PTP packets encapsulated in UDP over IPv4 packets. When this bit is clear, the MAC ignores the PTP transported over UDP-IPv4 packets. This bit is set by default.
TSEVNTENA	14	rw	Enable Timestamp Snapshot for Event Messages When set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When reset, the snapshot is taken for all messages except Announce, Management, and Signaling.
TSMSTRENA	15	rw	Enable Snapshot for Messages Relevant to Master When set, the snapshot is taken only for the messages relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.
SNAPTYPSEL	[17:16]	rw	Select PTP packets for Taking Snapshots These bits along with Bits 15 and 14 decide the set of PTP packet types for which snapshot needs to be taken.
TSENMADDR	18	rw	Enable MAC address for PTP Frame Filtering When set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP frames when PTP is directly sent over Ethernet.
Reserved_23_19	[31:19]	r	Reserved

SYSTEM_TIME_SECONDS

The System Time -Seconds register, along with System-TimeNanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies .

ETH0_SYSTEM_TIME_SECONDS

System Time - Seconds Register (708_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TSS	[31:0]	r	Timestamp Second The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

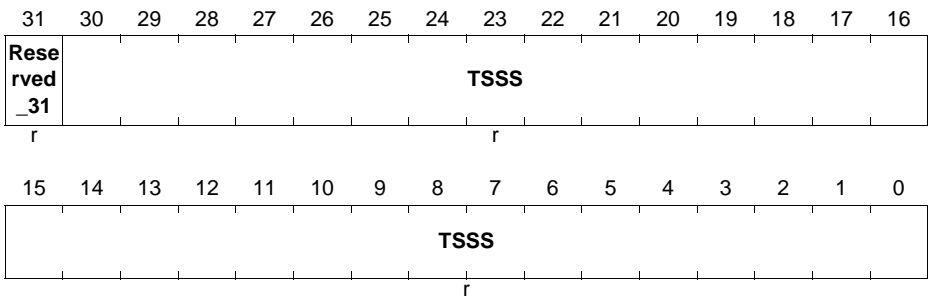
SYSTEM_TIME_NANOSECONDS

The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When `TIMESTAMP_CONTROL.TSCTRLSSR` is set, each bit represents 1 ns and the maximum value is `3B9A C9FFH`, after which it rolls-over to zero.

ETH0_SYSTEM_TIME_NANOSECONDS

System Time Nanoseconds Register (70C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TSSS	[30:0]	r	Timestamp Sub Seconds The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When <code>TIMESTAMP_CONTROL.TSCTRLSSR</code> is set, each bit represents 1 ns and the maximum value is <code>3B9A C9FFH</code> , after which it rolls-over to zero.
Reserved_31	31	r	Reserved

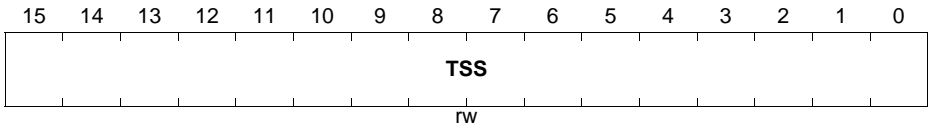
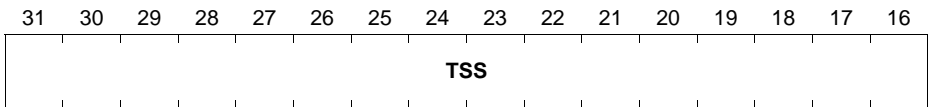
SYSTEM_TIME_SECONDS_UPDATE

The System Time - Seconds Update register, along with the System_Time_Nanoseconds_Update register, initializes or updates the system time maintained by the MAC. You must write both of these registers before setting the `TIMESTAMP_CONTROL.TSINIT` or `TIMESTAMP_CONTROL.TSUPDT` bits.

ETH0_SYSTEM_TIME_SECONDS_UPDATE

System Time - Seconds Update Register(710_H)

Reset Value: 0000 0000_H



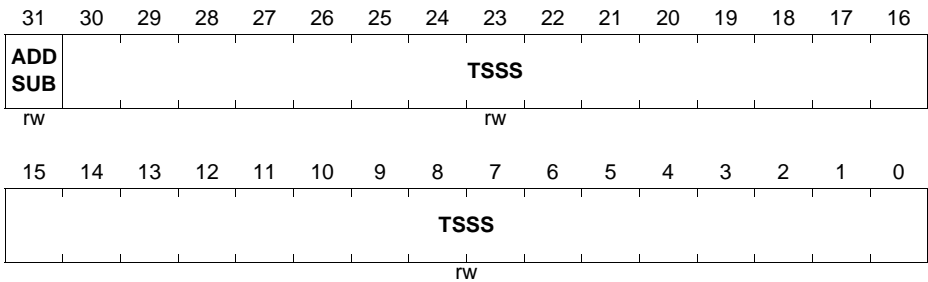
Field	Bits	Type	Description
TSS	[31:0]	rw	Timestamp Second The value in this field indicates the time in seconds to be initialized or added to the system time.

SYSTEM_TIME_NANOSECONDS_UPDATE

The System Time - Seconds Update register, along with the System_Time_Nanoseconds_Update register, initializes or updates the system time maintained by the MAC. You must write both of these registers before setting the `TIMESTAMP_CONTROL.TSINIT` or `TIMESTAMP_CONTROL.TSUPDT` bits.

ETH0_SYSTEM_TIME_NANOSECONDS_UPDATE

System Time Nanoseconds Update Register(714_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TSSS	[30:0]	rw	<p>Timestamp Sub Second</p> <p>The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When <code>TIMESTAMP_CONTROL.TSCTRLSSR</code> is set, each bit represents 1 ns and the programmed value should not exceed 3B9A C9FFH.</p>
ADDSUB	31	rw	<p>Add or subtract time</p> <p>When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register.</p>

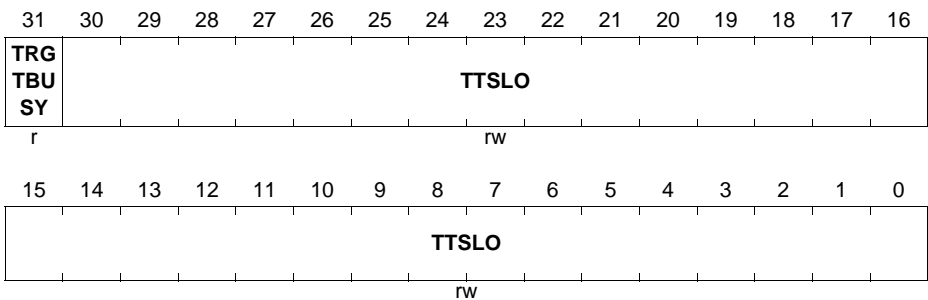
TARGET_TIME_NANOSECONDS

The Target Time Seconds register, along with Target Time Nanoseconds register, is used to schedule an interrupt event triggered by the TimestampStatus.TSTARGET bit when Advanced Timestamping is enabled; otherwise, the INTERRUPT_STATUS.TSIS will trigger the interrupt when the system time exceeds the value programmed in these registers.

ETH0_TARGET_TIME_NANOSECONDS

Target Time Nanoseconds Register (720_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TTSLO	[30:0]	rw	<p>Target Timestamp Low Register</p> <p>This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the both Target Timestamp registers, then based on the PPS_CONTROL.TPPSRGTMODSEL0 field, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled).</p> <p>This value should not exceed 3B9A C9FFH when TIMESTAMP_CONTROL.TSCTRLSSR is set. The actual start or stop time of the PPS signal output may have an error margin up to one unit of sub-second increment value.</p>

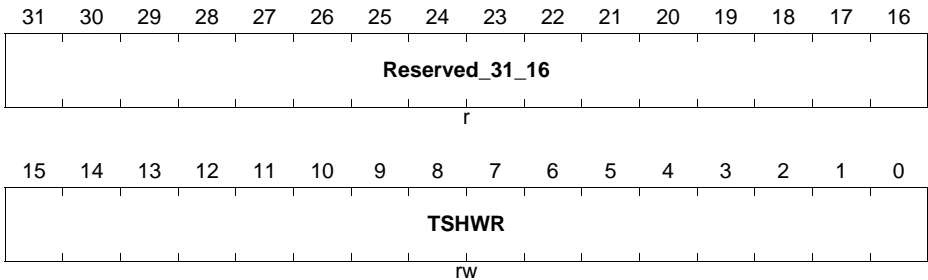
Field	Bits	Type	Description
TRGTBUSY	31	r	<p>Target Time Register Busy</p> <p>The MAC sets this bit when the PPS_CONTROL.PPSCMD field is programmed to 010B or 011B. Programming the PPSCMD field to 010B or 011B, instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers to the PTP clock domain. The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted. This bit is reserved when the Enable Flexible Pulse-Per-Second Output feature is not selected.</p>

SYSTEM_TIME_HIGHER_WORD_SECONDS

This register contains the most significant 16-bits of the timestamp seconds value.

ETH0_SYSTEM_TIME_HIGHER_WORD_SECONDS

System Time - Higher Word Seconds Register (724_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TSHWR	[15:0]	rw	Timestamp Higher Word Register This field contains the most significant 16-bits of the timestamp seconds value. The register is directly written to initialize the value. This register is incremented when there is an overflow from the 32-bits of the SYSTEM_TIME_SECONDS register.
Reserved_31_16	[31:16]	r	Reserved

TIMESTAMP_STATUS

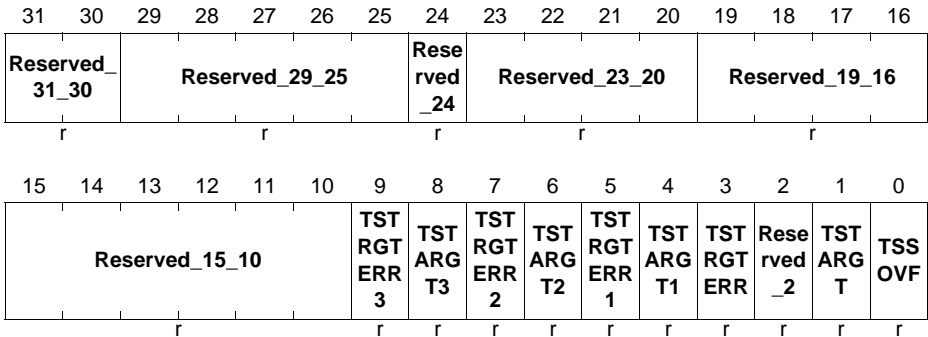
This register contains the Timestamp status information. All bits except Bits[27:25] gets cleared when the CPU reads this register.

ETH0_TIMESTAMP_STATUS

Timestamp Status Register

(728_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TSSOVF	0	r	Timestamp Seconds Overflow When set, this bit indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond FFFF_FFFFH.
TSTARGT	1	r	Timestamp Target Time Reached When set, this bit indicates that the value of system time is greater or equal to the value specified in the Target_Time_Seconds Register and Target Time Nanoseconds Register.
Reserved_2	2	r	Reserved
TSTRGTE RR	3	r	Timestamp Target Time Error This bit is set when the target time, being programmed in Target Time Registers, is already elapsed. This bit is cleared when read by the application.

Field	Bits	Type	Description
TSTARTG1	4	r	Timestamp Target Time Reached for Target Time PPS1 When set, this bit indicates that the value of system time is greater than or equal to the value specified in PPS1_Target_Time_High Register and PPS1_Target_Time_Low Register.
TSTRGTE RR1	5	r	Timestamp Target Time Error This bit is set when the target time, being programmed in Register 480 and Register 481, is already elapsed. This bit is cleared when read by the application.
TSTARTG2	6	r	Timestamp Target Time Reached for Target Time PPS2 When set, this bit indicates that the value of system time is greater than or equal to the value specified in Register 488 [PPS2 Target Time High Register] and Register 489 [PPS2 Target Time Low Register].
TSTRGTE RR2	7	r	Timestamp Target Time Error This bit is set when the target time, being programmed in Register 488 and Register 489, is already elapsed. This bit is cleared when read by the application.
TSTARTG3	8	r	Timestamp Target Time Reached for Target Time PPS3 When set, this bit indicates that the value of system time is greater than or equal to the value specified in Register 496 [PPS3 Target Time High Register] and Register 497 [PPS3 Target Time Low Register].
TSTRGTE RR3	9	r	Timestamp Target Time Error This bit is set when the target time, being programmed in Register 496 and Register 497, is already elapsed. This bit is cleared when read by the application.
Reserved_15_10	[15:10]	r	Reserved
Reserved_19_16	[19:16]	r	Reserved
Reserved_23_20	[23:20]	r	Reserved
Reserved_24	24	r	Reserved

Field	Bits	Type	Description
Reserved_29_25	[29:25]	r	Reserved
Reserved_31_30	[31:30]	r	Reserved

BUS_MODE

The Bus Mode register establishes the bus operating modes for the DMA.

ETH0_BUS_MODE

Bus Mode Register (1000_H) Reset Value: 0002 0101_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved_31_30		PRWG		TXP R	MB	AAL	PBL x8	USP	RPBL						FB
r		r		rw	rw	rw	rw	rw	rw						rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR		PBL						ATD S	DSL				DA	SWR	
rw		rw						rw	rw				rw	rw	

Field	Bits	Type	Description
SWR	0	rw	<p>Software Reset</p> <p>When this bit is set, the MAC DMA Controller resets the logic and all internal registers of the MAC. It is cleared automatically after the reset operation has completed in all of the DWC_ETH clock domains. Before reprogramming any register of the DWC_ETH, you should read a zero (0) value in this bit .</p> <p>Note:</p> <p>* The reset operation is completed only when all resets in all active clock domains are de-asserted. Therefore, it is essential that all the PHY inputs clocks (applicable for the selected PHY interface) are present for the software reset completion.</p>

Field	Bits	Type	Description
DA	1	rw	<p>DMA Arbitration Scheme</p> <p>This bit specifies the arbitration scheme between the transmit and receive paths of Channel 0.</p> <ul style="list-style-type: none"> * 0: Weighted round-robin with Rx:Tx or Tx:Rx. <p>The priority between the paths is according to the priority specified in BUS_MODE.PR and priority weights specified in BUS_MODE.TXPR.</p> <ul style="list-style-type: none"> * 1: Fixed priority. <p>The transmit path has priority over receive path when BUS_MODE.TXPR is set. Otherwise, receive path has priority over the transmit path.</p>
DSL	[6:2]	rw	<p>Descriptor Skip Length</p> <p>This bit specifies the number of Words to skip between two unchained descriptors. The address skipping starts from the end of current descriptor to the start of next descriptor. When the DSL value is equal to zero, then the descriptor table is taken as contiguous by the DMA in Ring mode.</p>
ATDS	7	rw	<p>Alternate Descriptor Size</p> <p>When set, the size of the alternate descriptor increases to 32 bytes. This is required when the Advanced Timestamp feature or Full IPC Offload Engine is enabled in the receiver.</p>

Field	Bits	Type	Description
PBL	[13:8]	rw	<p>Programmable Burst Length</p> <p>These bits indicate the maximum number of beats to be transferred in one DMA transaction. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a Burst transfer on the bus. PBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. When BUS_MODE.USP is set high, this BUS_MODE.PBL value is applicable only for Tx DMA transactions.</p> <p>If the number of beats to be transferred is more than 32, then perform the following steps:</p> <ol style="list-style-type: none"> 1. Set the PBLx8 mode. 2. Set the PBL. <p>For example, if the maximum number of beats to be transferred is 64, then first set PBLx8 to 1 and then set PBL to 8. The PBL values have the following limitation: The maximum number of possible beats (PBL) is limited by the size of the Tx FIFO and Rx FIFO in the MTL layer and the data bus width on the DMA. The FIFO has a constraint that the maximum beat supported is half the depth of the FIFO, except when specified.</p> <p>For different data bus widths and FIFO sizes, the valid PBL range (including x8 mode) is provided in the following list. If the PBL is common for both transmit and receive DMA, the minimum Rx FIFO and Tx FIFO depths must be considered.</p> <p>Note: In the half-duplex mode, the valid PBL range specified in the following list is applicable only for Tx FIFO.</p>
PR	[15:14]	rw	<p>Priority Ratio</p> <p>These bits control the priority ratio in the weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when Bit 1 (DA) is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether Bit 27 (TXPR) is reset or set.</p> <ul style="list-style-type: none"> * 00B: The Priority Ratio is 1:1. * 01B: The Priority Ratio is 2:1. * 10B: The Priority Ratio is 3:1. * 11B: The Priority Ratio is 4:1.

Field	Bits	Type	Description
FB	16	rw	<p>Fixed Burst</p> <p>This bit controls whether the Bus Master interface performs fixed burst transfers or not. When set, the AHB interface uses only SINGLE, INCR4, INCR8, or INCR16 during start of the normal burst transfers. When reset, the AHB interface uses SINGLE and INCR burst transfer operations.</p>
RPBL	[22:17]	rw	<p>Rx DMA PBL</p> <p>This field indicates the maximum number of beats to be transferred in one Rx DMA transaction. This is the maximum value that is used in a single block Read or Write.</p> <p>The Rx DMA always attempts to burst as specified in the RPBL bit each time it starts a Burst transfer on the bus. You can program RPBL with values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. This field is valid and applicable only when USP is set high.</p>
USP	23	rw	<p>Use Seperate PBL</p> <p>When set high, this bit configures the Rx DMA to use the value configured in Bits[22:17] as PBL. The BUS_MODE.PBL value is applicable only to the Tx DMA operations.</p> <p>When reset to low, the BUS_MODE.PBL value is applicable for both DMA engines.</p>
PBLx8	24	rw	<p>PBLx8 Mode</p> <p>When set high, this bit multiplies the programmed PBL value (Bits[22:17] and Bits[13:8]) eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the BUS_MODE.PBL value.</p>
AAL	25	rw	<p>Address Aligned Beats</p> <p>When this bit is set high and the BUS_MODE.FB bit is equal to 1, the AHB interface generates all bursts aligned to the start address LS bits. If the BUS_MODE.FB bit is equal to 0, the first burst (accessing the data buffer's start address) is not aligned, but subsequent bursts are aligned to the address.</p>

Field	Bits	Type	Description
MB	26	rw	Mixed Burst When this bit is set high and the BUS_MODE.FB bit is low, the Bus Master interface starts all bursts of length more than 16 with INCR (undefined burst) whereas it reverts to fixed burst transfers (INCRx and SINGLE) for burst length of 16 and less.
TXPR	27	rw	Transmit Priority When set, this bit indicates that the transmit DMA has higher priority than the receive DMA during arbitration for the system-side bus.
PRWG	[29:28]	r	Channel Priority Weights This field sets the priority weights for Channel 0 during the round-robin arbitration between the DMA channels for the system bus. * 00B: The priority weight is 1. * 01B: The priority weight is 2. * 10B: The priority weight is 3. * 11B: The priority weight is 4.
Reserved_31_30	[31:30]	r	Reserved

RECEIVE_POLL_DEMAND

The Receive Poll Demand register enables the receive DMA to check for new descriptors. This command is used to wake up the Rx DMA from the SUSPEND state. The RxDMA can go into the SUSPEND state only because of the unavailability of descriptors it owns.

ETH0_RECEIVE_POLL_DEMAND

Receive Poll Demand Register (1008_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
RPD	[31:0]	rw	<p>Receive Poll Demand</p> <p>When these bits are written with any value, the DMA reads the current descriptor pointed to by the Current Host Receive Descriptor Register. If that descriptor is not available (owned by the CPU), the reception returns to the Suspended state and STATUS.RU is not asserted. If the descriptor is available, the Rx DMA returns to the active state.</p>

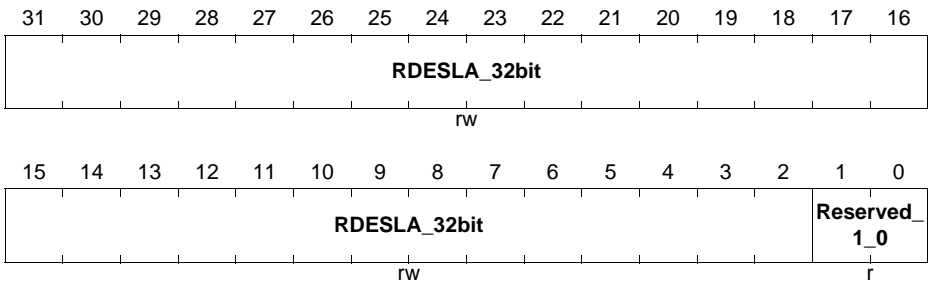
RECEIVE_DESCRIPTOR_LIST_ADDRESS

The Receive Descriptor List Address register points to the start of the Receive Descriptor List. The descriptor lists reside in the XMC4300's physical memory space and must be Word-aligned . The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, Bit 1 (SR) is set to zero in the Operation Mode Register. When stopped, this register can be written with a new descriptor list address. When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address. If this register is not changed when the SR bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

ETH0_RECEIVE_DESCRIPTOR_LIST_ADDRESS

Receive Descriptor Address Register (100C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
Reserved_1_0	[1:0]	r	Reserved
RDESLA_32bit	[31:2]	rw	Start of Receive List This field contains the base address of the first descriptor in the Receive Descriptor list. The LSB bits (1:0) for 32-bit bus width are ignored and internally taken as all-zero by the DMA. Therefore, these LSB bits are read-only (RO).

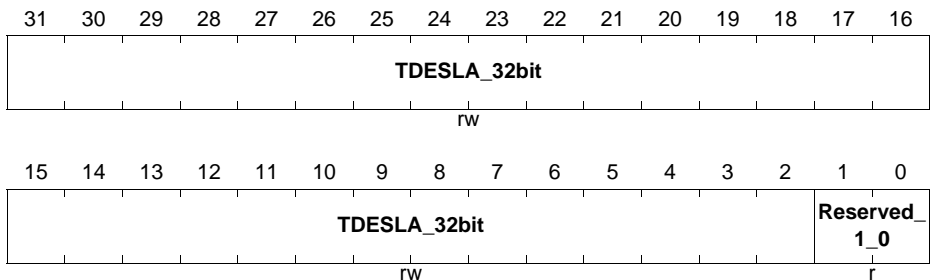
TRANSMIT_DESCRIPTOR_LIST_ADDRESS

The Transmit Descriptor List Address register points to the start of the Transmit Descriptor List. The descriptor lists reside in the XMC4300's physical memory space and must be Word-aligned . The DMA internally converts it to bus width aligned address by making the corresponding LSB to low. You can write to this register only when the Tx DMA has stopped, that is, OPERATION_MODE.ST is set to zero. When stopped, this register can be written with a new descriptor list address. When you set the OPERATION_MODE.ST bit to 1, the DMA takes the newly programmed descriptor base address. If this register is not changed when the ST bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

ETH0_TRANSMIT_DESCRIPTOR_LIST_ADDRESS

Transmit descriptor Address Register (1010_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
Reserved_1_0	[1:0]	r	Reserved
TDESLA_32bit	[31:2]	rw	Start of Transmit List This field contains the base address of the first descriptor in the Transmit Descriptor list. The LSB bits (1:0) are ignored and are internally taken as all-zero by the DMA. Therefore, these LSB bits are read-only (RO).

STATUS

The STATUS register contains all status bits that the DMA reports to the CPU. The Software driver reads this register during an interrupt service routine or polling. Most of the fields in this register cause the CPU to be interrupted. The bits of this register are not cleared when read. Writing 1 to (unreserved) Bits[16:0] of this register clears these bits and writing 0 has no effect. Each field (Bits[16:0]) can be masked by masking the appropriate bit in the Interrupt Enable Register.

ETH0_STATUS

Status Register

(1014_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved_31	Reserved_30	TTI	EPI	EMI	Reserved_26	EB			TS		RS			NIS	
r	r	r	r	r	r	r			r		r			rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIS	ERI	FBI	Reserved_12_11	ETI	RWT	RPS	RU	RI	UNF	OVF	TJT	TU	TPS	TI	
rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TI	0	rw	Transmit Interrupt This bit indicates that the frame transmission is complete. When transmission is complete, the Bit 31 (Interrupt on Completion) of TDES1 is reset in the first descriptor, and the specific frame status information is updated in the descriptor.
TPS	1	rw	Transmit Process Stopped This bit is set when the transmission is stopped.
TU	2	rw	Transmit Buffer Unavailable This bit indicates that the CPU owns the Next Descriptor in the Transmit List and the DMA cannot acquire it. Transmission is suspended. The TS bit field explains the Transmit Process state transitions. To resume processing Transmit descriptors, the CPU should change the ownership of the descriptor by setting TDES0[31] and then issue a Transmit Poll Demand command.

Field	Bits	Type	Description
TJT	3	rw	<p>Transmit Jabber Timeout</p> <p>This bit indicates that the Transmit Jabber Timer expired, which happens when the frame size exceeds 2,048 (10,240 bytes when the Jumbo frame is enabled). When the Jabber Timeout occurs, the transmission process is aborted and placed in the Stopped state. This causes the Transmit Jabber Timeout TDES0[14] flag to assert.</p>
OVF	4	rw	<p>Receive Overflow</p> <p>This bit indicates that the Receive Buffer had an Overflow during frame reception. If the partial frame is transferred to the application, the overflow status is set in RDES0[11].</p>
UNF	5	rw	<p>Transmit Underflow</p> <p>This bit indicates that the Transmit Buffer had an Underflow during frame transmission. Transmission is suspended and an Underflow Error TDES0[1] is set.</p>
RI	6	rw	<p>Receive Interrupt</p> <p>This bit indicates that the frame reception is complete. When reception is complete, the Bit 31 of RDES1 (Disable Interrupt on Completion) is reset in the last Descriptor, and the specific frame status information is updated in the descriptor. The reception remains in the Running state.</p>
RU	7	rw	<p>Receive Buffer Unavailable</p> <p>This bit indicates that the CPU owns the Next Descriptor in the Receive List and the DMA cannot acquire it. The Receive Process is suspended. To resume processing Receive descriptors, the CPU should change the ownership of the descriptor and issue a Receive Poll Demand command. If no Receive Poll Demand is issued, the Receive Process resumes when the next recognized incoming frame is received. This bit is set only when the previous Receive Descriptor is owned by the DMA.</p>
RPS	8	rw	<p>Receive Process Stopped</p> <p>This bit is asserted when the Receive Process enters the Stopped state.</p>

Field	Bits	Type	Description
RWT	9	rw	Receive Watchdog Timeout This bit is asserted when a frame with length greater than 2,048 bytes is received (10, 240 when Jumbo Frame mode is enabled).
ETI	10	rw	Early Transmit Interrupt This bit indicates that the frame to be transmitted is fully transferred to the MTL Transmit FIFO.
Reserved_12_11	[12:11]	r	Reserved
FBI	13	rw	Fatal Bus Error Interrupt This bit indicates that a bus error occurred, as described in the EB bit field. When this bit is set, the corresponding DMA engine disables all of its bus accesses.
ERI	14	rw	Early Receive Interrupt This bit indicates that the DMA had filled the first data buffer of the packet. STATUS.RI automatically clears this bit.
AIS	15	rw	Abnormal Interrupt Summary Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in Interrupt Enable Register: <ul style="list-style-type: none"> * STATUS.TPS: Transmit Process Stopped * STATUS.TJT: Transmit Jabber Timeout * STATUS.OVF : Receive FIFO Overflow * STATUS.UNF: Transmit Underflow * STATUS.RU: Receive Buffer Unavailable * STATUS.RPS: Receive Process Stopped * STATUS.RWT: Receive Watchdog Timeout * STATUS.ETI: Early Transmit Interrupt * STATUS.FBI: Fatal Bus Error Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit and must be cleared each time a corresponding bit, which causes AIS to be set, is cleared.

Field	Bits	Type	Description
NIS	16	rw	<p>Normal Interrupt Summary</p> <p>Normal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the Interrupt Enable Register:</p> <ul style="list-style-type: none"> * STATUS.TI: Transmit Interrupt * STATUS.TU: Transmit Buffer Unavailable * STATUS.RI: Receive Interrupt * STATUS.ERI: Early Receive Interrupt <p>Only unmasked bits (interrupts for which interrupt enable is set in Register 7) affect the Normal Interrupt Summary bit.</p> <p>This is a sticky bit and must be cleared (by writing 1 to this bit) each time a corresponding bit, which causes NIS to be set, is cleared.</p>
RS	[19:17]	r	<p>Received Process State</p> <p>This field indicates the Receive DMA FSM state. This field does not generate an interrupt.</p> <ul style="list-style-type: none"> * 000B: Stopped: Reset or Stop Receive Command issued * 001B: Running: Fetching Receive Transfer Descriptor * 010B: Reserved for future use * 011B: Running: Waiting for receive packet * 100B: Suspended: Receive Descriptor Unavailable * 101B: Running: Closing Receive Descriptor * 110B: TIME_STAMP write state * 111B: Running: Transferring the receive packet data from receive buffer to the XMC4300's memory

Field	Bits	Type	Description
TS	[22:20]	r	<p>Transmit Process State</p> <p>This field indicates the Transmit DMA FSM state. This field does not generate an interrupt.</p> <ul style="list-style-type: none"> * 000B: Stopped; Reset or Stop Transmit Command issued * 001B: Running; Fetching Transmit Transfer Descriptor * 010B: Running; Waiting for status * 011B: Running; Reading Data from the memory buffer and queuing it to transmit buffer (Tx FIFO) * 100B: TIME_STAMP write state * 101B: Reserved for future use * 110B: Suspended; Transmit Descriptor Unavailable or Transmit Buffer Underflow * 111B: Running; Closing Transmit Descriptor
EB	[25:23]	r	<p>Error Bits</p> <p>This field indicates the type of error that caused a Bus Error, for example, error response on the AHB interface. This field is valid only when Bit 13 (FBI) is set. This field does not generate an interrupt.</p> <ul style="list-style-type: none"> * Bit 23 <ul style="list-style-type: none"> - 1: Error during data transfer by the Tx DMA - 0: Error during data transfer by the Rx DMA * Bit 24 <ul style="list-style-type: none"> - 1: Error during read transfer - 0: Error during write transfer * Bit 25 <ul style="list-style-type: none"> - 1: Error during descriptor access - 0: Error during data buffer access
Reserved_26	26	r	Reserved
EMI	27	r	<p>ETH MMC Interrupt</p> <p>This bit reflects an interrupt event in the MMC module of the DWC_ETH. The software must read the corresponding registers in the DWC_ETH to get the exact cause of interrupt and clear the source of interrupt to make this bit as 0. The interrupt signal from the DWC_ETH subsystem is high when this bit is high.</p>

Field	Bits	Type	Description
EPI	28	r	<p>ETH PMT Interrupt</p> <p>This bit indicates an interrupt event in the PMT module of the ETH. The software must read the PMT Control and STATUS Register in the MAC to get the exact cause of interrupt and clear its source to reset this bit to 0. The interrupt signal from the ETH subsystem is high when this bit is high.</p> <p>Note: This interrupt is different from the Power Management interrupt.</p>
TTI	29	r	<p>Timestamp Trigger Interrupt</p> <p>This bit indicates an interrupt event in the Timestamp Generator block of ETH. The software must read the corresponding registers in the ETH to get the exact cause of interrupt and clear its source to reset this bit to 0. When this bit is high, the interrupt signal from the ETH subsystem is high.</p>
Reserved_30	30	r	Reserved
Reserved_31	31	r	Reserved

OPERATION_MODE

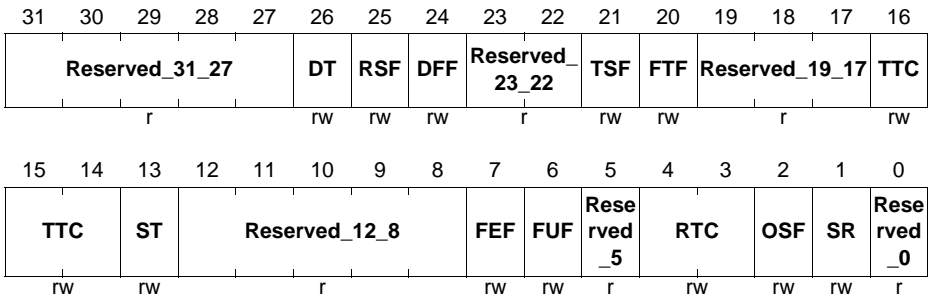
The Operation Mode register establishes the Transmit and Receive operating modes and commands. This register should be the last CSR to be written as part of the DMA initialization.

ETH0_OPERATION_MODE

Operation Mode Register

(1018_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
Reserved_0	0	r	Reserved

Field	Bits	Type	Description
SR	1	rw	<p>Start or Stop Receive</p> <p>When this bit is set, the Receive process is placed in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes the incoming frames. The descriptor acquisition is attempted from the current position in the list, which is the address set by Receive Descriptor List Address Register or the position retained when the Receive process was previously stopped. If the DMA does not own the descriptor, reception is suspended and STATUS.RU is set. The Start Receive command is effective only when the reception has stopped. If the command is issued before setting Receive Descriptor List Address Register, the DMA behavior is unpredictable.</p> <p>When this bit is cleared, the Rx DMA operation is stopped after the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process is restarted. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state.</p>
OSF	2	rw	<p>Operate on Second Frame</p> <p>When this bit is set, it instructs the DMA to process the second frame of the Transmit data even before the status for the first frame is obtained.</p>
RTC	[4:3]	rw	<p>Receive Threshold Control</p> <p>These two bits control the threshold level of the MTL Receive FIFO. Transfer (request) to DMA starts when the frame size within the MTL Receive FIFO is larger than the threshold. In addition, full frames with length less than the threshold are transferred automatically. The value of 11 is not applicable if the configured Receive FIFO size is 128 bytes. These bits are valid only when the RSF bit is zero, and are ignored when the RSF bit is set to 1.</p> <ul style="list-style-type: none"> * 00B: 64 * 01B: 32 * 10B: 96 * 11B: 128

Field	Bits	Type	Description
Reserved_5	5	r	Reserved
FUF	6	rw	<p>Forward Undersized Good Frames</p> <p>When set, the Rx FIFO forwards Undersized frames (frames with no Error and length less than 64 bytes) including pad-bytes and CRC.</p> <p>When reset, the Rx FIFO drops all frames of less than 64 bytes, unless a frame is already transferred because of the lower value of Receive Threshold, for example, RTC = 01B.</p>
FEF	7	rw	<p>Forward Error Frames</p> <p>When this bit is reset, the Rx FIFO drops frames with error status (CRC error, collision error, MII_ER, giant frame, watchdog timeout, or overflow). However, if the start byte (write) pointer of a frame is already transferred to the read controller side (in Threshold mode), then the frame is not dropped.</p> <p>In the ETH-MTL configuration in which the Frame Length FIFO is also enabled during core configuration, the Rx FIFO drops the error frames if that frame's start byte is not transferred (output) on the ARI bus.</p> <p>When the FEF bit is set, all frames except runt error frames are forwarded to the DMA. If the RSF bit is set and the Rx FIFO overflows when a partial frame is written, then the frame is dropped irrespective of the FEF bit setting. However, if the RSF bit is reset and the Rx FIFO overflows when a partial frame is written, then a partial frame may be forwarded to the DMA.</p>
Reserved_12_8	[12:8]	r	Reserved

Field	Bits	Type	Description
ST	13	rw	<p>Start or Stop Transmission Command</p> <p>When this bit is set, transmission is placed in the Running state, and the DMA checks the Transmit List at the current position for a frame to be transmitted. Descriptor acquisition is attempted either from the current position in the list, which is the Transmit List Base Address set by Register 4 [Transmit Descriptor List Address Register], or from the position retained when transmission was stopped previously. If the DMA does not own the current descriptor, transmission enters the Suspended state and Bit 2 (Transmit Buffer Unavailable) of Register 5 [STATUS Register] is set. The Start Transmission command is effective only when transmission is stopped. If the command is issued before setting Register 4 [Transmit Descriptor List Address Register], then the DMA behavior is unpredictable.</p> <p>When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and it becomes the current position when transmission is restarted. To change the list address, you need to program Register 4 [Transmit Descriptor List Address Register] with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current frame is complete or the transmission is in the Suspended state.</p>

Field	Bits	Type	Description
TTC	[16:14]	rw	<p>Transmit Threshold Control</p> <p>These bits control the threshold level of the MTL Transmit FIFO. Transmission starts when the frame size within the MTL Transmit FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are used only when Bit 21 (TSF) is reset.</p> <ul style="list-style-type: none"> * 000B: 64 * 001B: 128 * 010B: 192 * 011B: 256 * 100B: 40 * 101B: 32 * 110B: 24 * 111B: 16
Reserved_19_17	[19:17]	r	Reserved
FTF	20	rw	<p>Flush Transmit FIFO</p> <p>When this bit is set, the transmit FIFO controller logic is reset to its default values and thus all data in the Tx FIFO is lost or flushed. This bit is cleared internally when the flushing operation is completed. The Operation Mode register should not be written to until this bit is cleared. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt frame transmission.</p> <p>Note: The flush operation is complete only when the Tx FIFO is emptied of its contents and all the pending Transmit Status of the transmitted frames are accepted by the CPU. To complete this flush operation, the PHY transmit clock is required to be active.</p>
TSF	21	rw	<p>Transmit Store and Forward</p> <p>When this bit is set, transmission starts when a full frame resides in the MTL Transmit FIFO. When this bit is set, the TTC values specified in Bits[16:14] are ignored. This bit should be changed only when the transmission is stopped.</p>
Reserved_23_22	[23:22]	r	Reserved

Field	Bits	Type	Description
DFF	24	rw	Disable Flushing of Received Frames When this bit is set, the Rx DMA does not flush any frames because of the unavailability of receive descriptors or buffers as it does normally when this bit is reset.
RSF	25	rw	Receive Store and Forward When this bit is set, the MTL reads a frame from the Rx FIFO only after the complete frame has been written to it, ignoring the RTC bits. When this bit is reset, the Rx FIFO operates in the cut-through mode, subject to the threshold specified by the RTC bits.
DT	26	rw	Disable Dropping of TCP/IP Checksum Error Frames When this bit is set, the MAC does not drop the frames which only have errors detected by the Receive Checksum Offload engine. Such frames do not have any errors (including FCS error) in the Ethernet frame received by the MAC but have errors only in the encapsulated payload. When this bit is reset, all error frames are dropped if the FEF bit is reset.
Reserved_31_27	[31:27]	r	Reserved

INTERRUPT_ENABLE

The Interrupt Enable register enables the interrupts reported by **ETH0_STATUS** Register. Setting a bit to 1 enables a corresponding interrupt. After a hardware or software reset, all interrupts are disabled.

ETH0_INTERRUPT_ENABLE

Interrupt Enable Register

(101C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved_31_17															NIE
r															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIE	ERE	FBE	Reserved_12_11	ETE	RWE	RSE	RUE	RIE	UNE	OVE	TJE	TUE	TSE	TIE	
rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
TIE	0	rw	Transmit Interrupt Enable When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled.
TSE	1	rw	Transmit Stopped Enable When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmission Stopped Interrupt is enabled. When this bit is reset, the Transmission Stopped Interrupt is disabled.
TUE	2	rw	Transmit Buffer Unavailable Enable When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Transmit Buffer Unavailable Interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable Interrupt is disabled.
TJE	3	rw	Transmit Jabber Timeout Enable When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmit Jabber Timeout Interrupt is enabled. When this bit is reset, the Transmit Jabber Timeout Interrupt is disabled.

Field	Bits	Type	Description
OVE	4	rw	Overflow Interrupt Enable When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Overflow Interrupt is enabled. When this bit is reset, the Overflow Interrupt is disabled.
UNE	5	rw	Underflow Interrupt Enable When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmit Underflow Interrupt is enabled. When this bit is reset, the Underflow Interrupt is disabled.
RIE	6	rw	Receive Interrupt Enable When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled.
RUE	7	rw	Receive Buffer Unavailable Enable When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Buffer Unavailable Interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable Interrupt is disabled.
RSE	8	rw	Receive Stopped Enable When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped Interrupt is disabled.
RWE	9	rw	Receive Watchdog Timeout Enable When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Watchdog Timeout Interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout Interrupt is disabled.
ETE	10	rw	Early Transmit Interrupt Enable When this bit is set with an Abnormal Interrupt Summary Enable (Bit 15), the Early Transmit Interrupt is enabled. When this bit is reset, the Early Transmit Interrupt is disabled.
Reserved_12_11	[12:11]	r	Reserved

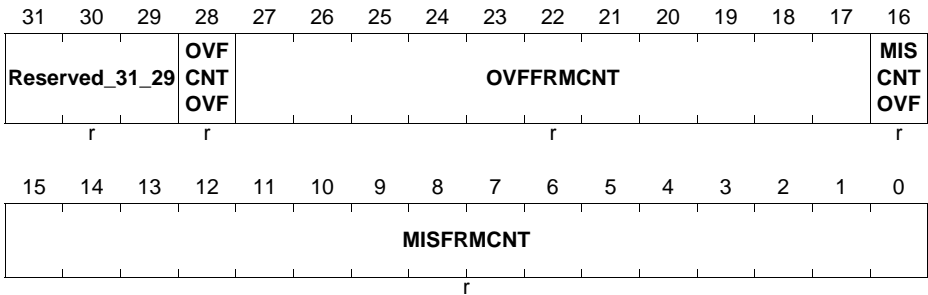
Field	Bits	Type	Description
FBE	13	rw	Fatal Bus Error Enable When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Fatal Bus Error Interrupt is enabled. When this bit is reset, the Fatal Bus Error Enable Interrupt is disabled.
ERE	14	rw	Early Receive Interrupt Enable When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Early Receive Interrupt is enabled. When this bit is reset, the Early Receive Interrupt is disabled.
AIE	15	rw	Abnormal Interrupt Summary Enable When this bit is set, abnormal interrupt summary is enabled. When this bit is reset, the abnormal interrupt summary is disabled. This bit enables the following interrupts in STATUS Register: <ul style="list-style-type: none"> * Transmit Process Stopped * Transmit Jabber Timeout * Receive Overflow * Transmit Underflow * Receive Buffer Unavailable * Receive Process Stopped * Receive Watchdog Timeout * Early Transmit Interrupt * Fatal Bus Error
NIE	16	rw	Normal Interrupt Summary Enable When this bit is set, normal interrupt summary is enabled. When this bit is reset, normal interrupt summary is disabled. This bit enables the following interrupts in Register 5 [STATUS Register]: <ul style="list-style-type: none"> * Transmit Interrupt * Transmit Buffer Unavailable * Receive Interrupt * Early Receive Interrupt
Reserved_31_17	[31:17]	r	Reserved

MISSED_FRAME_AND_BUFFER_OVERFLOW_COUNTER

The DMA maintains two counters to track the number of frames missed during reception. This register reports the current value of the counter. The counter is used for diagnostic purposes. Bits[15:0] indicate missed frames because of the RAM buffer being unavailable. Bits[27:17] indicate missed frames because of buffer overflow conditions (MTL and MAC) and runt frames (good frames of less than 64 bytes) dropped by the MTL.

ETH0_MISSED_FRAME_AND_BUFFER_OVERFLOW_COUNTER

Missed Frame and Buffer Overflow Counter Register (1020_H) **Reset Value: 0000**
0000_H



Field	Bits	Type	Description
MISFRMCNT	[15:0]	r	This field indicates the number of frames missed by the controller because of the RAM Receive Buffer being unavailable. This counter is incremented each time the DMA discards an incoming frame. The counter is cleared when this register is read.
MIS CNT OVF	16	r	Overflow bit for Missed Frame Counter
OVFFRMCNT	[27:17]	r	This field indicates the number of frames missed by the application. The counter is cleared when this register is read.
OVFCNT OVF	28	r	Overflow bit for FIFO Overflow Counter
Reserved_31_29	[31:29]	r	Reserved

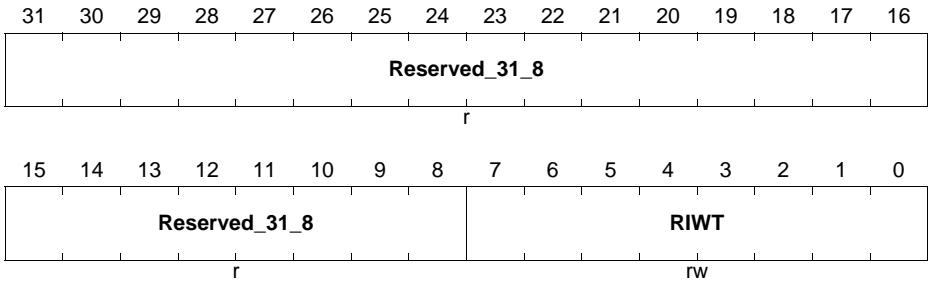
RECEIVE_INTERRUPT_WATCHDOG_TIMER

This register, when written with non-zero value, enables the watchdog timer for the Receive Interrupt (Bit 6) of STATUS Register]

ETH0_RECEIVE_INTERRUPT_WATCHDOG_TIMER

Receive Interrupt Watchdog Timer Register (1024_H)

Reset Value: 0000 0000_H



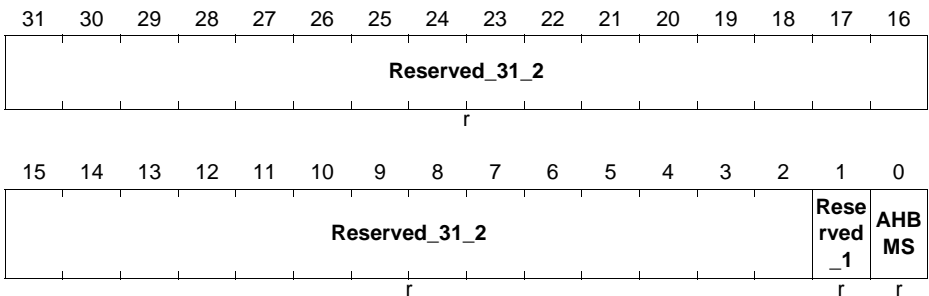
Field	Bits	Type	Description
RIWT	[7:0]	rw	RI Watchdog Timer Count This bit indicates the number of system clock cycles multiplied by 256 for which the watchdog timer is set. The watchdog timer gets triggered with the programmed value after the Rx DMA completes the transfer of a frame for which the RI status bit is not set because of the setting in the corresponding descriptor RDES1[31]. When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per RDES1[31] of any received frame.
Reserved_31_8	[31:8]	r	Reserved

AHB_Status

This register provides the active status of the AHB master interface. This register is useful for debugging purposes. In addition, this register is valid only in the Channel 0 DMA when multiple channels are present in the AV mode.

ETH0_AHB_Status

AHB Status Register (102C_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
AHBMS	0	r	AHB Master Status When high, it indicates that the AHB master interface FSMs are in the non-idle state.
Reserved_1	1	r	Reserved
Reserved_31_2	[31:2]	r	Reserved

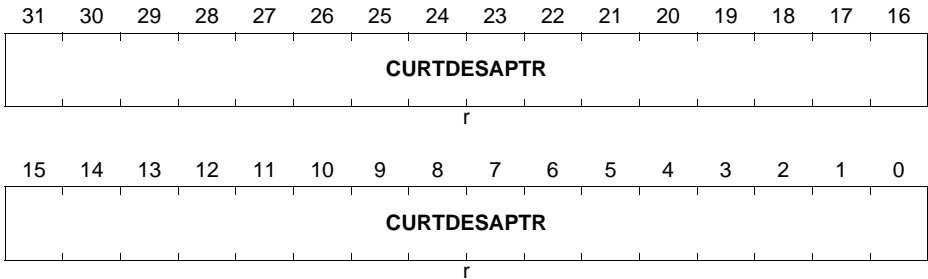
CURRENT_HOST_TRANSMIT_DESCRIPTOR

The Current Host Transmit Descriptor register points to the start address of the current Transmit Descriptor read by the DMA.

ETH0_CURRENT_HOST_TRANSMIT_DESCRIPTOR

Current Host Transmit Descriptor Register (1048_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CURTDES APTR	[31:0]	r	Host Transmit Descriptor Address Pointer Cleared on Reset. Pointer updated by the DMA during operation.

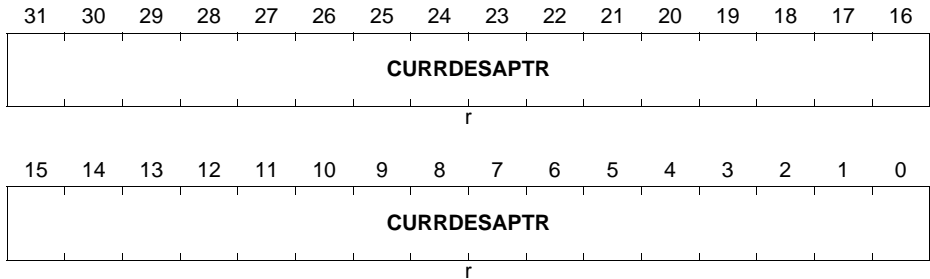
CURRENT_HOST_RECEIVE_DESCRIPTOR

The Current Host Receive Descriptor register points to the start address of the current Receive Descriptor read by the DMA.

ETH0_CURRENT_HOST_RECEIVE_DESCRIPTOR

Current Host Receive Descriptor Register (104C_H)

Reset Value: 0000 0000_H



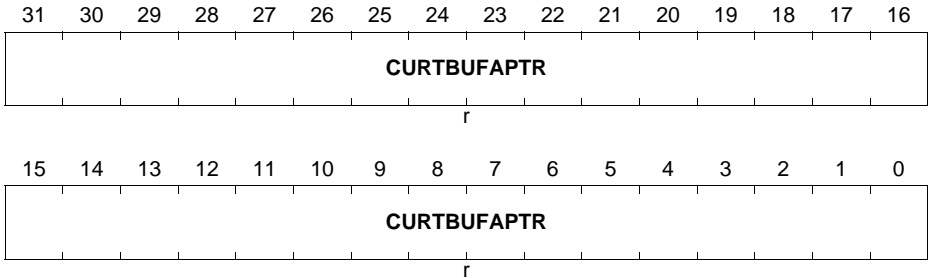
Field	Bits	Type	Description
CURRDES APTR	[31:0]	r	Host Receive Descriptor Address Pointer Cleared on Reset. Pointer updated by the DMA during operation.

CURRENT_HOST_TRANSMIT_BUFFER_ADDRESS

The Current Host Transmit Buffer Address register points to the current Transmit Buffer Address being read by the DMA.

ETH0_CURRENT_HOST_TRANSMIT_BUFFER_ADDRESS

Current Host Transmit Buffer Address Register (1050_H) Reset Value: 0000 0000_H



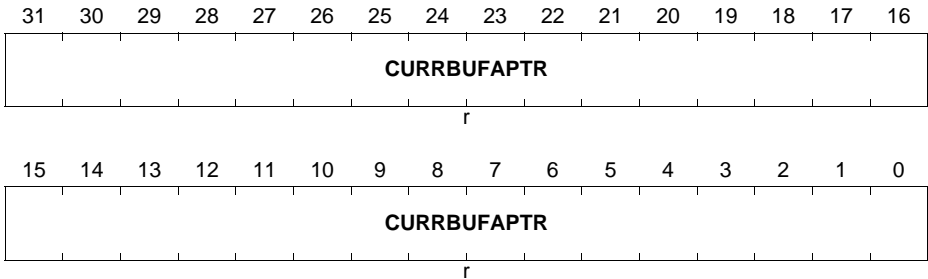
Field	Bits	Type	Description
CURTBUF APTR	[31:0]	r	Host Transmit Buffer Address Pointer Cleared on Reset. Pointer updated by the DMA during operation.

CURRENT_HOST_RECEIVE_BUFFER_ADDRESS

The Current Host Receive Buffer Address register points to the current Receive Buffer address being read by the DMA.

ETH0_CURRENT_HOST_RECEIVE_BUFFER_ADDRESS

Current Host Receive Buffer Address Register (1054_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CURRBUF APTR	[31:0]	r	Host Receive Buffer Address Pointer Cleared on Reset. Pointer updated by the DMA during operation.

HW_FEATURE

This register indicates the presence of the optional features or functions of the DWC_ETH. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks. Note: All bits are set or reset as per the selection of features during the DWC_ETH configuration.

ETH0_HW_FEATURE

HW Feature Register

(1058_H)

Reset Value: 0305 2F35_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved _31	ACTPHYIF			SAV LANI NS	FLE XIPP SEN	INTT SEN	ENH DES SEL	TXCHCNT	RXCHCNT	RXFI FOSI ZE	RXT YP2 COE	RXT YP1 COE	TXC OES EL		
r	r			r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AVS EL	EEE SEL	TSV ER2 SEL	TSV ER1 SEL	MMC SEL	MGK SEL	RWK SEL	SMA SEL	L3L4 FLT REN	PCS SEL	ADD MAC ADR SEL	HAS HSE L	EXT HAS HEN	HDS EL	GMI SEL	MIIS EL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
MIISEL	0	r	10 or 100 Mbps support
GMIISEL	1	r	1000 Mbps support
HDSEL	2	r	Half-Duplex support
EXTHASH EN	3	r	Expanded DA Hash Filter
HASHSEL	4	r	HASH Filter
ADDMAC ADRSEL	5	r	Multiple MAC Address Registers
PCSSEL	6	r	PCS registers (TBI, SGMII, or RTBI PHY interface)
L3L4FLT REN	7	r	Layer 3 and Layer 4 Filter Feature
SMASEL	8	r	SMA (MDIO) Interface
RWKSEL	9	r	PMT Remote Wakeup
MGKSEL	10	r	PMT Magic Packet
MMCSEL	11	r	RMON Module

Field	Bits	Type	Description
TSVER1SEL	12	r	Only IEEE 1588-2002 Timestamp
TSVER2SEL	13	r	IEEE 1588-2008 Advanced Timestamp
EEESEL	14	r	Energy Efficient Ethernet
AVSEL	15	r	AV Feature
TXCOESL	16	r	Checksum Offload in Tx
RXTYP1COE	17	r	IP Checksum Offload (Type 1) in Rx
RXTYP2COE	18	r	IP Checksum Offload (Type 2) in Rx
RXFIFOSIZE	19	rw	Rx FIFO > 2,048 Bytes
RXCHCNT	[21:20]	r	Number of additional Rx channels
TXCHCNT	[23:22]	r	Number of additional Tx channels
ENHDESEL	24	r	Alternate (Enhanced Descriptor)
INTTSEN	25	r	Timestamping with Internal System Time
FLEXIPPSEN	26	r	Flexible Pulse-Per-Second Output
SAVLANS	27	r	Source Address or VLAN Insertion
ACTPHYIF	[30:28]	r	Active or Selected PHY interface When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion * 0000: MII * 0001: RMII * All Others: Reserved
Reserved_31	31	r	Reserved

14.7 Interconnects

The tables that refer to the “global pins” are the ones that contain the inputs/outputs of the ETH.

The GPIO connections are available in the Package Pin Summary inside the Datasheet of the used device.

14.7.1 ETH Pins

The Ethernet MAC supports MII and RMI Interconnect Signals to connect a external PHY. **Figure 14-31** and **Table 14-41** show a typical MII connection to an external PHY.

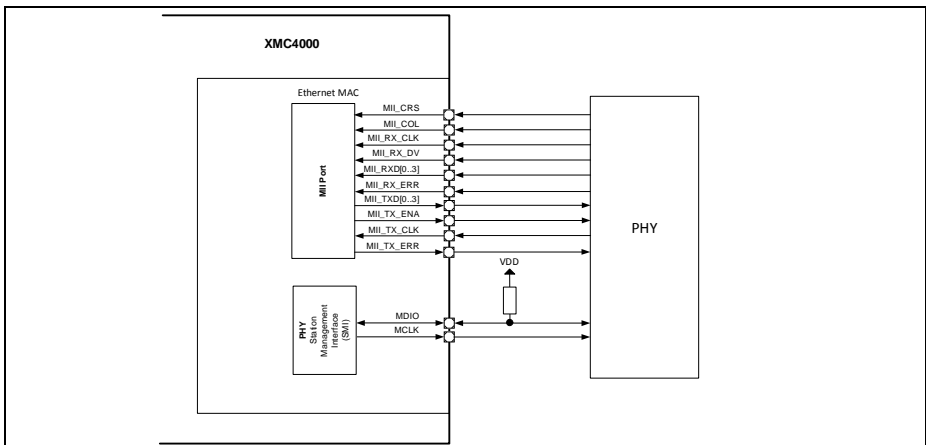


Figure 14-31 Ethernet MII - PHY Interconnect

Table 14-41 ETH Pin Connections for MII

Global Inputs/Outputs	I/O	Connected To	Description
Control Signals			
ETH0.CRS(A)	I	PORT	Carrier Sense
ETH0.CRS(B)	I	PORT	Carrier Sense
ETH0.CRS(C)	I	PORT	Carrier Sense
ETH0.CRS(D)	I	PORT	Carrier Sense
ETH0.COL(A)	I	PORT	Collision Detect
ETH0.COL(B)	I	PORT	Collision Detect
ETH0.COL(C)	I	PORT	Collision Detect

Table 14-41 ETH Pin Connections for MIII (cont'd)

Global Inputs/Outputs	I/O	Connected To	Description
ETH0.COL(D)	I	PORT	Collision Detect
ETH0.RXDV(A)	I	PORT	Receive Data Valid ¹⁾
ETH0.RXDV(B)	I	PORT	Receive Data Valid
ETH0.RXDV(C)	I	PORT	Receive Data Valid
ETH0.RXDV(D)	I	PORT	Receive Data Valid
ETH0.RXER(A)	I	PORT	Receive error
ETH0.RXER(B)	I	PORT	Receive error
ETH0.RXER(C)	I	PORT	Receive error
ETH0.RXER(D)	I	PORT	Receive error
Data Bus			
ETH0.RXD0(A)	I	PORT	Receive data line
ETH0.RXD0(B)	I	PORT	Receive data line
ETH0.RXD0(C)	I	PORT	Receive data line
ETH0.RXD0(D)	I	PORT	Receive data line
ETH0.RXD1(A)	I	PORT	Receive data line
ETH0.RXD1(B)	I	PORT	Receive data line
ETH0.RXD1(C)	I	PORT	Receive data line
ETH0.RXD1(D)	I	PORT	Receive data line
ETH0.RXD2(A)	I	PORT	Receive data line
ETH0.RXD2(B)	I	PORT	Receive data line
ETH0.RXD2(C)	I	PORT	Receive data line
ETH0.RXD2(D)	I	PORT	Receive data line
ETH0.RXD3(A)	I	PORT	Receive data line
ETH0.RXD3(B)	I	PORT	Receive data line
ETH0.RXD3(C)	I	PORT	Receive data line
ETH0.RXD3(D)	I	PORT	Receive data line
ETH0.TXEN	O	PORT	Transmit enable
ETH0.TXER	O	PORT	Transmit error
ETH0.TXD0	O	PORT	Transmit Data Line
ETH0.TXD1	O	PORT	Transmit data line

Table 14-41 ETH Pin Connections for MII (cont'd)

Global Inputs/Outputs	I/O	Connected To	Description
ETH0.TXD2	O	PORT	Transmit data line
ETH0.TXD3	O	PORT	Transmit data line
PHY Clocks			
ETH0.CLKTX(A)	I	PORT	PHY transmit clock
ETH0.CLKTX(B)	I	PORT	PHY transmit clock
ETH0.CLKTX(C)	I	PORT	PHY transmit clock
ETH0.CLKTX(D)	I	PORT	PHY transmit clock
ETH0.CLKRX(A)	I	PORT	PHY receive clock ²⁾
ETH0.CLKRX(B)	I	PORT	PHY receive clock
ETH0.CLKRX(C)	I	PORT	PHY receive clock
ETH0.CLKRX(D)	I	PORT	PHY receive clock

- 1) ETH0.CRS_DV and ETH0.RXDV are physically the same signal in MII and RMII mode. In Register ETH0_CON bitfield CRS_DV should be used to select the input signal of ETH0.CRS_DV.
- 2) ETH0.CLK_RX and ETH0.CLK_RMII are physically the same signal in MII and RMII mode. In Register ETH0_CON bitfield CLK_RMII should be used to select the input signal of ETH0.CLK_RX.

Figure 14-32 and Table 14-42 show a typical RMII connection to an external PHY.

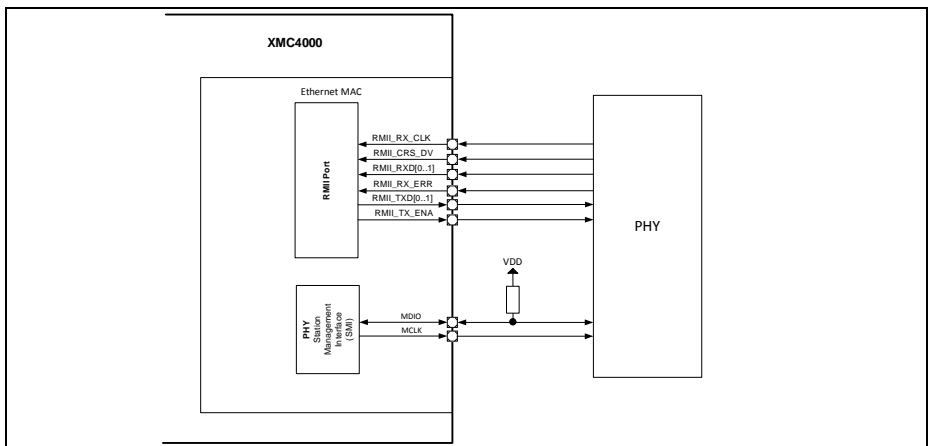


Figure 14-32 Ethernet RMII - PHY Interconnect

Table 14-42 ETH Pin Connections for RMII

Global Inputs/Outputs	I/O	Connected To	Description
Control Signals			
ETH0.CRS_DV(A)	I/O	PORT	Carrier Sense Data Valid
ETH0.CRS_DV(B)	I/O	PORT	Carrier Sense Data Valid
ETH0.CRS_DV(C)	I/O	PORT	Carrier Sense Data Valid
ETH0.CRS_DV(D)	I/O	PORT	Carrier Sense Data Valid
ETH0.RXER(A)	I/O	PORT	Receive error
ETH0.RXER(B)	I/O	PORT	Receive error
ETH0.RXER(C)	I/O	PORT	Receive error
ETH0.RXER(D)	I/O	PORT	Receive error
Data Bus			
ETH0.RXD0(A)	I	PORT	Receive data line
ETH0.RXD0(B)	I	PORT	Receive data line
ETH0.RXD0(C)	I	PORT	Receive data line
ETH0.RXD0(D)	I	PORT	Receive data line
ETH0.RXD1(A)	I	PORT	Receive data line
ETH0.RXD1(B)	I	PORT	Receive data line
ETH0.RXD1(C)	I	PORT	Receive data line
ETH0.RXD1(D)	I	PORT	Receive data line
ETH0.TXEN	O	PORT	Transmit enable
ETH0.TXD0	O	PORT	Transmit data line
ETH0.TXD1	O	PORT	Transmit data line
PHY Clocks			
ETH0.CLK_RMII(A)	I	PORT	PHY Clock
ETH0.CLK_RMII(B)	I	PORT	PHY Clock
ETH0.CLK_RMII(C)	I	PORT	PHY Clock
ETH0.CLK_RMII(D)	I	PORT	PHY Clock

Table 14-43 ETH Pin Connections for MDIO

Global Inputs/Outputs	I/O	Connected To	Description
Clock			
ETH0.MDC	O	PORT	Management Data Clock
Data¹⁾			
ETH0.MDIO(A)	I/O	PORT	Management Data I/O
ETH0.MDIO(B)	I/O	PORT	Management Data I/O
ETH0.MDIO(C)	I/O	PORT	Management Data I/O
ETH0.MDIO(D)	I/O	PORT	Management Data I/O

1) Ethernet MDIO Signals are bidirectional Signals and use Hardware controlled I/Os. Typically a pull-up is required on that pin.

Table 14-44 ETH System Related connections

Global Inputs/Outputs	I/O	Connected To	Description
ETH0.SRC0	O	NVIC	interrupt output line
...

15 EtherCAT Slave Controller (ECAT)

The EtherCAT Slave Controller (ESC) takes care of the EtherCAT communication as an interface between the EtherCAT fieldbus and the slave application. This chapter describes the implementation of the ECAT module into the XMC4300 devices. More information about the EtherCAT working principle can be found under the referenced documents.

References

- [15] EtherCAT_ET1100_Datasheet_all_v1i8.pdf
- [16] EtherCAT_ESC_Datasheet_Sec1_Technology_2i1.pdf
- [17] EtherCAT_IPCore_Altera_V2.4.3_Datasheet_v1i0.pdf

Table 15-1 Abbreviations

ESC	EtherCAT Slave Controller
FMMU	Fieldbus Memory Units
EPU	EtherCAT Processing Unit
ESI	EtherCAT Slave Information, stored in EEPROM
PDI	Process Data Interface (AHB-lite)

15.1 Overview

EtherCAT is an Ethernet-based fieldbus system, invented by Beckhoff Automation and is standardized in IEC 61158. Using standard Ethernet packets or frames (according to IEEE 802.3) the EtherCAT Slave Controller (ECAT) does not received, interpret, and copies the process data at every node. The the EtherCAT Slave Controller read the data addressed to them while the telegram passes through the device.

15.1.1 Features

The ECAT provides the following functionality:

- 2 MII Ports to connect Ethernet PHYs
- 8 FMMUs.
- 8 SyncManagers
- 8 Kbyte of Process Data Memory with Parity
- 64Bit Distributed Clocks

15.1.2 Block Diagram

The ECAT block diagram is shown in [Figure 15-1](#).

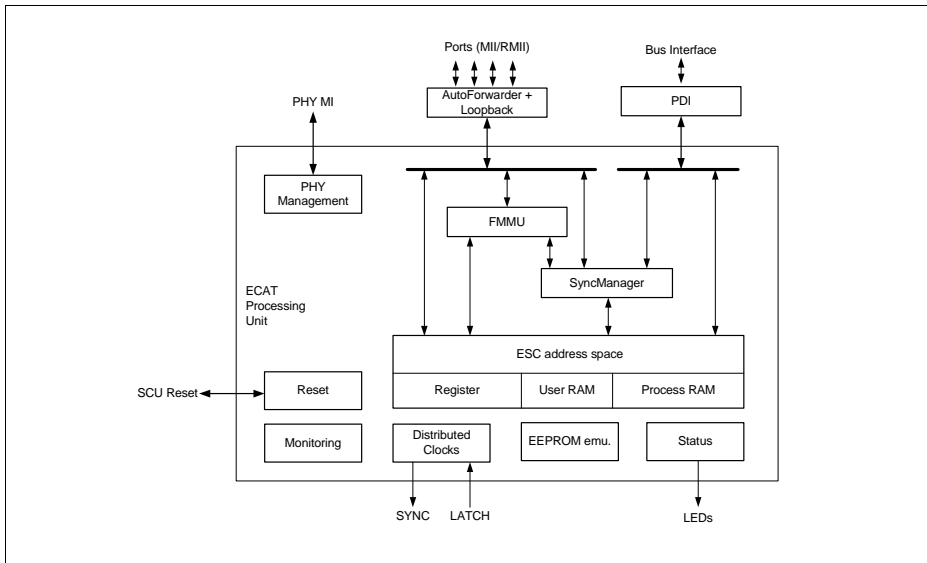


Figure 15-1 EtherCAT Slave Controller Block Diagram

15.1.3 EtherCAT Slave Controller Function Blocks

EtherCAT Processing Unit

The EtherCAT Processing Unit (EPU) receives, analyses and processes the EtherCAT data stream. It is logically located between port 0 and port 1. The main purpose of the EtherCAT Processing unit is to enable and coordinate access to the internal registers and the memory space of the ESC, which can be addressed both from the EtherCAT master and from the local application via the PDI. Data exchange between master and slave application is comparable to a dual-ported memory (process memory), enhanced by special functions e.g. for consistency checking (SyncManager) and data mapping (FMMU). The EtherCAT Processing Units contains the main function blocks of EtherCAT slaves besides Auto-Forwarding, Loop-back function, and PDI.

Memory

The EtherCAT Slave controller has an address space of up to 64 Kbyte. The first block of 4 Kbyte (0x0000 to 0x0FFF) is used for registers and user memory. The memory space from address 0x1000 onwards is used as the process memory. The XMC4300 supports 8 Kbyte of Process Data RAM. .

FMMU

Fieldbus Memory Management Units are used for bitwise mapping of logical addresses to physical addresses of the ESC.

SyncManager

SyncManagers are responsible for consistent data exchange and mailbox communication between EtherCAT master and slaves. The communication direction can be configured for each SyncManager. Read or write transactions may generate events for the EtherCAT master and an attached EtherCAT Slave respectively. The SyncManagers are responsible for the main difference between an ESC and a dual-ported memory, because they map addresses to different buffers and block accesses depending on the SyncManager state. This is also a fundamental reason for bandwidth restrictions of the PDI.

15.1.4 Further Reading on EtherCAT

For further information on EtherCAT, refer to the EtherCAT specification ETG.1000, available from the EtherCAT Technology Group (ETG, <http://www.ethercat.org>), and the IEC standard “Digital data communications for measurement and control – Fieldbus for use in industrial control systems”, IEC 61158 Type 12: EtherCAT, available from the IEC (<http://www.iec.ch>).

15.2 EtherCAT Protocol

EtherCAT uses standard IEEE 802.3 Ethernet frames, thus a standard network controller can be used and no special hardware is required on master side. EtherCAT has a reserved EtherType of 0x88A4 that distinguishes it from other Ethernet frames. Thus, EtherCAT can run in parallel to other Ethernet protocols. EtherCAT does not need the IP protocol, however it can be encapsulated in IP/UDP. The EtherCAT Slave Controller processes the frame in hardware. Thus, communication performance is independent from processor power. An EtherCAT frame is subdivided into the EtherCAT frame header followed by one or more EtherCAT datagrams. At least one EtherCAT datagram has to be in the frame. Only EtherCAT frames with Type 1 in the EtherCAT Header are currently processed by the ESCs. The ESCs also support IEEE802.1Q VLAN Tags, although the VLAN Tag contents are not evaluated by the ESC. If the minimum Ethernet frame size requirement is not fulfilled, padding bytes have to be added. Otherwise the EtherCAT frame is exactly as large as the sum of all EtherCAT datagrams plus EtherCAT frame header.

15.3 Ethernet Physical Layer

EtherCAT slave devices with Ethernet Physical Layer usually support MII interfaces, some do also support the RMII interface. Since RMII PHYs include TX FIFO's, they increase the packet forwarding delay of an EtherCAT slave device as well as the jitter.

EtherCAT Slave Controller (ECAT)

RMII as a Ethernet Physical Layer is not supported by the XMC4300 due to these reasons.

15.3.1 MII Interface

The following signals are used by the ECAT to connect a Ethernet PHY.

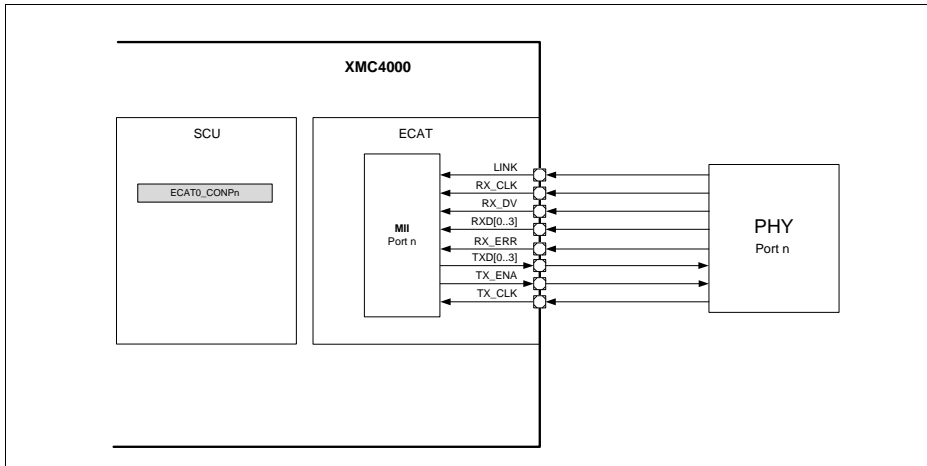


Figure 15-2 MII Interface Signals

The MII signals TX_ERR, COL and CRS are not used by the ESC. TX_CLK is used for automatic TX Shift compensation.

If an ESC MII interface is not used, input pins like LINK_MII has to be tied to the logic value high which indicates no link. RX_CLK, RXD, RX_ER, and especially RX_DV have to be tied to GND. For this purpose you can select a input position in register ECAT0_CONPx of these signals which are not connected to a pin on the XMC4300.

For more details about the MII interface, refer to IEEE Standard 802.3, available from the IEEE (<http://standards.ieee.org/getieee802>).

15.3.1.1 LINK_MII Signal

The LINK_MII signal used for link detection is typically an LED output signal of the Ethernet PHY. If available, LINK_MII should be connected to a combined signal indicating a 100 Mbit/s Full Duplex link. If such a signal is not available, a signal indicating a 100 Mbit/s link (speed LED) might be used. If only a Link signal is available (link LED), this might be used. Never use (combined) activity signals, e.g., Link/Act LED outputs, because the link state will toggle upon activity.

EtherCAT Slave Controller (ECAT)

The main advantage of using a dedicated link signal instead of reading out MII management interface registers is the fast reaction time in case of a link loss. This is crucial for redundancy operation, since only one lost frame is tolerated. The EtherCAT port of an ESC which loses a link has to be closed as fast as possible to maintain EtherCAT communication at the other ports and to reduce the number of lost frames.

The LINK_MII signal state is reflected in the register.

15.3.2 PHY Management Interface

Most EtherCAT slave controllers with MII/RMII/RGMII ports use the MII management interface for communication with the Ethernet PHYs. The MII management interface can be used as well by the EtherCAT master if supported by the ESC. Enhanced MII link detection uses the management interface for configuration and restarting auto negotiation after communication errors occurred (TX PHYs only). Some ESCs can make use of the MII management interface for link detection and PHY configuration. For fast link detection, the ESCs require to use a separate signal (LINK_MII).

15.3.2.1 PHY Management Interface Signals

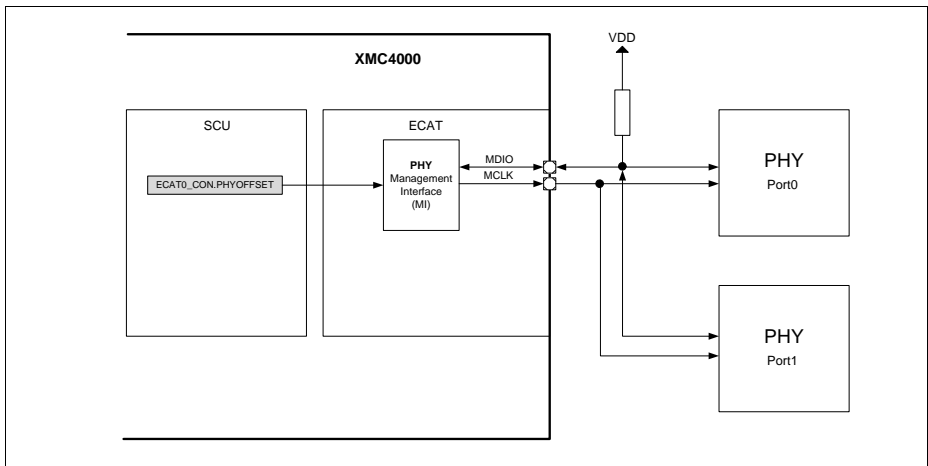


Figure 15-3 PHY management Interface Signals

MDIO must have a pull-up resistor (4.7 kOhm recommended) externally. MCLK is driven rail-to-rail, idle value is High.

15.3.2.2 PHY Address Configuration

The EtherCAT IP Core addresses Ethernet PHYs typically using logical port number plus PHY address offset. Ideally, the Ethernet PHY addresses should correspond with the logical port number, so PHY addresses 0 and 1 are used.

A PHY address offset of 0-31 can be applied which moves the PHY addresses to any consecutive address range. The ECAT module expects logical port 0 to have PHY address 0 plus PHY address offset. The PHY address offset can be selected in register ECAT0_CON.

15.3.2.3 PHY Reset Signal

The PHY reset signal is generated out of the ECAT module. It is required to release the both, PHY and ECAT module synchronous out of reset, this signal should be used. Since there are no pull devices active on the XMC4300 during and after reset, a pull down resistor must be added on this signal on board level. (Figure 15-10)

In some case PHYs may be released from reset after releasing the ECAT module. To generate a delay, the pin for nPHY_RESET can be used as an I/O and shall be switched later to the alternate output function.

15.3.2.4 PHY Clock

Both PHYs connected to the ECAT module and the ECAT itself must share the same clock source. This is achieved by providing the 25MHz clock signal which is used for the ECAT module on a output pin (PHY_CLK25) of the XMC4300. (Figure 15-10)

15.4 FMMU

Fieldbus Memory Management Units (FMMU) convert logical addresses into physical addresses by the means of internal address mapping. Thus, FMMUs allow to use logical addressing for data segments that span several slave devices: one datagram addresses data within several arbitrarily distributed ESCs. Each FMMU channel maps one continuous logical address space to one continuous physical address space of the slave. The FMMUs of Beckhoff ESCs support bit wise mapping, the number of supported FMMUs depends on the ESC. The access type supported by an FMMU is configured to be either read, write, or read/write.

15.5 SyncManager

The memory of an ESC can be used for exchanging data between the EtherCAT master and a local application (on a micro controller attached to the PDI) without any restrictions. Using the memory for communication like this has some draw-backs which are addressed by the SyncManagers inside the ESCs:

EtherCAT Slave Controller (ECAT)

Data consistency is not guaranteed. Semaphores have to be implemented in software for exchanging data in a coordinated way.

Data security is not guaranteed. Security mechanisms have to be implemented in software.

Both EtherCAT master and application have to poll the memory in order to get to know when the access of the other side has finished.

SyncManagers enable consistent and secure data exchange between the EtherCAT master and the local application, and they generate interrupts to inform both sides of changes.

SyncManagers are configured by the EtherCAT master. The communication direction is configurable, as well as the communication mode (Buffered Mode and Mailbox Mode). SyncManagers use a buffer located in the memory area for exchanging data. Access to this buffer is controlled by the hardware of the SyncManagers.

A buffer has to be accessed beginning with the start address, otherwise the access is denied. After accessing the start address, the whole buffer can be accessed, even the start address again, either as a whole or in several strokes. A buffer access finishes by accessing the end address, the buffer state changes afterwards and an interrupt or a watchdog trigger pulse is generated (if configured). The end address cannot be accessed twice inside a frame.

Two communication modes are supported by SyncManagers:

Buffered Mode

The buffered mode allows both sides, EtherCAT master and local application, to access the communication buffer at any time. The consumer gets always the latest consistent buffer which was written by the producer, and the producer can always update the content of the buffer. If the buffer is written faster than it is read out, old data will be dropped.

The buffered mode is typically used for cyclic process data.

Mailbox Mode

The mailbox mode implements a handshake mechanism for data exchange, so that no data will be lost. Each side, EtherCAT master or local application, will get access to the buffer only after the other side has finished its access. At first, the producer writes to the buffer. Then, the buffer is locked for writing until the consumer has read it out. Afterwards, the producer has write access again, while the buffer is locked for the consumer.

The mailbox mode is typically used for application layer protocols.

The SyncManagers accept buffer changes caused by the master only if the FCS of the frame is correct, thus, buffer changes take effect shortly after the end of the frame.

15.6 EtherCAT State Machine

The EtherCAT State machine (ESM) is responsible for the coordination of master and slave applications at start up and during operation. State changes are typically initiated by requests of the master. They are acknowledged by the local application after the associated operations have been executed. Unsolicited state changes of the local application are also possible.

There are four states an EtherCAT slave shall support, plus one optional state:

- Init (state after Reset)
- Pre-Operational
- Safe-Operational
- Operational
- Bootstrap (optional)

The states and the allowed state changes are shown in [Figure 15-4](#).

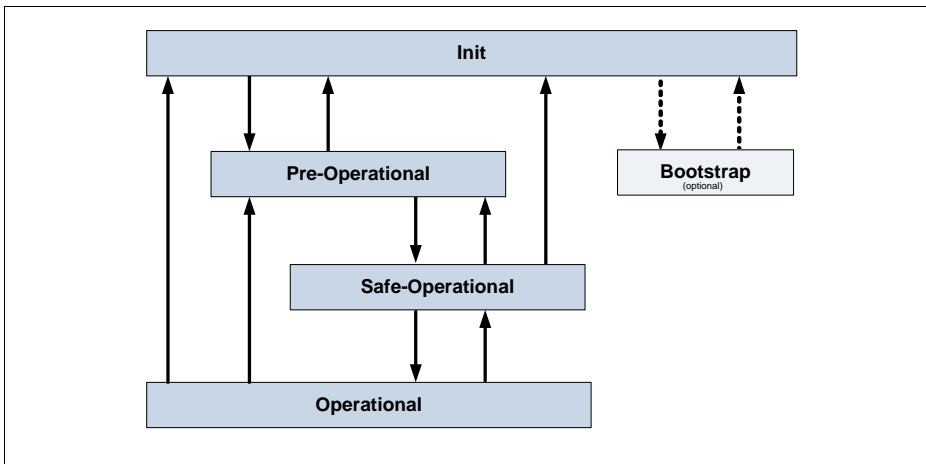


Figure 15-4 EtherCAT State Machine

Not all state changes are possible, e.g., the transition from Init to Operational requires the following sequence: Init -> Pre-Operational -> Safe-Operational -> Operational.

Each state defines required services. Before a state change is confirmed by the slave all services required for the requested state have to be provided or stopped respectively.

15.7 EtherCAT Slave Controller Address Space Overview

An EtherCAT Slave Controller has an maximum address space of 64KByte. The first block of 4KByte (0x0000 to 0x0FFF) is dedicated for registers. The Process Data RAM

EtherCAT Slave Controller (ECAT)

starts at address 0x1000. The size of the Process Data RAM is 8 Kbyte for the XMC4300.

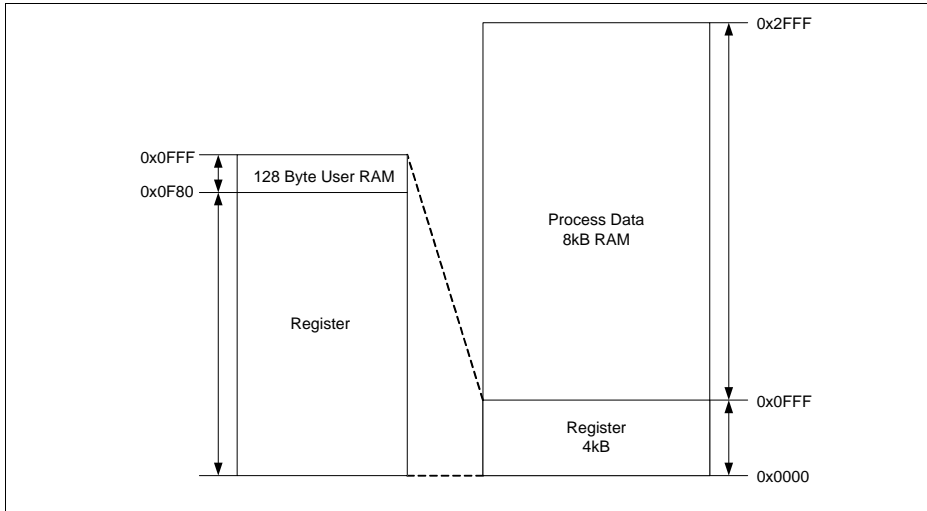


Figure 15-5 EtherCAT Slave Controller Address Space Overview

15.8 ESI EEPROM

EtherCAT slave controllers use a mandatory NVRAM (typically a serial EEPROM with I²C interface) to store device configuration and device description.

The EtherCAT IP Core supports omitting the serial IIC EEPROM if a micro controller with a read/write access to an NVRAM (embedded Flash) can be used to emulate the EEPROM transactions. Since the logical interface is the same in this case, the EEPROM emulation is treated to be equivalent to the typical IIC EEPROM solution. XMC4300 uses EEPROM emulation.

EtherCAT Slave Controller (ECAT)

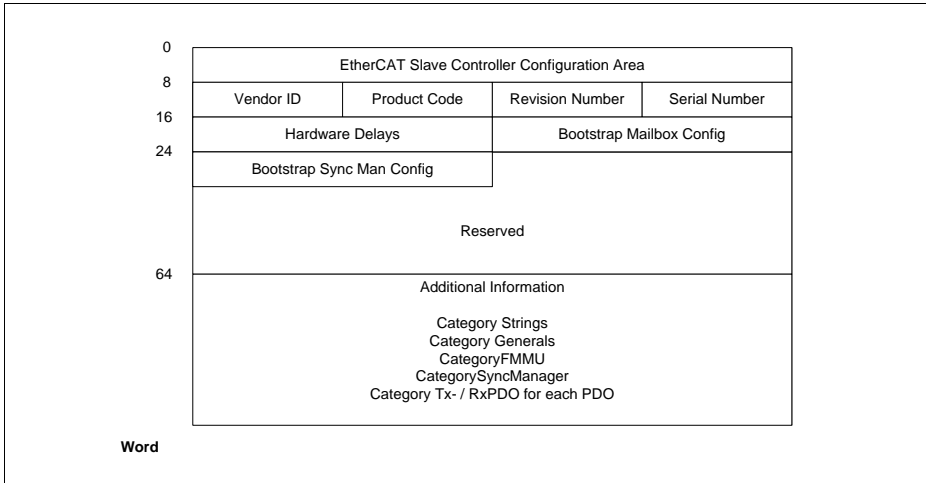


Figure 15-6 ESI EEPROM Layout

At least the information stored in the address range from word 0 to 63 (0x00 to 0x3F) is mandatory.

15.8.1 EEPROM Emulation

The EEPROM emulation mode is used in IP Core based ESCs with a non-volatile memory (NVRAM) attached or integrated to a micro controller. The ESC configuration and the device description can be stored in the Flash of the micro controller, e.g., together with the program or other configuration data. An additional external EEPROM chip for the system is not needed any more if EEPROM emulation is used.

The micro controller emulates the EEPROM interface actions of the ESC and executes all EEPROM reload, read, and write requests. EEPROM write data is stored in the Flash of the micro controller, and EEPROM read data is read from the Flash and presented to the EEPROM interface of the ESC.

From the EtherCAT master's point of view, EEPROM emulation mode is equivalent to an external connected IIC EEPROM. The master issues EEPROM commands and waits until the EEPROM interface is not busy anymore.

In EEPROM emulation mode, the EEPROM interface of the ESC issues an interrupt to the micro controller if an EEPROM command is pending and sets the busy bit. While the busy bit is set, the micro controller can read out the command and the EEPROM address. For a write access, write data is present in the data register. For a read command, read data has to be stored in the data register by the micro controller.

EtherCAT Slave Controller (ECAT)

A reload command requires the micro controller to place the Configured Station Alias and Enhanced Link detection settings in the data register.

Once the micro controller has finished reading/writing the EEPROM data register, it acknowledges the command by writing to the EEPROM command register bits. The micro controller has to write the command value it has executed into the EEPROM command register. Errors can be indicated using two of the error bits. After acknowledging the command, the EEPROM state machine is not busy anymore and the interrupt is released.

Table 15-2 EtherCAT EEPROM Configuration Area

Word Address	Parameter	Description	Register
0x0	PDI Control	Initialization value for PDI Control register	PDI_CONTROL ¹⁾ , ESC_CONFIG
0x1	PDI Configuration	Initialization value for PDI Configuration register	PDI_CONFIG ²⁾ , SYNC_LATCH_CONFIG
0x2	Pulse Length of SYNC Signals	Initialization value for Pulse Length of SYNC Signals register	DC_PULSE_LEN
0x3	Extended PDI Configuration	Initialization value for extended PDI Configuration register, shall be set to all zero (0x0000)	PDI_EXT_CONFIG
0x4	Configured Station Alias	Initialization value for Configured Station Alias Address register	STATION_ALIAS
0x5	Reserved	Reserved, shall be zero	-
0x6	Reserved	Reserved, shall be zero	-
0x7	Checksum	Low byte contains remainder of division of word 0 to word 6 as unsigned number divided by the polynomial x^8+x^2+x+1 (initial value 0xFF). <i>Note: For debugging purposes it is possible to disable the checksum validation with a checksum value of 0x88A4. Never use this for production!</i>	-

1) Register value for PDI_CONTROL must be set to 0x80. All other values are reserved and shall not be used.

2) Register value for PDI_CONFIG must be 0x81. All other values are reserved and shall not be used.

EtherCAT Slave Controller (ECAT)

Note: The EEPROM can be assigned to the PDI even if EEPROM Emulation is used. EEPROM_SIZE has to be 0 for EEPROM emulation (EEPROM emulation with EEPROM_SIZE=1 is for testing only; all commands are acknowledged automatically).

Note: Reserved words or reserved bits of the ESC Configuration Area should be filled with 0.

15.9 Sync/Latch Signals

ESCs with Distributed Clocks support generation of Sync-Signals and time stamping of Latch-Signals.

The SyncSignals can be used internally for:

- Interrupt generation (mapping to AL Event Request register 0x0220:0x0223 and PDI IRQ to the XMC4300 NVIC)
- PDI Digital Output Update events
- PDI Digital Input Latch events

The Latch Event unit supports time stamping of up to two Latch-Signals (LATCH edge separately), and time stamping of SyncManager events for debugging purposes.

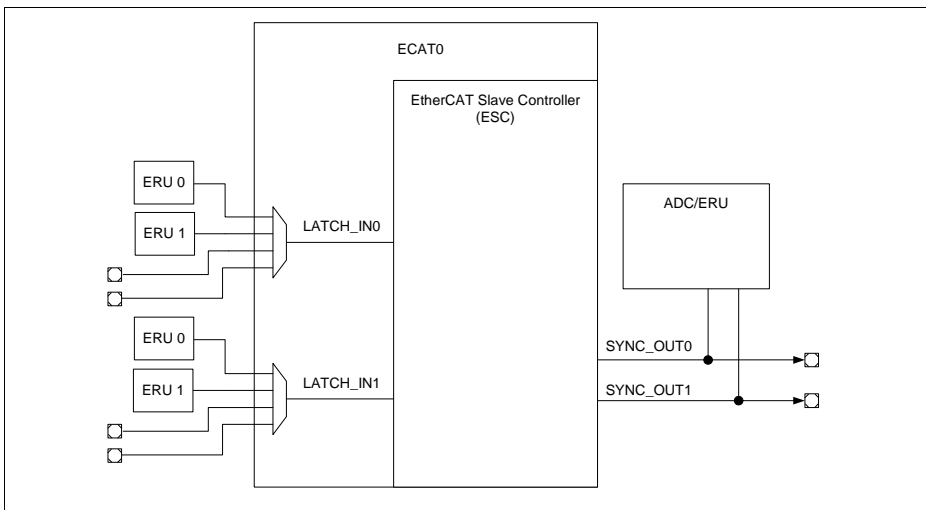


Figure 15-7 EtherCAT Slave Controller Sync/Latch Signals

Note: For LATCH signals derived from pins should not exceed ECAT CLK100/2 = 50MHz. Peripheral trigger signals derived from ERU1 need to be gated by the ERU to meet this requirements.

EtherCAT Slave Controller (ECAT)

Note: The length of SYNC pulses can be adjusted via EEPROM to meet the system requirements of a minimum plus length of $> f_{sys}/2$ (ERU) and $f_{ADC}/2$ (Peripherals 0 clock).

15.10 LED Signals (Indicators)

EtherCAT slave controllers support different LED's regarding link state and AL status. The XMC4300 support a LED_RUN, LED_ERR, LED_STAT_RUN and LED_LINK_ACT for each port.

15.10.1 RUN LED

The AL status is displayed with the RUN LED (green). The RUN output of an ESC is controlled by the AL status register [Page 15-47](#) and supports the following states, which are automatically translated into blink codes:

Table 15-3 RUN LED States

ERR LED	Description
OFF	The device is in state INIT
Blinking (slow)	The device is in state PRE-OPERATIONAL
Single Flash	The device is in state SAFE-OPERATIONAL
On	The device is in state OPERATIONAL
Flickering (fast)	The device is in state BOOTSTRAP or loading the EEPROM

The XMC4300 support optional RUN LED outputs by overriding the state indication of the RUN LED. The output can be set by master or local application. This can be used e.g. for locating a specific slave by forcing the RUN LED to indicate a triple flash (Device Identification).

15.10.2 ERR LED

The ERR LED indicates local errors and application errors. Some errors are automatically indicated by the ECAT module, other error states are detected by the XMC4300, and indicated by writing to the ERR LED Override register [Page 15-50](#).

The following ERR LED states can be automatically generated by an ECAT, without interaction of the XMC4300.

Table 15-4 ERR LED States

ERR LED	Description	Examples
OFF	No Error	
Flickering (fast)	EEPROM loading error	not loaded (default after reset deassert of ECAT module) or CRC error
Single Flash	AL Status register Error Indication bit 4 is set while device emulation is disabled	XMC4300 sets Error Indication bit.
Double Flash	Process Data Watchdog time-out (edge detected) while device is OPERATIONAL	master is disconnected/not sending process data any more
On	PDI Watchdog time-out (edge detected) or Build-In self-test error	XMC4300 is not accessing the ECAT module (example: CPU is halted by the debugger)

15.10.3 STATE RUN LED

The STATE LED is used to drive a bicolor-LED combining RUN and ERR LED. Since the RUN LED part of the STATE LED must be turned off while the ERR LED part is active, the RUN and ERR LED signals cannot be simply combined to drive the bicolor LED. XMC4300 support a STATE_RUN signal, which is turned off while ERR LED is on, so STATE_RUN and ERR signals can be used to drive the bicolor STATE LED.

15.10.4 LINK ACT LED

The Link/Activity state of each port is displayed with the LINK_ACT LED. It is recommended to use a green LED for that purpose. It is recommended to use the LINK_ACT LED signals of the ECAT module instead of the Link/Activity LED signals of the PHY, because the ECAT signals reflect the actual link/activity state of the device, not only the state of the PHYs.

Table 15-5 LINK_ACT LED States

LINK_ACT LED	Description
OFF	No Link
Blinking	Link and activity
ON	Link without activity

15.11 Service Request Generation

ESCs support two types of interrupts. AL Event Requests dedicated for the XMC4300, and ECAT event requests dedicated for the EtherCAT master.

AL Event Requests can be signaled to the NVIC using the PDI Interrupt Request signal (SR0). For Interrupt generation, the AL Event Request register is combined with the AL Event Mask register using a logical AND operation, then all resulting bits are combined (logical OR) into one interrupt signal. The AL Event Mask register allows for selecting the interrupts which are relevant for the system and handled by the application.

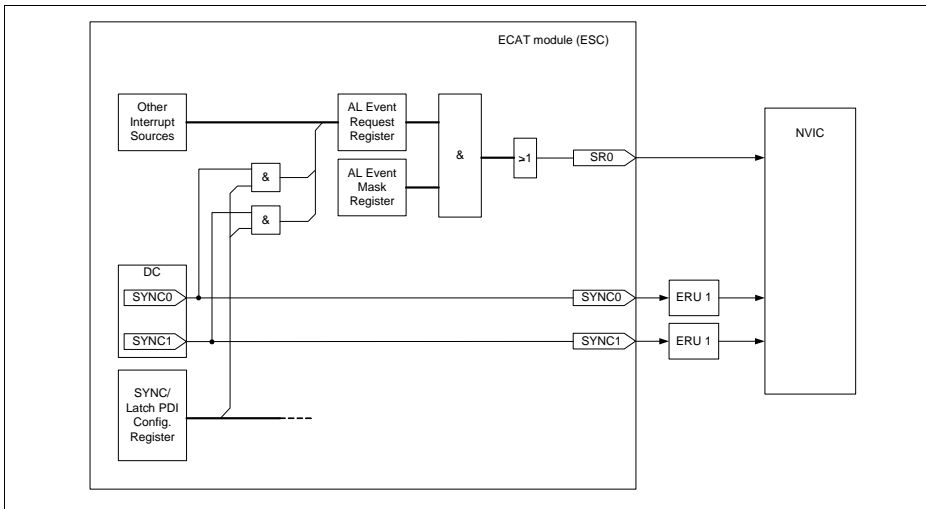


Figure 15-8 EtherCAT Interrupt Masking and Interrupt Signals

The DC SyncSignals can be used for interrupt generation in two ways.

The DC SYNC signals are mapped into the AL Event Request Register (configured with SYNC/LATCH PDI Configuration register 0x0151.3/7). In this case, all interrupts from the ESC to the NVIC are combined into one IRQ signal, and the Distributed Clocks LATCH0/1 inputs can still be used.

The DC SyncSignals are connected via ERU to NVIC interrupt inputs. The System can react on DC Sync-Signal interrupts faster (without reading AL Request register). To route the signal through ERU the customer software must setup the ERU module.

15.12 Debug Behavior

The ECAT module is not supporting any special mode when the CPU enters HALT mode.

15.13 Power, Reset and Clock

The following sections describe the operating conditions, characteristics and timing requirements for the ECAT. All the timing information is related to the module clock, f_{ECAT} .

15.13.1 Power

The ECAT module is inside the power core domain, therefore no special considerations about power up or power down sequences need to be taken. For a explanation about the different power domains, please address the SCU (System Control Unit) chapter.

15.13.2 Module Reset

The ECAT module has one reset source. This reset source is handled at system level and it can be generated independently via a system control register, PRSET2 (address SCU chapter for a full description). After setting the bit PRCLR2.ECATORS it's recommended to poll the bit PRSTAT2.ECATORS before accessing the ECAT module first time. The sequence to release the module out of reset is explained in [Chapter 15.14](#).

15.13.3 EtherCAT Reset

An EtherCAT master is capable to request a reset of an EtherCAT slave. Therefore a reset signal is connected to the XMC4300 SCU. It can be selected in if a TRAP should be requested or a system reset generated. For more information about global parity handling please refer to the SCU chapter EtherCAT Reset.

15.13.4 Module Clock

The module clock of the ECAT module is described in the SCU chapter as f_{ECAT} . The clock for the PHY's is generated out of the ECAT module shown as f_{ECATPHY} and available on GPIO's as signal PHY_CLK25.

It's recommended to supervise the clock generation in the SCU. For this in case of clock fails or PLL loss of lock a TRAP should be generated to signalize the application software the clock fail. After Power-on reset the module clock is gated and need to be de-asserted. Once done and starting up/running the ECAT the clock should not be gated again without set the module reset before since this would breakdown the EtherCAT bus system.

The bus interface clock of the ECAT module is described in the SCU chapter as f_{PERIPH} .

15.14 Initialization and System Dependencies

After Power up the ECAT module is in reset state. The following sequence shown in [Figure 15-9](#) should be executed.

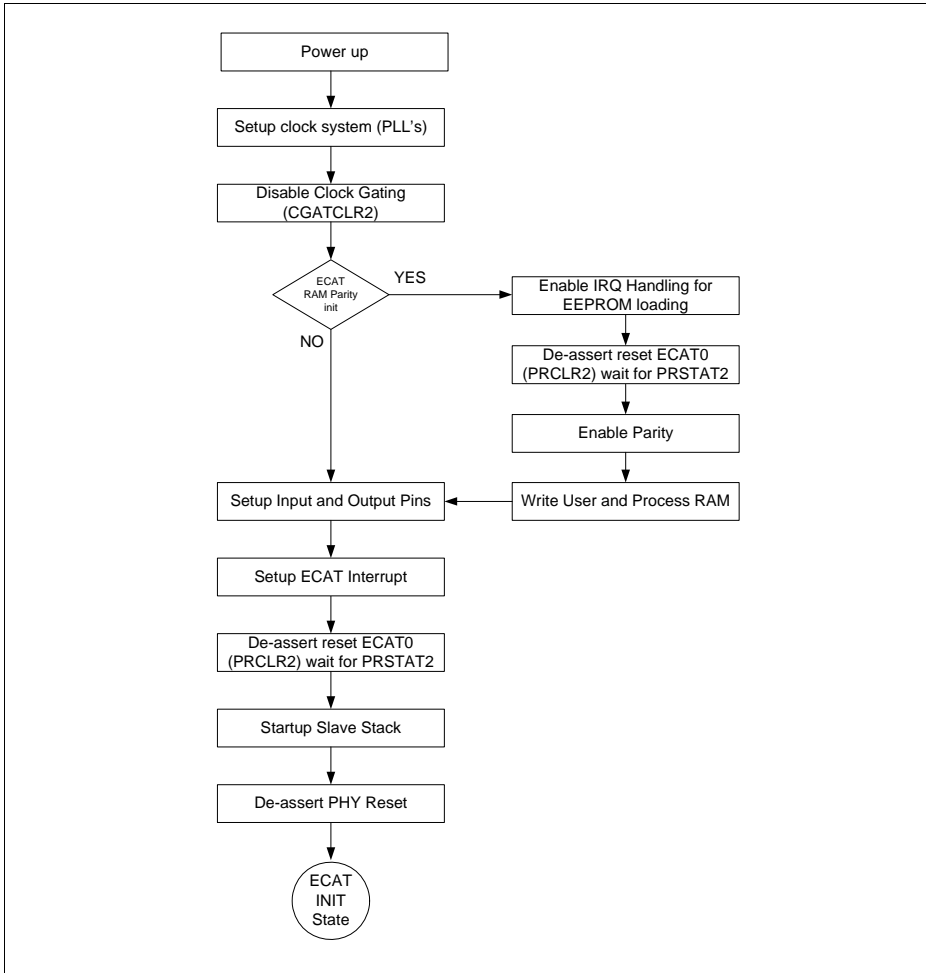


Figure 15-9 EtherCAT Module Initialization

15.14.1 Process and User RAM Parity handling

The process and user RAM provide protection of data integrity via parity checking. A read operation implies checking of the previous stored parity information.

EtherCAT Slave Controller (ECAT)

An occurrence of a parity error is forwarded to the SCU and reflected in the ECAT register ECAT0_STATUS [Page 15-135](#). At the same time a PDI Error is activated. In this state the process data RAM returns 00_H for PDI and Master accesses (EEP_CONT_STAT.L_STAT). By resetting the parity error in SCU the modules returns in normal operating mode, all flags in the ECAT module a cleared automatic.

For more information about global parity handling please refer to the SCU chapter Memory Parity Protection.

15.15 Registers

Registers Overview

the absolute register address is calculated by adding:

module base address + offset address

More information of the address space is shown in [Chapter 15.7](#).

An EtherCAT slave controller (ESC) has an address space of 64 kBytes. The first block of 4 kBytes (0000_H:0FFF_H) is dedicated for registers. The process data RAM starts at address 1000_H, its size depends on the ESC implementation. Availability of registers depends on the ESC as well.

Note: Address areas not listed in the [Table 15-7](#) are reserved. They are not writable. Read data from reserved addresses has to be ignored. Reserved addresses must not be written.

Table 15-6 Registers Address Space

Module	Base Address	End Address	Note
ECAT0	5401 0000 _H	5401 FFFF _H	

Table 15-7 Register Overview

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
ECAT Kernel Registers					
TYPE	Type	0000 _H	U, PV	U, PV	Page 15-26
REVISION	Revision	0001 _H	U, PV	U, PV	Page 15-26
BUILD	Build	0002 _H	U, PV	U, PV	Page 15-27
FMMU_NUM	FMMUs supported	0004 _H	U, PV	U, PV	Page 15-27
SYNC_MANAGER	SyncManagers supported	0005 _H	U, PV	U, PV	Page 15-28
RAM_SIZE	RAM Size	0006 _H	U, PV	U, PV	Page 15-28
PORT_DESC	Port Descriptor	0007 _H	U, PV	U, PV	Page 15-29
FEATURE	ESC Features supported	0008 _H	U, PV	U, PV	Page 15-30
Station Address					
STATION_ADR	Configured Station Address	0010 _H	U, PV	U, PV	Page 15-32

EtherCAT Slave Controller (ECAT)

Table 15-7 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
STATION_ALIAS	Configured Station Alias	0012 _H	U, PV	U, PV	Page 15-32

Write Protection

WR_REG_ENABLE	Write Register Enable	0020 _H	U, PV	U, PV	Page 15-33
WR_REG_PROTECT	Write Register Protection	0021 _H	U, PV	U, PV	Page 15-34
ESC_WR_ENABLE	ESC Write Enable	0030 _H	U, PV	U, PV	Page 15-35
ESC_WR_PROTECT	ESC Write Protection	0031 _H	U, PV	U, PV	Page 15-36

Data Link Layer

ESC_RESET_ECAT	ESC Reset ECAT	0040 _H	U, PV	U, PV	Page 15-37
ESC_RESET_PDI	ESC Reset PDI	0041 _H	U, PV	U, PV	Page 15-38
ESC_DL_CONTROL	ESC DL Control	0100 _H	U, PV	U, PV	Page 15-39
PHYSICAL_RW_OFF SET	Physical Read/Write Offset	0108 _H	U, PV	U, PV	Page 15-41
ESC_DL_STATUS	ESC DL Status	0110 _H	U, PV	U, PV	Page 15-43

Application Layer

AL_CONTROL	AL Control	0120 _H	U, PV	U, PV	Page 15-46
AL_STATUS	AL Status	0130 _H	U, PV	U, PV	Page 15-47
AL_STATUS_CODE	AL Status Code	0134 _H	U, PV	U, PV	Page 15-48
RUN_LED	RUN LED Override	0138 _H	U, PV	U, PV	Page 15-49
ERR_LED	ERR LED Override	0139 _H	U, PV	U, PV	Page 15-50

PDI / ESC Configuration

PDI_CONTROL	PDI Control	0140 _H	U, PV	U, PV	Page 15-51
ESC_CONFIG	ESC Configuration	0141 _H	U, PV	U, PV	Page 15-52
PDI_CONFIG	PDI Configuration	0150 _H	U, PV	U, PV	Page 15-53
SYNC_LATCH_CONFIG	SYNC/LATCH[1:0] PDI Configuration	0151 _H	U, PV	U, PV	Page 15-53
PDI_EXT_CONFIG	Extended PDI Configuration	0152 _H	U, PV	U, PV	Page 15-55

Interrupts

EtherCAT Slave Controller (ECAT)
Table 15-7 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
ECAT_EVENT_MASK	ECAT Event Mask	0200 _H	U, PV	U, PV	Page 15-56
AL_EVENT_MASK	PDI AL Event Mask	0204 _H	U, PV	U, PV	Page 15-58
ECAT_EVENT_REQ	ECAT Event Request	0210 _H	U, PV	U, PV	Page 15-62
AL_EVENT_REQ	AL Event Request	0220 _H	U, PV	U, PV	Page 15-64
Error Counters					
RX_ERR_COUNT _p	Rx Error Counter[3:0]	0300 _H	U, PV	U, PV	Page 15-68
FWD_RX_ERR_COUNTER _p	Forwarded Rx Error counter[3:0]	0308 _H	U, PV	U, PV	Page 15-69
ECAT_PROC_ERR_COUNT	ECAT Processing Unit Error Counter	030C _H	U, PV	U, PV	Page 15-69
PDI_ERR_COUNT	PDI Error Counter	030D _H	U, PV	U, PV	Page 15-70
LOST_LINK_COUNT _p	Lost Link Counter[3:0]	0310 _H	U, PV	U, PV	Page 15-70
Watchdogs					
WD_DIVIDE	Watchdog Divider	0400 _H	U, PV	U, PV	Page 15-71
WD_TIME_PDI	Watchdog Time PDI	0410 _H	U, PV	U, PV	Page 15-71
WD_TIME_PDATA	Watchdog Time Process Data	0420 _H	U, PV	U, PV	Page 15-72
WD_STAT_PDATA	Watchdog Status Process Data	0440 _H	U, PV	U, PV	Page 15-73
WD_COUNT_PDATA	Watchdog Counter Process Data	0442 _H	U, PV	U, PV	Page 15-74
WD_COUNT_PDI	Watchdog Counter PDI	0443 _H	U, PV	U, PV	Page 15-74
SII EEPROM Interface					
EED_CONF	EEPROM Configuration	0500 _H	U, PV	U, PV	Page 15-75
EED_STATE	EEPROM PDI Access State	0501 _H	U, PV	U, PV	Page 15-76
EED_CONT_STAT	EEPROM Control/Status	0502 _H	U, PV	U, PV	Page 15-77
EED_ADR	EEPROM Address	0504 _H	U, PV	U, PV	Page 15-79

EtherCAT Slave Controller (ECAT)

Table 15-7 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
EEP_DATA	EEPROM Data	0508 _H	U, PV	U, PV	Page 15-79
MII Management Interface					
MII_CONT_STAT	MII Management Control/Status	0510 _H	U, PV	U, PV	Page 15-81
MII_PHY_ADR	PHY Address	0512 _H	U, PV	U, PV	Page 15-83
MII_PHY_REG_ADR	PHY Register Address	0513 _H	U, PV	U, PV	Page 15-83
MII_PHY_DATA	PHY Data	0514 _H	U, PV	U, PV	Page 15-85
MII_ECAC_ACS_STATE	MII Management ECAT Access State	0516 _H	U, PV	U, PV	Page 15-86
MII_PDI_ACS_STATE	MII Management PDI Access State	0517 _H	U, PV	U, PV	Page 15-87
FMMU [15:0]		0600 _H	U, PV	U, PV	
FMMU_L_START_ADRy	Logical Start Address	+0x0	U, PV	U, PV	Page 15-89
FMMU_LENy	Length	+0x4	U, PV	U, PV	Page 15-90
FMMU_L_START_BITy	Logical Start bit	+0x6	U, PV	U, PV	Page 15-90
FMMU_L_STOP_BITy	Logical Stop bit	+0x7	U, PV	U, PV	Page 15-91
FMMU_P_START_ADRy	Physical Start Address	+0x8	U, PV	U, PV	Page 15-91
FMMU_P_START_BITy	Physical Start bit	+0xA	U, PV	U, PV	Page 15-92
FMMU_TYPEy	Type	+0xB	U, PV	U, PV	Page 15-93
FMMU_ACTy	Activate	+0xC	U, PV	U, PV	Page 15-94
SyncManager [15:0]		0800 _H	U, PV	U, PV	Page 15-95
SM_P_START_ADRx	Physical Start Address	+0x0	U, PV	U, PV	Page 15-95
SM_LENx	Length	+0x2	U, PV	U, PV	Page 15-96
SM_CONTROLx	Control Register	+0x4	U, PV	U, PV	Page 15-97
SM_STATUSx	Status Register	+0x5	U, PV	U, PV	Page 15-98
SM_ACTx	Activate	+0x6	U, PV	U, PV	Page 15-100

EtherCAT Slave Controller (ECAT)

Table 15-7 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
SM_PDI_CTRLx	PDI Control	+0x7	U, PV	U, PV	Page 15-10 1

DC – Receive Times

DC_RCV_TIME_POR T0	Receive Time Port 0	0900 _H	U, PV	U, PV	Page 15-10 2
DC_RCV_TIME_POR T1	Receive Time Port 1	0904 _H	U, PV	U, PV	Page 15-10 3

DC – Time Loop Control Unit

DC_SYS_TIME	System Time	0910 _H	U, PV	U, PV	Page 15-10 4
DC_RCV_TIME_UNIT	Receive Time ECAT Processing Unit	0918 _H	U, PV	U, PV	Page 15-10 4
DC_SYS_TIME_OFFSET	System Time Offset	0920 _H	U, PV	U, PV	Page 15-10 6
DC_SYS_TIME_DELAY	System Time Delay	0928 _H	U, PV	U, PV	Page 15-10 8
DC_SYS_TIME_DIFF	System Time Difference	092C _H	U, PV	U, PV	Page 15-10 8
DC_SPEED_COUNT_START	Speed Counter Start	0930 _H	U, PV	U, PV	Page 15-11 0
DC_SPEED_COUNT_DIFF	Speed Counter Diff	0932 _H	U, PV	U, PV	Page 15-11 1
DC_SYS_TIME_FILTER_DEPTH	System Time Difference Filter Depth	0934 _H	U, PV	U, PV	Page 15-11 2
DC_SPEED_COUNT_FILTER_DEPTH	Speed Counter Filter Depth	0935 _H	U, PV	U, PV	Page 15-11 3
	Receive Time Latch Mode	0936 _H	U, PV	U, PV	

DC – Cyclic Unit Control

DC_CYC_CONT	Cyclic Unit Control	0980 _H	U, PV	U, PV	Page 15-11 4
-------------	---------------------	-------------------	-------	-------	------------------------------

DC – SYNC Out Unit

EtherCAT Slave Controller (ECAT)

Table 15-7 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
DC_ACT	Activation	0981 _H	U, PV	U, PV	Page 15-11 5
DC_PULSE_LEN	Pulse Length of SyncSignals	0982 _H	U, PV	U, PV	Page 15-11 6
DC_ACT_STAT	Activation Status	0984 _H	U, PV	U, PV	Page 15-11 7
DC_SYNC0_STAT	SYNC0 Status	098E _H	U, PV	U, PV	Page 15-11 8
DC_SYNC1_STAT	SYNC1 Status	098F _H	U, PV	U, PV	Page 15-11 9
DC_CYC_START_TIME	Start Time Cyclic Operation/Next SYNC0 Pulse	0990 _H	U, PV	U, PV	Page 15-11 9
DC_NEXT_SYNC1_PULSE	Next SYNC1 Pulse	0998 _H	U, PV	U, PV	Page 15-12 0
DC_SYNC0_CYC_TIME	SYNC0 Cycle Time	09A0 _H	U, PV	U, PV	Page 15-12 1
DC_SYNC1_CYC_TIME	SYNC1 Cycle Time	09A4 _H	U, PV	U, PV	Page 15-12 2
DC – Latch In Unit					
DC_LATCH0_CONT	Latch0 Control	09A8 _H	U, PV	U, PV	Page 15-12 3
DC_LATCH1_CONT	Latch1 Control	09A9 _H	U, PV	U, PV	Page 15-12 4
DC_LATCH0_STAT	Latch0 Status	09AE _H	U, PV	U, PV	Page 15-12 5
DC_LATCH1_STAT	Latch1 Status	09AF _H	U, PV	U, PV	Page 15-12 6
DC_LATCH0_TIME_POS	Latch0 Time Positive Edge	09B0 _H	U, PV	U, PV	Page 15-12 6
DC_LATCH0_TIME_NEG	Latch0 Time Negative Edge	09B8 _H	U, PV	U, PV	Page 15-12 7

EtherCAT Slave Controller (ECAT)

Table 15-7 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
DC_LATCH1_TIME_POS	Latch1 Time Positive Edge	09C0 _H	U, PV	U, PV	Page 15-12 8
DC_LATCH1_TIME_NEG	Latch1 Time Negative Edge	09C8 _H	U, PV	U, PV	Page 15-12 9
DC – SyncManager Event Times					
DC_ECAC_CNG_EV_TIME	EtherCAT Buffer Change Event Time	09F0 _H	U, PV	U, PV	Page 15-13 1
DC_PDI_START_EV_TIME	PDI Buffer Start Event Time	09F8 _H	U, PV	U, PV	Page 15-13 1
DC_PDI_CNG_EV_TIME	PDI Buffer Change Event Time	09FC _H	U, PV	U, PV	Page 15-13 3
ESC specific					
ID	Module ID	0E00 _H	U, PV	U, PV	Page 15-13 4
ECAT0_STATUS	ECAT0 Status	0E08 _H	U, PV	U, PV	Page 15-13 5
User RAM/Extended ESC features					
USER_RAM	User RAM	0F80 _H	U, PV	U, PV	

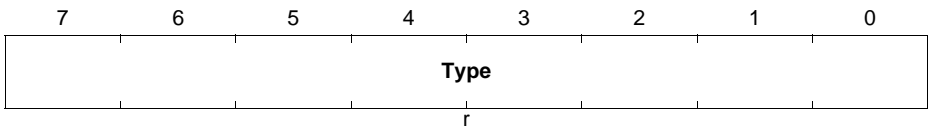
15.15.1 ECAT kernel registers

Type register of ethercat controller (TYPE)

The TYPE register shows the implementation type of the ECAT module. This number is defined by Beckhoff.

ECAT0_TYPE

Type of EtherCAT Controller (0000_H) Reset Value: 98_H



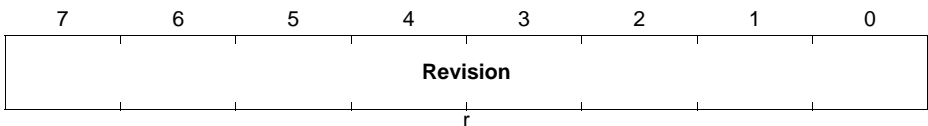
Field	Bits	Type	Description
Type	[7:0]	r	Type of EtherCAT controller

Revision of EtherCAT controller (REVISION)

The revision register shows the revision of the ECAT module.

ECAT0_REVISION

Revision of EtherCAT Controller (0001_H) Reset Value: 01_H



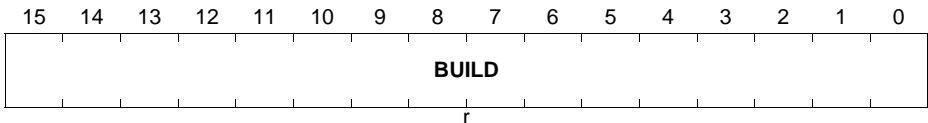
Field	Bits	Type	Description
Revision	[7:0]	r	Revision of EtherCAT controller major version X

Build register (BUILD)

The build register shows the actual build version of the ECAT module.

ECAT0_BUILD

Build Version (0002_H) **Reset Value: 0001_H**



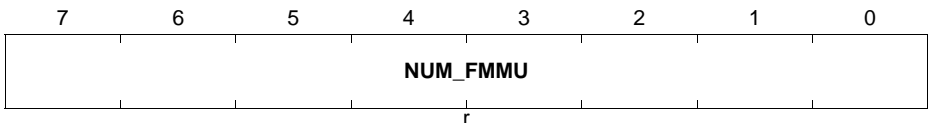
Field	Bits	Type	Description
BUILD	[15:0]	r	Actual build of EtherCAT controller

FMMUs supported (FMMU_NUM)

The register FMMU_NUM shows the number of supported FMMU channels.

ECAT0_FMMU_NUM

FMMUs Supported (0004_H) **Reset Value: 08_H**



Field	Bits	Type	Description
NUM_FMMU	[7:0]	r	Number of supported FMMU channels (or entities) of the EtherCAT Slave Controller

EtherCAT Slave Controller (ECAT)

SyncManager supported (SYNC_MANAGER)

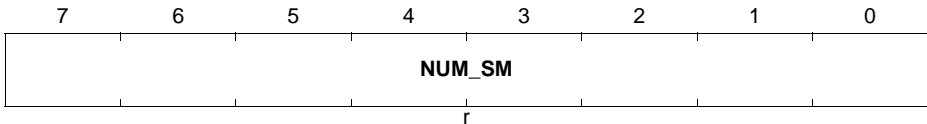
The register SYNC_MANAGER shows the number of supported SyncManager channels.

ECAT0_SYNC_MANAGER

SyncManagers Supported

(0005_H)

Reset Value: 08_H



Field	Bits	Type	Description
NUM_SM	[7:0]	r	Number of supported SyncManager channels (or entities) of the EtherCAT Slave Controller

RAM size of the ECAT module (RAM_SIZE)

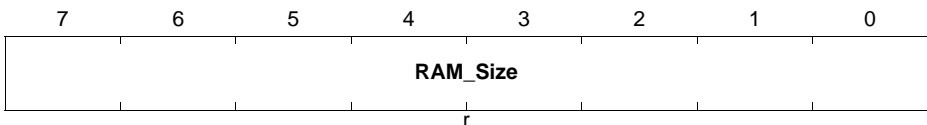
The RAM_SIZE register shows implemented Process Data RAM size in KByte.

ECAT0_RAM_SIZE

RAM Size

(0006_H)

Reset Value: 08_H



Field	Bits	Type	Description
RAM_Size	[7:0]	r	Process Data RAM size supported by the EtherCAT Slave Controller in KByte

EtherCAT Slave Controller (ECAT)

Port descriptor register (PORT_DESC)

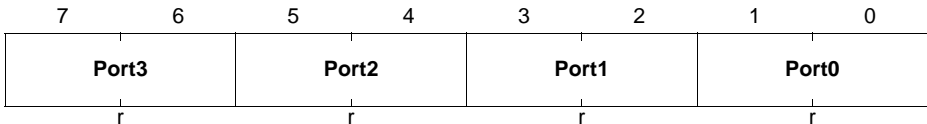
The port descriptor register shows the feature and availability of the implemented communication port of the ECAT module.

ECAT0_PORT_DESC

Port Descriptor

(0007_H)

Reset Value: 0F_H



Field	Bits	Type	Description
Port0	[1:0]	r	Port Configuration 00 _B Not implemented 01 _B Not configured (SII EEPROM) 10 _B EBUS 11 _B MII / RMII / RGMII
Port1	[3:2]	r	Port Configuration 00 _B Not implemented 01 _B Not configured (SII EEPROM) 10 _B EBUS 11 _B MII / RMII / RGMII
Port2	[5:4]	r	Port Configuration 00 _B Not implemented 01 _B Not configured (SII EEPROM) 10 _B EBUS 11 _B MII / RMII / RGMII
Port3	[7:6]	r	Port Configuration 00 _B Not implemented 01 _B Not configured (SII EEPROM) 10 _B EBUS 11 _B MII / RMII / RGMII

EtherCAT Slave Controller (ECAT)

ECAT features supported (FEATURE)

The register features supported shows the activated functionality of the ECAT module.

ECAT0_FEATURE

ESC Features Supported

(0008_H)

Reset Value: 01CC_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			FX CON F	RW CS	LRW _CS	EDC _SY _NCA	SH FCS E	ELD _MII	ELD _EB US	LJ_E BUS	CLK S_W	CLK S	Res	FMM U	
r			r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
FMMU	0	r	FMMU Operation 0 _B Bit oriented 1 _B Byte oriented
Res	1	r	Reserved Read as 0; should be written with 0.
CLKS	2	r	Distributed Clocks 0 _B Not available 1 _B Available
CLKS_W	3	r	Distributed Clocks (width) 0 _B 32 bits 1 _B 64 bits
LJ_EBUS	4	r	Low Jitter EBUS 0 _B Not available, standard jitter 1 _B Available, jitter minimized
ELD_EBUS	5	r	Enhanced Link Detection EBUS 0 _B Not available 1 _B Available
ELD_MII	6	r	Enhanced Link Detection MII 0 _B Not available 1 _B Available
SH_FCSE	7	r	Separate Handling of FCS Errors 0 _B Not supported 1 _B Supported, frames with wrong FCS and additional nibble will be counted separately in Forwarded RX Error Counter

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
EDC_SYNCA	8	r	Enhanced DC SYNC Activation 0 _B Not available 1 _B Available <i>Note: This feature refers to Bits 3-7 and</i>
LRW_CS	9	r	EtherCAT LRW command support 0 _B Supported 1 _B Not supported
RW_CS	10	r	EtherCAT read/write command support (BRW, APRW, FPRW) 0 _B Supported 1 _B Not supported
FX_CONF	11	r	Fixed FMMU/SyncManager configuration 0 _B Variable configuration 1 _B Fixed configuration (refer to documentation of supporting ESCs)
Res	[15:12]	r	Reserved Read as 0; should be written with 0.

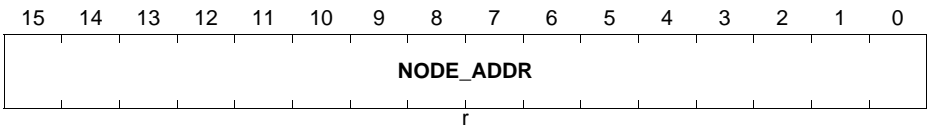
15.15.2 Station Address

Configured Station Address (STATION_ADR)

The STATION_ADR register is used for node addressing.

ECAT0_STATION_ADR

Configured Station Address (0010_H) Reset Value: 0000_H



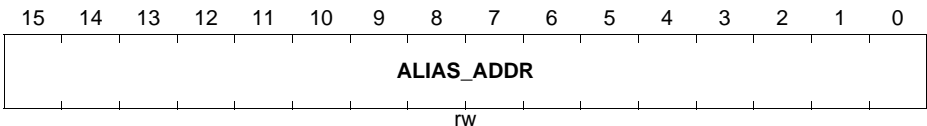
Field	Bits	Type	Description
NODE_ADDR	[15:0]	r	Address used for node addressing (FPxx commands)

Configured Station Alias (STATION_ALIAS)

The STATION_ALIAS register is used for station addressing.

ECAT0_STATION_ALIAS

Configured Station Alias (0012_H) Reset Value: 0000_H



Field	Bits	Type	Description
ALIAS_ADDR	[15:0]	rw	Alias Address used for node addressing (FPxx commands) The use of this alias is activated by Register . / . (0x0103.0) <i>Note: NOTE: EEPROM value is only taken over at first EEPROM load after power-on or reset.</i>

15.15.3 Write Protection

Write Register Enable (WR_REG_ENABLE)

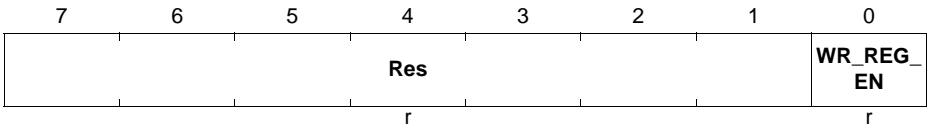
The Write Register Enable register has to be written in the same Ethernet frame (value does not care) before other writes to this station are allowed.

ECAT0_WR_REG_ENABLE

Write Register Enable

(0020_H)

Reset Value: 00_H



Field	Bits	Type	Description
WR_REG_EN	0	r	Write register protection enabled If write register protection is enabled, this register has to be written in the same Ethernet frame (value does not care) before other writes to this station are allowed. Write protection is still active after this frame (if Write Register Protection register is not changed).
Res	[7:1]	r	Reserved Read as 0; should be written with 0.

EtherCAT Slave Controller (ECAT)

Write Register Protection (WR_REG_PROTECT)

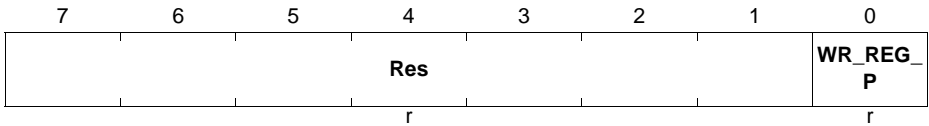
The WR_REG_PROTECT register shows the status of the write register protection.

ECAT0_WR_REG_PROTECT

Write Register Protection

(0021_H)

Reset Value: 00_H



Field	Bits	Type	Description
WR_REG_P	0	r	Write register protection 0 _B Protection disabled 1 _B Protection enabled <i>Note: Registers -, - are write protected, except for .</i>
Res	[7:1]	r	Reserved Read as 0; should be written with 0.

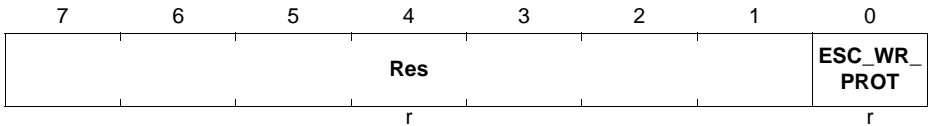
EtherCAT Slave Controller (ECAT)

ESC Write Enable (ESC_WR_ENABLE)

The ESC_WR_ENABLE register has to be written in the same Ethernet frame (value does not care) before other writes to this station are allowed.

ECAT0_ESC_WR_ENABLE

ESC Write Enable (0030_H) **Reset Value: 00_H**



Field	Bits	Type	Description
ESC_WR_PROT	0	r	Write protection enabled If ESC write protection is enabled, this register has to be written in the same Ethernet frame (value does not care) before other writes to this station are allowed. ESC write protection is still active after this frame (if ESC Write Protection register is not changed).
Res	[7:1]	r	Reserved Read as 0; should be written with 0.

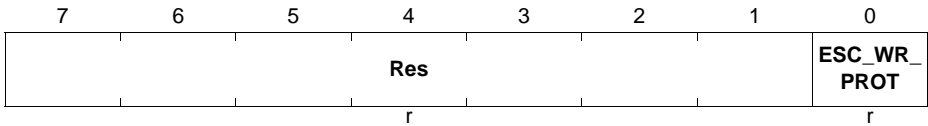
EtherCAT Slave Controller (ECAT)

ESC Write Protection (ESC_WR_PROTECT)

The ESC_WR_PROTECT register shows the status of the ESC write protection.

ECAT0_ESC_WR_PROTECT

ESC Write Protection (0031_H) **Reset Value: 00_H**



Field	Bits	Type	Description
ESC_WR_PROT	0	r	Write protect 0 _B Protection disabled 1 _B Protection enabled <i>Note: All areas are write protected, except for .</i>
Res	[7:1]	r	Reserved Read as 0; should be written with 0.

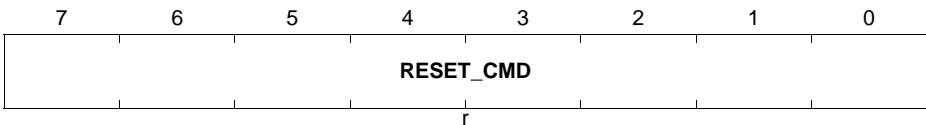
15.15.4 Data Link Layer

ESC Reset ECAT - Write (ESC_RESET_ECAT)

The ESC_RESET_ECAT register is used from the EtherCAT Master to reset the ESC.

ECAT0_ESC_RESET_ECAT

ESC Reset ECAT [WRITE Mode] (0040_H) Reset Value: 00_H

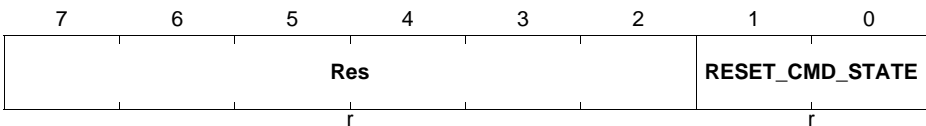


Field	Bits	Type	Description
RESET_CMD	[7:0]	r	Reset commands issued by EtherCAT Master A reset is asserted after writing 0x52 ('R'), 0x45 ('E') and 0x53 ('S') in this register with 3 consecutive frames.

Register ESC Reset ECAT - Read (ESC_RESET_ECAT)

ECAT0_ESC_RESET_ECAT

ESC Reset ECAT [READ Mode] (0040_H) Reset Value: 00_H



Field	Bits	Type	Description
RESET_CMD_STATE	[1:0]	r	Progress of the reset procedure 01 _B after writing 0x52 10 _B after writing 0x45 (if 0x52 was written before) 00 _B default
Res	[7:2]	r	Reserved Read as 0; should be written with 0.

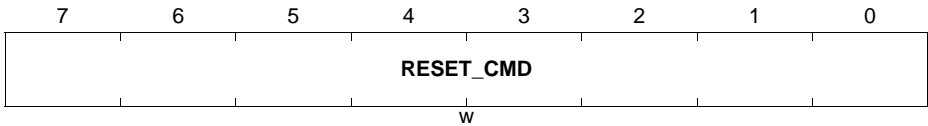
EtherCAT Slave Controller (ECAT)

ESC Reset PDI - Write (ESC_RESET_PDI)

The ESC_RESET_PDI register is used from the XMC4300 to reset the ESC.

ECAT0_ESC_RESET_PDI

ESC Reset PDI [WRITE Mode] (0041_H) Reset Value: 00_H

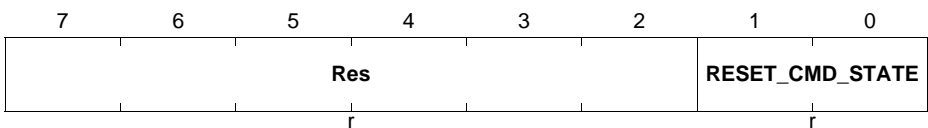


Field	Bits	Type	Description
RESET_CMD	[7:0]	w	Reset commands issued by XMC4300 A reset is asserted after writing 0x52 ('R'), 0x45 ('E') and 0x53 ('S') in this register with 3 consecutive writes accesses.

Register ESC Reset PDI - Read (ESC_RESET_PDI)

ECAT0_ESC_RESET_PDI

ESC Reset PDI [READ Mode] (0041_H) Reset Value: 00_H



Field	Bits	Type	Description
RESET_CMD_STATE	[1:0]	r	Progress of the reset procedure 01 _B after writing 0x52 10 _B after writing 0x45 (if 0x52 was written before) 00 _B default
Res	[7:2]	r	Reserved Read as 0; should be written with 0.

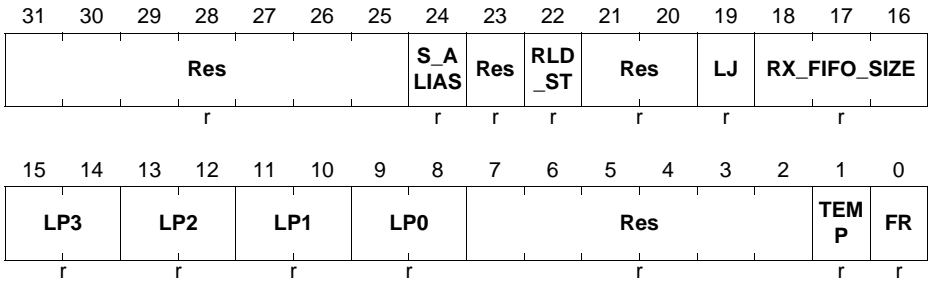
EtherCAT Slave Controller (ECAT)

ESC DL Control (ESC_DL_CONTROL)

The ESC_DL_CONTROL register is used to handle specific feature of the Data Link Layer.

ECAT0_ESC_DL_CONTROL

ESC DL Control (0100_H) Reset Value: 0007 C001_H



Field	Bits	Type	Description
FR	0	r	<p>Forwarding rule</p> <p>0_B EtherCAT frames are processed, Non-EtherCAT frames are forwarded without processing</p> <p>1_B EtherCAT frames are processed, Non- EtherCAT frames are destroyed</p> <p>The source MAC address is changed for every frame (SOURCE_MAC[1] is set to 1 – locally administered address) regardless of the forwarding rule.</p>
TEMP	1	r	<p>Temporary use of settings in LP1-LP3</p> <p>0_B permanent use</p> <p>1_B use for about 1 second, then revert to previous settings</p>
Res	[7:2]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
LP0	[9:8]	r	<p>Loop Port 0</p> <p>00_B Auto 01_B Auto Close 10_B Open 11_B Closed</p> <p><i>Note: Loop open means sending/receiving over this port is enabled, loop closed means sending/receiving is disabled and frames are forwarded to the next open port internally.</i></p> <p><i>Auto: loop closed at link down, opened at link up</i> <i>Auto Close: loop closed at link down, opened with writing 01 again after link up (or receiving a valid Ethernet frame at the closed port)</i> <i>Open: loop open regardless of link state</i> <i>Closed: loop closed regardless of link state</i></p>
LP1	[11:10]	r	<p>Loop Port 1</p> <p>00_B Auto 01_B Auto Close 10_B Open 11_B Closed</p>
LP2	[13:12]	r	<p>Loop Port 2</p> <p>00_B Auto 01_B Auto Close 10_B Open 11_B Closed</p>
LP3	[15:14]	r	<p>Loop Port 3</p> <p>00_B Auto 01_B Auto Close 10_B Open 11_B Closed</p>

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
RX_FIFO_SIZE	[18:16]	r	RX FIFO Size ESC delays start of forwarding until FIFO is at least half full RX FIFO Size/RX delay reduction value. ¹⁾ MII 0 _D -40 ns (-80 ns ²⁾ 1 _D -40 ns (-80 ns ²⁾ 2 _D -40 ns 3 _D -40 ns 4 _D no change 5 _D no change 6 _D no change 7 _D default
LJ	19	r	EBUS Low Jitter 0 _B Normal jitter 1 _B Reduced jitter
Res	[21:20]	r	Reserved Read as 0; should be written with 0.
RLD_ST	22	r	EBUS remote link down signaling time 0 _B Default (~660 ms) 1 _B Reduced (~80 µs)
Res	23	r	Reserved Read as 0; should be written with 0.
S_ALIAS	24	r	Station alias 0 _B Ignore Station Alias 1 _B Alias can be used for all configured address command types (FPRD, FPWR, ...)
Res	[31:25]	r	Reserved Read as 0; should be written with 0.

1) The possibility of RX FIFO Size reduction depends on the clock source accuracy of the ESC and of every connected EtherCAT/Ethernet devices (master, slave, etc.). RX FIFO Size of 7 is sufficient for 100ppm accuracy, FIFO Size 0 is possible with 25ppm accuracy (frame size of 1518/1522 Byte).

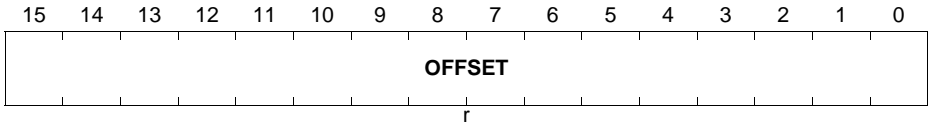
2) IP Core since V3.0.0/V3.00c only

Physical Read/Write Offset (PHYSICAL_RW_OFFSET)

The PHYSICAL_RW_OFFSET register is used by the EtherCAT Master to generate offset of R/W Commands (FPRW, APRW) between Read address and Write address.

ECAT0_PHYSICAL_RW_OFFSET

Physical Read/Write Offset (0108_H) Reset Value: 0000_H



Field	Bits	Type	Description
OFFSET	[15:0]	r	Offset of R/W Commands (FPRW, APRW) between Read address and Write address RD_ADR = ADR and WR_ADR = ADR + R/W-Offset

EtherCAT Slave Controller (ECAT)

ESC DL Status (ESC_DL_STATUS)

The ESC_DL_STATUS register shows the current state of the Data Link Layer.

ECAT0_ESC_DL_STATUS

ESC DL Status (0110_H) **Reset Value: 5000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COM_P3	LP3	COM_P2	LP2	COM_P1	LP1	COM_P0	LP0	LINK_P3	LINK_P2	LINK_P1	LINK_P0	Res	ELD	PDI_WDT_S	PDI_EEPROM
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
PDI_EEPROM	0	r	PDI operational/EEPROM loaded correctly 0 _B EEPROM not loaded, PDI not operational (no access to Process Data RAM) 1 _B EEPROM loaded correctly, PDI operational (access to Process Data RAM)
PDI_WDT_S	1	r	PDI Watchdog Status 0 _B Watchdog expired 1 _B Watchdog reloaded
ELD	2	r	Enhanced Link detection 0 _B Deactivated for all ports 1 _B Activated for at least one port <i>Note: EEPROM value is only taken over at first EEPROM load after power-on or reset</i>
Res	3	r	Reserved Read as 0; should be written with 0.
LINK_P0	4	r	Physical link on Port 0 0 _B No link 1 _B Link detected
LINK_P1	5	r	Physical link on Port 1 0 _B No link 1 _B Link detected
LINK_P2	6	r	Physical link on Port 2 0 _B No link 1 _B Link detected

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
LINK_P3	7	r	Physical link on Port 3 0 _B No link 1 _B Link detected
LP0	8	r	Loop Port 0 0 _B Open 1 _B Closed
COM_P0	9	r	Communication on Port 0 0 _B No stable communication 1 _B Communication established
LP1	10	r	Loop Port 1 0 _B Open 1 _B Closed
COM_P1	11	r	Communication on Port 1 0 _B No stable communication 1 _B Communication established
LP2	12	r	Loop Port 2 0 _B Open 1 _B Closed
COM_P2	13	r	Communication on Port 2 0 _B No stable communication 1 _B Communication established
LP3	14	r	Loop Port 3 0 _B Open 1 _B Closed
COM_P3	15	r	Communication on Port 3 0 _B No stable communication 1 _B Communication established

Decoding of ECAT Port states

**Table 15-8 Decoding port state in ESC DL Status register high byte
(typical modes only)**

Register 0111_H	Port 3	Port 2	Port 1	Port 0
0x55	No link, closed	No link, closed	No link, closed	No link, closed
0x56	No link, closed	No link, closed	No link, closed	Link, open
0x59	No link, closed	No link, closed	Link, open	No link, closed
0x5A	No link, closed	No link, closed	Link, open	Link, open
0x65	No link, closed	Link, open	No link, closed	No link, closed
0x66	No link, closed	Link, open	No link, closed	Link, open
0x69	No link, closed	Link, open	Link, open	No link, closed
0x6A	No link, closed	Link, open	Link, open	Link, open
0x95	Link, open	No link, closed	No link, closed	No link, closed
0x96	Link, open	No link, closed	No link, closed	Link, open
0x99	Link, open	No link, closed	Link, open	No link, closed
0x9A	Link, open	No link, closed	Link, open	Link, open
0xA5	Link, open	Link, open	No link, closed	No link, closed
0xA6	Link, open	Link, open	No link, closed	Link, open
0xA9	Link, open	Link, open	Link, open	No link, closed
0xAA	Link, open	Link, open	Link, open	Link, open
0xD5	Link, closed	No link, closed	No link, closed	No link, closed
0xD6	Link, closed	No link, closed	No link, closed	Link, open
0xD9	Link, closed	No link, closed	Link, open	No link, closed
0xDA	Link, closed	No link, closed	Link, open	Link, open

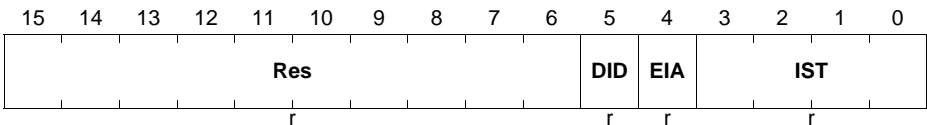
15.15.5 Application Layer

AL Control (AL_CONTROL)

The AL_CONTROL register reflect the current state of the ESC.

ECAT0_AL_CONTROL

AL Control (0120_H) **Reset Value: 0001_H**



Field	Bits	Type	Description
IST	[3:0]	r	Initiate State Transition of the Device State Machine 1 _H Request Init State 2 _H Request Pre-Operational State 3 _H Request Bootstrap State 4 _H Request Safe-Operational State 8 _H Request Operational State
EIA	4	r	Error Ind Ack 0 _B No Ack of Error Ind in AL status register 1 _B Ack of Error Ind in AL status register
DID	5	r	Device Identification 0 _B No request 1 _B Device Identification request
Res	[15:6]	r	Reserved Read as 0; should be written with 0.

Note: AL Control register behaves like a mailbox if Device Emulation is off (0x0140.8=0): The PDI has to read/write the AL Control register after EtherCAT Master has written it. Otherwise the EtherCAT Master cannot write again to the AL Control register. After Reset, AL Control register can be written by EtherCAT Master. (Regarding mailbox functionality, both registers and are equivalent, e.g. reading is sufficient to make this register writeable again.)*

* PDI register function acknowledge by Write command is disabled: Reading AL Control from PDI clears AL Event Request .AL_CE. Writing to this register from PDI is not possible.

EtherCAT Slave Controller (ECAT)

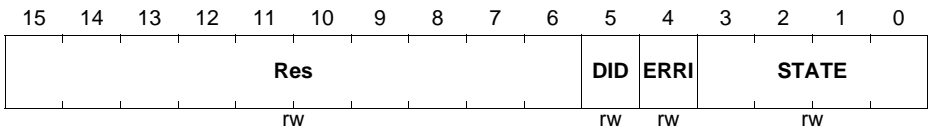
PDI register function acknowledge by Write command is enabled: Writing AL Control from PDI clears AL Event Request .AL_CE. Writing to this register from PDI is possible; write value is ignored (write 0).

AL Status (AL_STATUS)

The AL_STATUS register is used to acknowledge the state change request from the EtherCAT Master.

ECAT0_AL_STATUS

AL Status (0130_H) **Reset Value: 0001_H**



Field	Bits	Type	Description
STATE	[3:0]	rw	Actual State of the Device State Machine 1 _H Init State 2 _H Pre-Operational State 3 _H Bootstrap State 4 _H Safe-Operational State 8 _H Operational State
ERRI	4	rw	Error Ind 0 _B Device is in State as requested or Flag cleared by command 1 _B Device has not entered requested State or changed State as result of a local action
DID	5	rw	Device Identification 0 _B Device Identification not valid 1 _B Device Identification loaded
Res	[15:6]	rw	Reserved Read as 0; should be written with 0.

Note: AL Status register is only writable from PDI if Device Emulation is off (.EMUL), otherwise AL Status register will reflect AL Control register values.

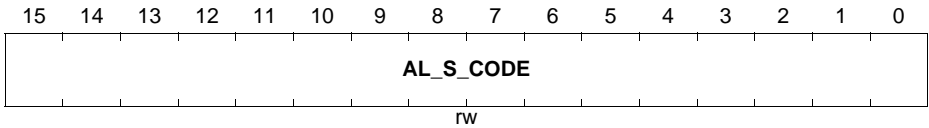
EtherCAT Slave Controller (ECAT)

AL Status Code (AL_STATUS_CODE)

The AL_STATUS_CODE register is used to exchange Status Codes of the Application Layer.

ECAT0_AL_STATUS_CODE

AL Status Code (0134_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
AL_S_CODE	[15:0]	rw	AL Status Code

EtherCAT Slave Controller (ECAT)

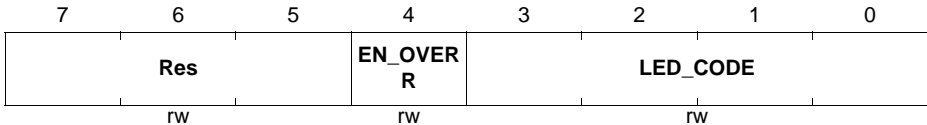
RUN LED Override (RUN_LED)

The RUN_LED register is used to overwrite the blinking sequences of the ECAT RUN LED.

ECAT0_RUN_LED
RUN LED Override

(0138_H)

Reset Value: 0E_H



Field	Bits	Type	Description
LED_CODE	[3:0]	rw	LED Code 0 _H OFF - Init State 1 _H Flash - SafeOp D _H Blinking - PreOp E _H Flickering - Bootstrap F _H On - Operational Values 2-C are used for future LED flash sequences.
EN_OVERR	4	rw	Enable Override 0 _B Override disable 1 _B Override enable
Res	[7:5]	rw	Reserved Read as 0; should be written with 0.

Note: The Reset value will change after reloading the EEPROM. This value will change to 0x00 to signalize the init State of the ECAT module.

Note: Changes to AL Status register with valid values will disable RUN LED Override. The value read in this register always reflects current LED output.

EtherCAT Slave Controller (ECAT)

ERR LED Override (ERR_LED)

The ERR_LED register is used to overwrite the blinking sequences of the ECAT ERR LED.

ECAT0_ERR_LED
RUN ERR Override

(0139_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
Res		EN_OVERR		LED_CODE			
rw		rw		rw			

Field	Bits	Type	Description
LED_CODE	[3:0]	rw	LED Code 0 _H OFF D _H Blinking E _H Flickering F _H On Values 1-C are used for future LED flash sequences.
EN_OVERR	4	rw	Enable Override 0 _B Override disable 1 _B Override enable
Res	[7:5]	rw	Reserved Read as 0; should be written with 0.

Note: New error conditions will disable the ERR LED Overwrite. The value read in this register always reflects the current LED output.

15.15.6 PDI / ESC Configuration

PDI CONTROL

The PDI_CONTROL register shows the used Interface for the process data interface.

ECAT0_PDI_CONTROL

PDI Control (0140_H) **Reset Value: 80_H**



Field	Bits	Type	Description
PDI	[7:0]	r	On-chip bus clock 00 _H Interface deactivated (no PDI) 80 _H On-chip Bus

EtherCAT Slave Controller (ECAT)

ESC Configuration (ESC_CONFIG)

The ESC_CONFIG register shows the used features depending on the EEPROM content.

ECAT0_ESC_CONFIG

ESC Configuration

(0141_H)

Reset Value: FE_H

7	6	5	4	3	2	1	0
EHL_D_P3	EHL_D_P2	EHL_D_P1	EHL_D_P0	CLKS_IN	CLKS_OUT T	EHL_D	EMUL
r	r	r	r	r	r	r	r

Field	Bits	Type	Description
EMUL	0	r	Device emulation (control of AL status) 0 _B AL status register has to be set by PDI 1 _B AL status register will be set to value written to AL control register
EHL_D	1	r	Enhanced Link detection all ports 0 _B disabled (if bits [7:4]=0) 1 _B enabled at all ports (overrides bits [7:4])
CLKS_OUT	2	r	Distributed Clocks SYNC Out Unit 0 _B disabled (power saving) 1 _B enabled
CLKS_IN	3	r	Distributed Clocks Latch In Unit 0 _B disabled (power saving) 1 _B enabled
EHL_D_P0	4	r	Enhanced Link port 0 0 _B disabled (if bit 1 = 0) 1 _B enabled
EHL_D_P1	5	r	Enhanced Link port 1 0 _B disabled (if bit 1 = 0) 1 _B enabled
EHL_D_P2	6	r	Enhanced Link port 2 0 _B disabled (if bit 1 = 0) 1 _B enabled
EHL_D_P3	7	r	Enhanced Link port 3 0 _B disabled (if bit 1 = 0) 1 _B enabled

EtherCAT Slave Controller (ECAT)

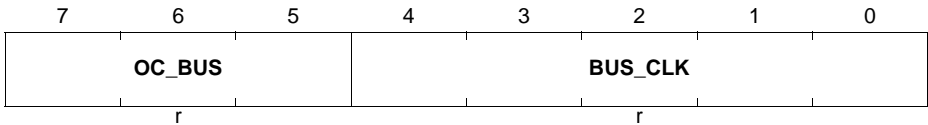
Note: The Reset value will change after reloading the EEPROM.

PDI On-chip bus configuration (PDI_CONFIG)

The PDI_CONFIG register shows used on-chip bus used inside the XMC4300. The configuration of the PDI configuration can not be changed via EEPROM.

ECAT0_PDI_CONFIG

PDI Control (0150_H) Reset Value: 81_H



Field	Bits	Type	Description
BUS_CLK	[4:0]	r	On-chip bus clock 00 _H asynchronous 01 _H values 1-31 is used for synchronous multiplication factor (N*25Mhz)
OC_BUS	[7:5]	r	On-chip bus 000 _H Altera Avalon 001 _H AXI 010 _H Xilinx PLB v4.6 100 _H Xilinx OPB

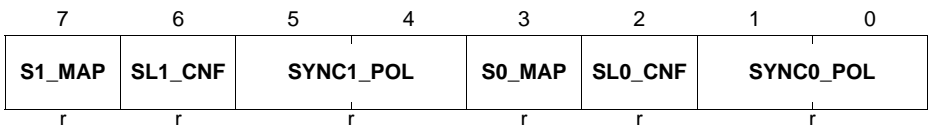
Note: The On chip bus Xilinx OPB is bridged to AHB lite.

Sync/Latch[1:0] PDI Configuration (SYNC_LATCH_CONFIG)

The SYNC_LATCH_CONFIG register shows the setup of the SYNC pins.

ECAT0_SYNC_LATCH_CONFIG

Sync/Latch[1:0] PDI Configuration (0151_H) Reset Value: EE_H



Field	Bits	Type	Description
SYNC0_POL	[1:0]	r	SYNC0 output driver/polarity¹⁾ 00 _B Push-Pull active low 01 _B Open Drain (active low) 10 _B Push-Pull active high 11 _B Open Source (active high)
SL0_CNF	2	r	SYNC0/LATCH0 configuration²⁾ 0 _B LATCH0 Input 1 _B SYNC0 Output
S0_MAP	3	r	SYNC0 mapped to register . 0 _B Disabled 1 _B Enabled
SYNC1_POL	[5:4]	r	SYNC1 output driver/polarity³⁾ 00 _B Push-Pull active low 01 _B Open Drain (active low) 10 _B Push-Pull active high 11 _B Open Source (active high)
SL1_CNF	6	r	SYNC1/LATCH1 configuration 0 _B LATCH1 Input 1 _B SYNC1 Output
S1_MAP	7	r	SYNC1 mapped to register . 0 _B Disabled 1 _B Enabled

1) SYNC0 output driver modes are selected in the IOCR bitfile of the respective pin which is used.

2) The ECAT0 module uses independent SYNC[1:0] outputs and LATCH[1:0] inputs, independent of this configuration.

3) SYNC1 output driver modes are selected in the IOCR bitfile of the respective pin which is used.

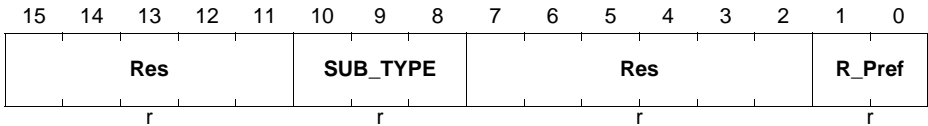
Note: Register SYNC_LATCH_CONFIG is reloaded from EEPROM ADR 0x0001.

Register PDI On-chip bus extended Configuration (PDI_EXT_CONFIG)

ECAT0_PDI_EXT_CONFIG

PDI Synchronous Microcontroller extended Configuration
(0152_H)

Reset Value: 0000_H



Field	Bits	Type	Description
R_Pref	[1:0]	r	Read Prefetch Size 00 _B 4 cycles 01 _B 1 cycle (typical) 10 _B 2 cycles 11 _B Reserved
Res	[7:2]	r	Reserved
SUB_TYPE	[10:8]	r	On-chip Sub Type for AXI 000 _B AXI3 001 _B AXI4 010 _B AXI4 Lite
Res	[15:11]	r	Reserved

15.15.7 Interrupts

ECAT Event Mask (EVENT_MASK)

The ECAT_EVENT_MASK shows the status of the masked ECAT events.

ECAT0_EVENT_MASK

ECAT Event Mask (0200_H) Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			MIR_7_M ASK	MIR_6_M ASK	MIR_5_M ASK	MIR_4_M ASK	MIR_3_M ASK	MIR_2_M ASK	MIR_1_M ASK	MIR_0_M ASK	AL_SE_MASK	DL_SE_MASK	Res	DC_LE_MASK	
			r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
DC_LE_MASK	0	r	DC Latch event 0 _B Corresponding ECAT Event Request register bit is not mapped 1 _B Corresponding ECAT Event Request register bit is mapped
Res	1	r	Reserved Read as 0; should be written with 0.
DL_SE_MASK	2	r	DL Status event 0 _B Corresponding ECAT Event Request register bit is not mapped 1 _B Corresponding ECAT Event Request register bit is mapped
AL_SE_MASK	3	r	AL Status event 0 _B Corresponding ECAT Event Request register bit is not mapped 1 _B Corresponding ECAT Event Request register bit is mapped
MIR_0_MASK	4	r	Mirrors values of each SyncManager Status 0 _B Corresponding ECAT Event Request register bit is not mapped 1 _B Corresponding ECAT Event Request register bit is mapped

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
MIR_1_MASK	5	r	Mirrors values of each SyncManager Status 0 _B Corresponding ECAT Event Request register bit is not mapped 1 _B Corresponding ECAT Event Request register bit is mapped
MIR_2_MASK	6	r	Mirrors values of each SyncManager Status 0 _B Corresponding ECAT Event Request register bit is not mapped 1 _B Corresponding ECAT Event Request register bit is mapped
MIR_3_MASK	7	r	Mirrors values of each SyncManager Status 0 _B Corresponding ECAT Event Request register bit is not mapped 1 _B Corresponding ECAT Event Request register bit is mapped
MIR_4_MASK	8	r	Mirrors values of each SyncManager Status 0 _B Corresponding ECAT Event Request register bit is not mapped 1 _B Corresponding ECAT Event Request register bit is mapped
MIR_5_MASK	9	r	Mirrors values of each SyncManager Status 0 _B Corresponding ECAT Event Request register bit is not mapped 1 _B Corresponding ECAT Event Request register bit is mapped
MIR_6_MASK	10	r	Mirrors values of each SyncManager Status 0 _B Corresponding ECAT Event Request register bit is not mapped 1 _B Corresponding ECAT Event Request register bit is mapped
MIR_7_MASK	11	r	Mirrors values of each SyncManager Status 0 _B Corresponding ECAT Event Request register bit is not mapped 1 _B Corresponding ECAT Event Request register bit is mapped
Res	[15:12]	r	Reserved Read as 0; should be written with 0.

EtherCAT Slave Controller (ECAT)

Register PDI AL Event Mask (AL_EVENT_MASK)

The AL_EVENT_MASK register is used to mask the interrupt source for the AL Interrupt.

ECAT0_AL_EVENT_MASK

PDI AL Event Mask

(0204_H)

Reset Value: 00FF FF2F_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								SMI_15_MAS	SMI_14_MAS	SMI_13_MAS	SMI_12_MAS	SMI_11_MAS	SMI_10_MAS	SMI_9_MAS	SMI_8_MAS
rw								K	K	K	K	K	K	K	K
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMI_7_MAS	SMI_6_MAS	SMI_5_MAS	SMI_4_MAS	SMI_3_MAS	SMI_2_MAS	SMI_1_MAS	SMI_0_MAS	Res	WP_D_MAS	EEP_E_MAS	SM_A_MAS	ST_S1_MAS	ST_S0_MAS	DC_LE_MAS	AL_CE_MAS
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
AL_CE_MASK	0	rw	AL Control event 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
DC_LE_MASK	1	rw	DC Latch event 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
ST_S0_MASK	2	rw	State of DC SYNC0 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
ST_S1_MASK	3	rw	State of DC SYNC1 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
SM_A_MASK	4	rw	SyncManager activation register changed 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
EEP_E_MASK	5	rw	EEPROM Emulation 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
WP_D_MASK	6	rw	Watchdog Process Data 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
Res	7	rw	Reserved Read as 0; should be written with 0.
SMI_0_MASK	8	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_1_MASK	9	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_2_MASK	10	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_3_MASK	11	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
SMI_4_MASK	12	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_5_MASK	13	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_6_MASK	14	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_7_MASK	15	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_8_MASK	16	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_9_MASK	17	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_10_MASK	18	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
SMI_11_MASK	19	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_12_MASK	20	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_13_MASK	21	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_14_MASK	22	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
SMI_15_MASK	23	rw	SyncManager interrupt 0 _B Corresponding AL Event Request register bit is not mapped 1 _B Corresponding AL Event Request register bit is mapped
Res	[31:24]	rw	Reserved Read as 0; should be written with 0.

EtherCAT Slave Controller (ECAT)

Register ECAT Event Request (EVENT_REQ)

The ECAT_EVENT_REQ register is used to show the active ECAT Event requests.

ECAT0_EVENT_REQ

ECAT Event Request

(0210_H)

Reset Value: 0004_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				MIR_7	MIR_6	MIR_5	MIR_4	MIR_3	MIR_2	MIR_1	MIR_0	AL_SE	DL_SE	Res	DC_LE
				r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
DC_LE	0	r	DC Latch event 0 _B No change on DC Latch Inputs 1 _B At least one change on DC Latch Inputs <i>Note: Bit is cleared by reading DC Latch event times from ECAT for ECAT controlled Latch Units, so that Latch 0/1 Status 09AEH:09AFH indicates no event</i>
Res	1	r	Reserved Read as 0; should be written with 0.
DL_SE	2	r	DL Status event 0 _B No change in DL Status 1 _B DL Status change <i>Note: Bit is cleared by reading out DL Status 0110H:0111H from ECAT</i>
AL_SE	3	r	AL Status event 0 _B No change in AL Status 1 _B AL Status change <i>Note: Bit is cleared by reading out AL Status 0130H:0131H from ECAT</i>
MIR_0	4	r	Mirrors values of each SyncManager Status 0 _B No Sync Channel 0 event 1 _B Sync Channel 0 event pending
MIR_1	5	r	Mirrors values of each SyncManager Status 0 _B No Sync Channel 1 event 1 _B Sync Channel 1 event pending

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
MIR_2	6	r	Mirrors values of each SyncManager Status 0 _B No Sync Channel 2 event 1 _B Sync Channel 2 event pending
MIR_3	7	r	Mirrors values of each SyncManager Status 0 _B No Sync Channel 3 event 1 _B Sync Channel 3event pending
MIR_4	8	r	Mirrors values of each SyncManager Status 0 _B No Sync Channel 4 event 1 _B Sync Channel 4 event pending
MIR_5	9	r	Mirrors values of each SyncManager Status 0 _B No Sync Channel 5 event 1 _B Sync Channel 5 event pending
MIR_6	10	r	Mirrors values of each SyncManager Status 0 _B No Sync Channel 6 event 1 _B Sync Channel 6 event pending
MIR_7	11	r	Mirrors values of each SyncManager Status 0 _B No Sync Channel 7 event 1 _B Sync Channel 7 event pending
Res	[15:12]	r	Reserved Read as 0; should be written with 0.

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
SM_A	4	r	<p>SyncManager activation register changed SyncManager register offset 0x6 0_B No change in any SyncManager 1_B At least one change on DC Latch Inputs <i>Note: Bit is cleared by reading SyncManager Activation registers 0806H etc. from PDI</i></p>
EEP_E	5	r	<p>EEPROM Emulation 0_B No command pending 1_B EEPROM command pending <i>Note: Bit is cleared by acknowledging the command in from PDI</i></p>
WP_D	6	r	<p>Watchdog Process Data 0_B Has not expired 1_B Has expired <i>Note: Bit is cleared by reading from PDI</i></p>
Res	7	r	<p>Reserved Read as 0; should be written with 0.</p>
SMI_0	8	r	<p>SyncManager interrupt SyncManager, bit 0 or 1 0_B No SyncManager 0 interrupt 1_B SyncManager 0 interrupt pending</p>
SMI_1	9	r	<p>SyncManager interrupt SyncManager, bit 0 or 1 0_B No SyncManager 0 interrupt 1_B SyncManager 0 interrupt pending</p>
SMI_2	10	r	<p>SyncManager interrupt SyncManager, bit 0 or 1 0_B No SyncManager 0 interrupt 1_B SyncManager 0 interrupt pending</p>
SMI_3	11	r	<p>SyncManager interrupt SyncManager, bit 0 or 1 0_B No SyncManager 0 interrupt 1_B SyncManager 0 interrupt pending</p>
SMI_4	12	r	<p>SyncManager interrupt SyncManager, bit 0 or 1 0_B No SyncManager 0 interrupt 1_B SyncManager 0 interrupt pending</p>

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
SMI_5	13	r	SyncManager interrupt SyncManager , bit 0 or 1 0 _B No SyncManager 0 interrupt 1 _B SyncManager 0 interrupt pending
SMI_6	14	r	SyncManager interrupt SyncManager , bit 0 or 1 0 _B No SyncManager 0 interrupt 1 _B SyncManager 0 interrupt pending
SMI_7	15	r	SyncManager interrupt SyncManager , bit 0 or 1 0 _B No SyncManager 0 interrupt 1 _B SyncManager 0 interrupt pending
SMI_8	16	r	SyncManager interrupt SyncManager , bit 0 or 1 0 _B No SyncManager 0 interrupt 1 _B SyncManager 0 interrupt pending
SMI_9	17	r	SyncManager interrupt SyncManager , bit 0 or 1 0 _B No SyncManager 0 interrupt 1 _B SyncManager 0 interrupt pending
SMI_10	18	r	SyncManager interrupt SyncManager , bit 0 or 1 0 _B No SyncManager 0 interrupt 1 _B SyncManager 0 interrupt pending
SMI_11	19	r	SyncManager interrupt SyncManager , bit 0 or 1 0 _B No SyncManager 0 interrupt 1 _B SyncManager 0 interrupt pending
SMI_12	20	r	SyncManager interrupt SyncManager , bit 0 or 1 0 _B No SyncManager 0 interrupt 1 _B SyncManager 0 interrupt pending
SMI_13	21	r	SyncManager interrupt SyncManager , bit 0 or 1 0 _B No SyncManager 0 interrupt 1 _B SyncManager 0 interrupt pending

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
SMI_14	22	r	SyncManager interrupt SyncManager , bit 0 or 1 0 _B No SyncManager 0 interrupt 1 _B SyncManager 0 interrupt pending
SMI_15	23	r	SyncManager interrupt SyncManager , bit 0 or 1 0 _B No SyncManager 0 interrupt 1 _B SyncManager 0 interrupt pending
Res	[31:24]	r	Reserved Read as 0; should be written with 0.

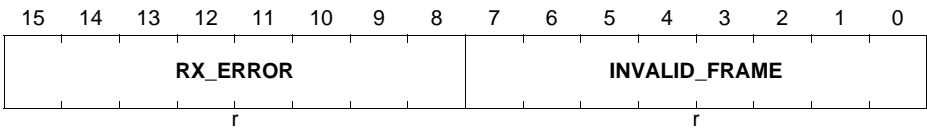
15.15.8 Error Counters

Register RX Error Counter Port p (RX_ERR_COUNT)

The RX_ERR_COUNT register shows the actual error count for each available and activated port (p).

ECAT0_RX_ERR_COUNT_p (p = 0-1)

RX Error Counter Port p (0300_H+p*2H) Reset Value: 0000_H



Field	Bits	Type	Description
INVALID_FRAME	[7:0]	r	Invalid frame counter of Port y Counting is stopped when FF _H is reached
RX_ERROR	[15:8]	r	RX Error counter of Port y (counting is stopped when 0xFF is reached). This is coupled directly to RX ERR of MII interface/EBUS interface.

Note: Error Counters 0300H-030BH are cleared if one of the RX Error counters 0300H-030BH is written by the ECAT master. Write value is ignored (write 0).

EtherCAT Slave Controller (ECAT)

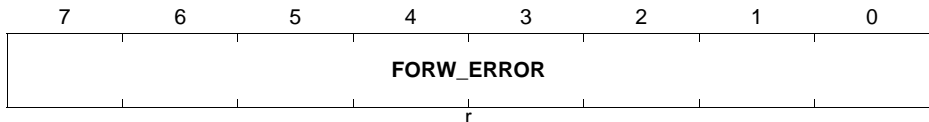
Register Forwarded RX Error Counter Port p (FWD_RX_ERR_COUNT)

The FWD_RX_ERR_COUNT shows the actual Forwarded RX Error Counter.

ECAT0_FWD_RX_ERR_COUNT_p (p = 0-1)

Forwarded RX Error Counter Port p (0308_H+p)

Reset Value: 00_H



Field	Bits	Type	Description
FORW_ERROR	[7:0]	r	Forwarded error counter of Port y counting is stopped when 0xFF is reached

Note: Error Counters 0300H-030BH are cleared if one of the RX Error counters 0300H-030BH is written by the ECAT master. Write value is ignored (write 0).

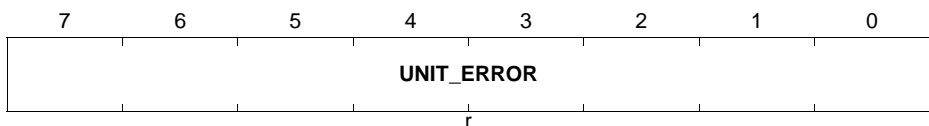
Register ECAT Processing Unit Error Counter (PROC_ERR_COUNT)

The ECAT_PROC_ERR_COUNT shows the actual Processing Unit Error Counter.

ECAT0_PROC_ERR_COUNT

ECAT Processing Unit Error Counter (030C_H)

Reset Value: 00_H



Field	Bits	Type	Description
UNIT_ERROR	[7:0]	r	ECAT Processing Unit error counter (counting is stopped when 0xFF is reached). Counts errors of frames passing the Processing Unit (e.g., FCS is wrong or datagram structure is wrong).

Note: Error Counter 030CH is cleared if error counter 030CH is written by the ECAT master. Write value is ignored (write 0).

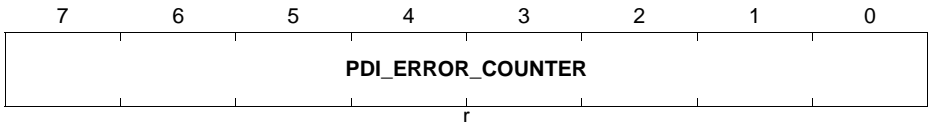
EtherCAT Slave Controller (ECAT)

Register PDI Error Counter (PDI_ERR_COUNT)

The PDI_ERR_COUNT shows the actual PDI Error Counter.

ECAT0_PDI_ERR_COUNT

PDI Error Counter (030D_H) **Reset Value: 00_H**



Field	Bits	Type	Description
PDI_ERROR_COUNTER	[7:0]	r	PDI Error counter (counting is stopped when 0xFF is reached). Counts if a PDI access has an interface error.

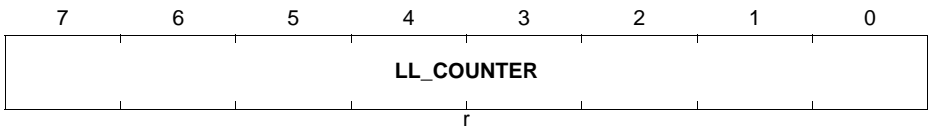
Note: Error Counter 030DH and Error Code 030EH are cleared if error counter 030DH is written. Write value is ignored (write 0).

Register Lost Link Counter Port y

The LOST_LINK_COUNT shows the actual Lost Link Counter for each available/activated port.

ECAT0_LOST_LINK_COUNT_p (p = 0-1)

Lost Link Counter Port p (0310_{H+p}) **Reset Value: 00_H**



Field	Bits	Type	Description
LL_COUNTER	[7:0]	r	Lost Link counter of Port p (counting is stopped when 0xFF is reached). Counts only if port loop is Auto.

Note: Only lost links at open ports are counted. Lost Link Counters 0310H-0313H are cleared if one of the Lost Link Counters 0310H-0313H is written by the ECAT master. Write value is ignored (write 0).

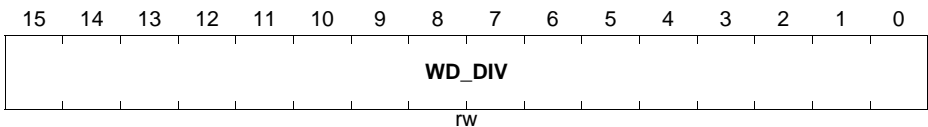
15.15.9 Watchdogs

Register Watchdog Divider (WD_DIVIDE)

The WD_DIVIDE register is used to configure the timebase of the Watchdogs.

ECAT0_WD_DIVIDE

Watchdog Divider (0400_H) **Reset Value: 09C2_H**



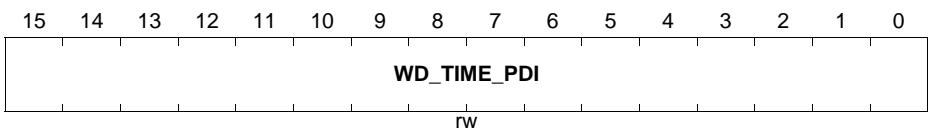
Field	Bits	Type	Description
WD_DIV	[15:0]	rw	Watchdog divider Number of 25 MHz tics (minus 2) that represents the basic watchdog increment. (Default value is 100is = 2498)

Register Watchdog Time PDI (WD_TIME_PDI)

The WD_TPDI register is used to setup the time for the PDI Watchdog.

ECAT0_WD_TIME_PDI

Watchdog Time PDI (0410_H) **Reset Value: 03E8_H**



Field	Bits	Type	Description
WD_TIME_PDI	[15:0]	rw	Watchdog Time PDI Number or basic watchdog increments (Default value with Watchdog divider 100is means 100ms Watchdog)

Note: Watchdog is disabled if Watchdog time is set to 0000H. Watchdog is restarted with every PDI access.

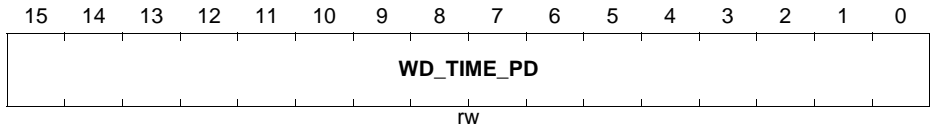
EtherCAT Slave Controller (ECAT)

Register Watchdog Time Process Data (WD_TIME_PDATA)

The WD_TIME_PDATA register is used to setup the time for the Process Data Watchdog.

ECAT0_WD_TIME_PDATA

Watchdog Time Process Data (0420_H) **Reset Value: 03E8_H**



Field	Bits	Type	Description
WD_TIME_PD	[15:0]	rw	Watchdog Time Process Data Number of basic watchdog increments (Default value with Watchdog divider 100is means 100 ms Watchdog)

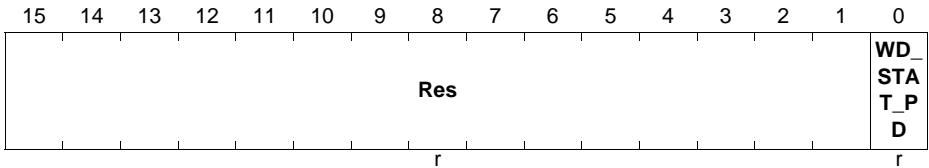
Note: There is one Watchdog for all SyncManagers. Watchdog is disabled if Watchdog time is set to 0000H. Watchdog is restarted with every write access to SyncManagers with Watchdog Trigger Enable Bit set.

Register Watchdog Status Process Data (WD_STAT_PDATA)

The WD_STAT_PDATA register shows the status of the Process Data WDT.

ECAT0_WD_STAT_PDATA

Watchdog Status Process Data (0440_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
WD_STAT_PD	0	r	Watchdog Status of Process Data triggered by SyncManagers 0 _B Watchdog Process Data expired 1 _B Watchdog Process Data is active or disabled
Res	[15:1]	r	Reserved Read as 0; should be written with 0.

Note: PDI register function acknowledge by Write command is enabled: Writing this register from PDI clears AL Event Request 0220H[6]. Writing to this register from PDI is possible; write value is ignored (write 0).

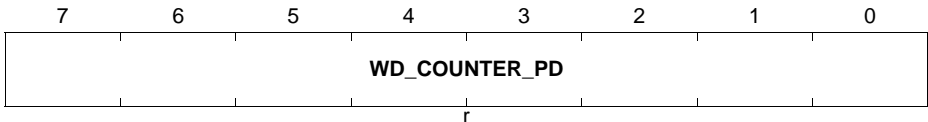
EtherCAT Slave Controller (ECAT)

Register Watchdog Counter Process Data (WD_COUNT_PDATA)

The WDC_PDI register shows the actual WDT counter for the Process Data.

ECAT0_WD_COUNT_PDATA

Watchdog Counter Process Data (0442_H) **Reset Value: 00_H**



Field	Bits	Type	Description
WD_COUNTER_PD	[7:0]	r	Watchdog Counter Process Data (counting is stopped when 0xFF is reached). Counts if Process Data Watchdog expires.

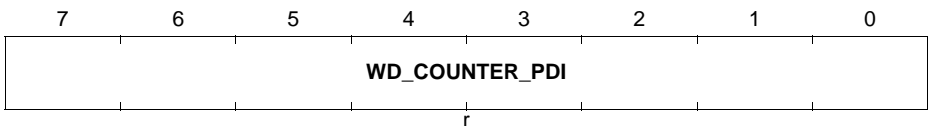
Note: Watchdog Counters or are cleared if one of the Watchdog Counters is written by the ECAT master. Write value is ignored (write 0).

Register Watchdog Counter PDI (WD_COUNT_PDI)

The WDC_PDI register shows the actual WDT counter for the PDI.

ECAT0_WD_COUNT_PDI

Watchdog Counter PDI (0443_H) **Reset Value: 00_H**



Field	Bits	Type	Description
WD_COUNTER_PDI	[7:0]	r	Watchdog PDI counter (counting is stopped when 0xFF is reached). Counts if PDI Watchdog expires.

Note: Watchdog Counters or are cleared if one of the Watchdog Counters is written by the ECAT master. Write value is ignored (write 0).

15.15.10 SII EEPROM Interface

SII EEPROM Interface (0500_H:050F_H)

Table 15-9 Table 63: SII EEPROM Interface Register overview

Register Address	Length (Byte)	Description
0500H	1	EEPROM Configuration
0501H	1	EEPROM PDI Access State
0502H:0503H	2	EEPROM Control/Status
0504H:0507H	4	EEPROM Address
0508H:050FH	4/8	EEPROM Data

EtherCAT controls the SSI EEPROM interface if EEPROM configuration register .TO_PDI = 0 and EEPROM PDI Access register .ACCESS = 0, otherwise PDI controls the EEPROM interface.

In EEPROM emulation mode, the PDI executes outstanding EEPROM commands. The PDI has access to some registers while the EEPROM Interface is busy.

Register EEPROM Configuration (EEP_CONF)

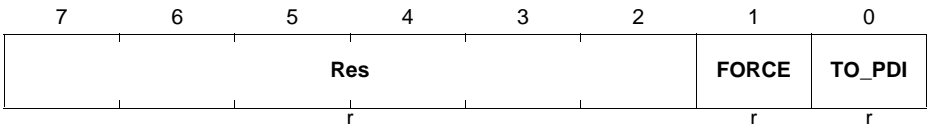
The EEP_CONF register is typically used to decide if the EtherCAT slave controller CPU gets access to the EEPROM.

ECAT0_EEP_CONF

EEPROM Configuration

(0500_H)

Reset Value: 00_H



Field	Bits	Type	Description
TO_PDI	0	r	EEPROM control is offered to PDI 0 _B No 1 _B Yes (PDI has EEPROM control)
FORCE	1	r	Force ECAT access 0 _B Do not change Bit 501.0 1 _B Reset Bit 501.0 to 0

EtherCAT Slave Controller (ECAT)

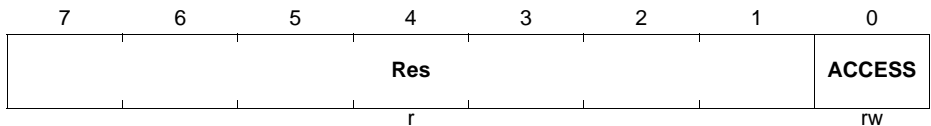
Field	Bits	Type	Description
Res	[7:2]	r	Reserved Read as 0; should be written with 0.

Register EEPROM PDI Access State (EEP_STATE)

The EEP_STATE register is used by the PDI to take over EEPROM access.

ECAT0_EEP_STATE

EEPROM PDI Access State (0501_H) **Reset Value: 00_H**



Field	Bits	Type	Description
ACCESS	0	rw	Access to EEPROM 0 _B PDI releases EEPROM access 1 _B PDI takes EEPROM access (PDI has EEPROM control)
Res	[7:1]	r	Reserved Read as 0; should be written with 0.

Note: r/(w): write access is only possible if .TO_PDI =1 and .FORCE =0.

EtherCAT Slave Controller (ECAT)

Register EEPROM Control/Status (EEP_CONT_STAT)

The EEP_CONT_STAT register is used to handle the EEPROM operation triggered by the ECAT master, show status of the EEPROM process.

ECAT0_EEP_CONT_STAT

EEPROM Control/Status

(0502_H)

Reset Value: 9460_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUS Y	ERR OR WE	ERR OR AC	L ST AT	ERR OR	CMD_REG		ALG	BYT ES	EMU L	Res				W_E N	
rwh	r	rw	r	r	rw		r	r	r	r				r	

Field	Bits	Type	Description
W_EN	0	r	ECAT write enable¹⁾ 0 _B Write requests are disabled 1 _B Write requests are enabled This bit is always 1 if PDI has EEPROM control.
Res	[4:1]	r	Reserved Read as 0; should be written with 0.
EMUL	5	r	EEPROM emulation 0 _B Normal operation (I ² C interface used) 1 _B PDI emulates EEPROM (I ² C not used)
BYTES	6	r	Supported number of EEPROM read bytes 0 _B 4 Bytes 1 _B 8 Bytes
ALG	7	r	Selected EEPROM Algorithm 0 _B 1 address byte (1 KBit – 16 KBit EEPROMs) 1 _B 2 address bytes (32 KBit – 4 MBit EEPROMs)
CMD_REG	[10:8]	rw	Command register¹⁾ Write: Initiate command. Read: Currently executed command Commands: 000 _B No command/EEPROM idle (clear error bits) 001 _B Read 010 _B Write 100 _B Reload Others: Reserved/invalid commands (do not issue) EEPROM emulation only: after execution, PDI writes command value to indicate operation is ready.

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
ERROR	11	r	Checksum Error at in ESC Configuration Area 0 _B Checksum OK 1 _B Checksum error
L_STAT	12	r	EEPROM loading status 0 _B EEPROM loaded, device information OK 1 _B EEPROM not loaded, device information not available (EEPROM loading in progress or finished with a failure)
ERROR_AC	13	rw	Error Acknowledge/Command²⁾ 0 _B No error 1 _B Missing EEPROM acknowledge or invalid command EEPROM emulation only: PDI writes 1 if a temporary failure has occurred.
ERROR_WE	14	r	Error Write Enable²⁾ 0 _B No error 1 _B Write Command without Write enable
BUSY	15	rwh	Busy 0 _B EEPROM Interface is idle 1 _B EEPROM Interface is busy

1) Write Enable bit 0 is self-clearing at the SOF of the next frame, Command bits [10:8] are self-clearing after the command is executed (EEPROM Busy ends). Writing "000" to the command register will also clear the error bits [14:13]. Command bits [10:8] are ignored if Error Acknowledge/Command is pending (bit 13).

2) Error bits are cleared by writing "000" (or any valid command) to Command Register Bits [10:8].

Notes

- r(w): write access depends upon the assignment of the EEPROM interface (ECAT/PDI). Write access is generally blocked if EEPROM interface is busy (.BUSY = 1).*
- r/[w]: EEPROM emulation only: write access is possible if EEPROM interface is busy (.BUSY = 1). PDI acknowledges pending commands by writing a 1 into the corresponding command register bits (.CMD_REG). Errors can be indicated by writing a 1 into the error bit .ERROR_AC. Acknowledging clears AL Event Request [5].*

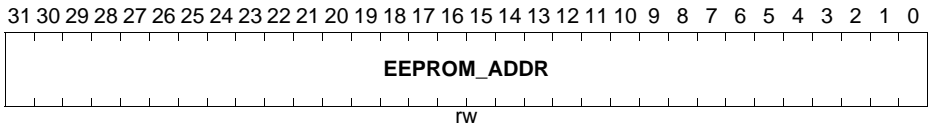
EtherCAT Slave Controller (ECAT)

Register EEPROM Address (EEP_ADR)

The EEPROM_ADR register defines the current used EEPROM address.

ECAT0_EEP_ADR

EEPROM Address (0504_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
EEPROM_ADDR	[31:0]	rw	EEPROM Address 0 _B First word (= 16 bits) 1 _B Second word ... Actually used EEPROM Address bits: [9:0]: EEPROM size up to 16 kBit [17:0]: EEPROM size 32 kBit – 4 Mbit [32:0]: EEPROM Emulation

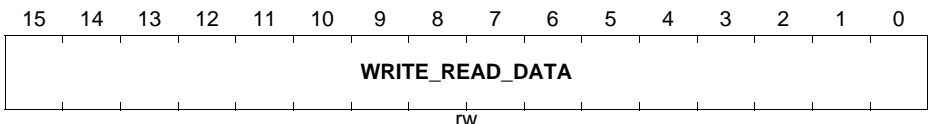
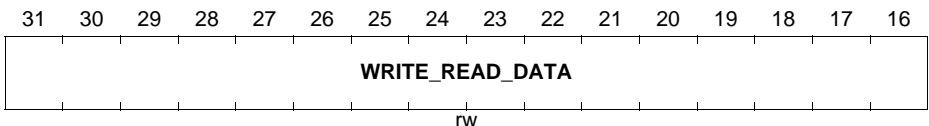
Note: r/(w): write access depends upon the assignment of the EEPROM interface (ECAT/PDI). Write access is generally blocked if EEPROM interface is busy (.BUSY = 1).

Register EEPROM Data (EEP_DATA)

The EEPROM_DATA register is used to load and store data on the EEPROM.

ECAT0_EEP_DATAx (x = 0-1)

EEPROM Data x (0508H + x *4) Reset Value: 0000 0000_H



Field	Bits	Type	Description
WRITE_READ_DATA	[31:0]	rw	EEPROM Read/Write data data to be written to EEPROM or data read from EEPROM

Note: r/(w): write access to this register depends upon the assignment of the EEPROM interface (ECAT/PDI). Write access is generally blocked if EEPROM interface is busy (.BUSY = 1).

EEPROM emulation with IP Core:

Note: r/[w]: write access for EEPROM emulation if read or reload command is pending. See the following information for further details:

Write access to EEPROM Data register 0x0508:0x050F is possible if EEPROM interface is busy (0x0502.15=1). PDI places EEPROM read data in this register before the pending EEPROM Read command is acknowledged (writing to 0x0502[10:8]). For Reload command: place the following information in the EEPROM Data register before acknowledging the command. This data is automatically transferred to the designated registers when the Reload command is acknowledged.

15.15.11 MII Management Interface

MII Management Interface (0510H:0515H)

Table 15-10 MII Management Interface Register Overview

Register Address	Length (Byte)	Description
0510H:0511H	2	MII Management Control/Status
0512H	1	PHY Address
0513H	1	PHY Register Address
0514H:0515H	2	PHY Data
0516H	1	MII Management ECAT Access State
0517H	1	MII Management PDI Access State
0518H:051BH	4	PHY Port Status

PDI controls the MII management interface if MII Management PDI Access register 0x0517.0=1, otherwise EtherCAT controls the MII management interface. Exception for ET1100: PDI controls the MII management interface if Transparent Mode is enabled.

Register MII Management Control/Status (MII_CONT_STAT)

The MII_CONT_STAT register is used to control the MI PHY interface.

ECAT0_MII_CONT_STAT

MII Management Control/Status (0510_H) **Reset Value: XXXX_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUS Y	ERR OR	Res			CMD_REG		PHY_ADDR				MI_L D	MIC_PDI	W_EN		
r	r	r			rw		r				r	r	r		

Field	Bits	Type	Description
W_EN	0	r	Write enable¹⁾ 0 _B Write disabled 1 _B Write enabled This bit is always 1 if PDI has MI control.

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
MIC_PDI	1	r	Management Interface can be controlled by PDI Registers 0x0516-0 x0517 0 _B Only ECAT control 1 _B PDI control possible
MI_LD	2	r	MI link detection Registers 0x0516-0 x0517 0 _B Not available 1 _B MI link detection active
PHY_ADDR	[7:3]	r	PHY address of port 0 PHY address of port 0-3 depending on .7
CMD_REG	[9:8]	rw	Command register Write: Initiate command. Read: Currently executed command Commands: 00 _B No command/MII idle (clear error bits) 01 _B Read 10 _B Write 11 _B Reserved/invalid commands (do not issue)
Res	[13:10]	r	Reserved Read as 0; should be written with 0.
ERROR	14	r	Command error 0 _B Last Command was successful 1 _B Invalid command or write command without Write Enable Cleared with a valid command or by writing "00" to Command register bits [9:8].
BUSY	15	r	Busy 0 _B MI control state machine is idle 1 _B MI control state machine is active

1) Write enable bit 0 is self-clearing at the SOF of the next frame (or at the end of the PDI access). Command bits [9:8] are self-clearing after the command is executed (Busy ends). Writing "00" to the command register will also clear the error bits [14:13]. The Command bits are cleared after the command is executed.

Note: r/ (w): write access depends on assignment of MI (ECAT/PDI). Write access is generally blocked if Management interface is busy (.BUSY = 1.

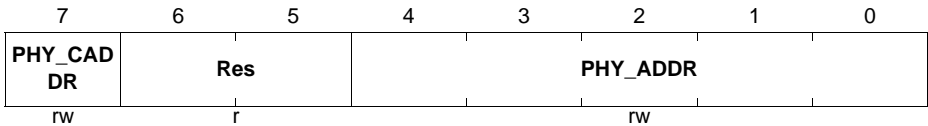
EtherCAT Slave Controller (ECAT)

Register MII PHY Address (MII_PHY_ADR)

The MII_PHY_ADR register selects the PHY which is intended to be read/write.

ECAT0_MII_PHY_ADR

PHY Address (0512_H) **Reset Value: 00_H**



Field	Bits	Type	Description
PHY_ADDR	[4:0]	rw	PHY Address
Res	[6:5]	r	Reserved Read as 0; should be written with 0.
PHY_CADDR	7	rw	Show configured PHY address of port 0-3 in register [7:3]. Select port x with bits [4:0] of this register (valid values are 0-3) 0 _B Show address of port 0 (offset) 1 _B Show individual address of port x

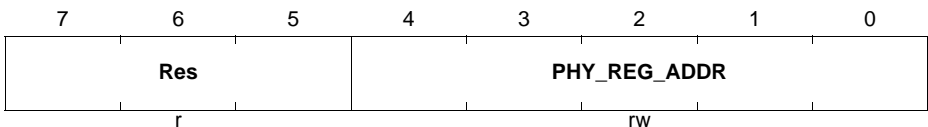
Note: r/ (w): write access depends on assignment of MI (ECAT/PDI). Write access is generally blocked if Management interface is busy (.BUSY = 1).

Register MII PHY Register Address (MII_PHY_REG_ADR)

The MII_PHY_REG_ADR register is used to select the register of the PHY for read/write.

ECAT0_MII_PHY_REG_ADR

PHY Register Address (0513_H) **Reset Value: 00_H**



Field	Bits	Type	Description
PHY_REG_ADDR	[4:0]	rw	Address of PHY Register that shall be read/written

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
Res	[7:5]	r	Reserved Read as 0; should be written with 0.

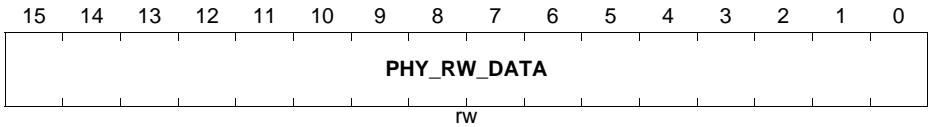
Note: rr/ (w): write access depends on assignment of MI (ECAT/PDI). Write access is generally blocked if Management interface is busy (.BUSY = 1).

Register MII PHY Data (MII_PHY_DATA)

The PHY_DATA register handles the read/write data.

ECAT0_MII_PHY_DATA

PHY Data (0514_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
PHY_RW_DATA	[15:0]	rw	PHY Read/Write Data

Note: r/ (w): write access depends on assignment of MI (ECAT/PDI). Access is generally blocked if Management interface is busy (.BUSY = 1).

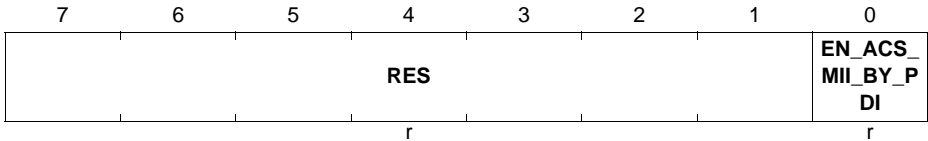
EtherCAT Slave Controller (ECAT)

Register MII Management ECAT Access State (MII_ECAC_ACS_STATE)

The MII_PDI_ACS_STATE register defines the access state to the MII management for the PDI interface.

ECAT0_MII_ECAC_ACS_STATE

MII ECAT ACS STATE (0516_H) **Reset Value: 00_H**



Field	Bits	Type	Description
EN_ACS_MII_BY_PDI	0	r	Access to Mii management 0 _B ECAT enables PDI takeover of MII management control 1 _B ECAT claims exclusive access to MII management
RES	[7:1]	r	Reserved, write 0

EtherCAT Slave Controller (ECAT)

Register MII Management PDI Access State (MII_PDI_ACS_STATE)

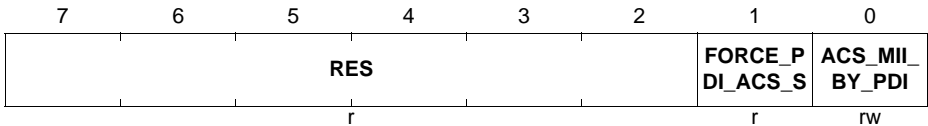
The MII_PDI_ACS_STATE register defines the access state to the MII management for the PDI interface.

ECAT0_MII_PDI_ACS_STATE

MII PDI ACS STATE

(0517_H)

Reset Value: 00_H



Field	Bits	Type	Description
ACS_MII_BY_PDI	0	rw	Access to MII management 0 _B ECAT has access to MII management 1 _B PDI has access to MII management
FORCE_PDI_ACS_S	1	r	Force PDI Access State by ECAT master 0 _B no change 1 _B Reset Bit ACS_MII_BY_PDI
RES	[7:2]	r	Reserved, write 0

Note: r/ (w): assigning access to PDI (bit 0 = 1) is only possible if 0x0516.0=0 and 0x0517.1=0, and if the SII EEPROM is loaded (0x0110[0]=1).

15.15.12 FMMU

FMMU (0600H:06FFH)

Each FMMU entry is described in 16 Bytes from 0600H:060FH to 06F0H:06FFH. y is the FMMU index (y=0 to 15).

Table 15-11 Table 78: FMMU Register overview

Register Address Offset	Length (Byte)	Description
+0x0:0x3	4	Logical Start Address
+0x4:0x5	2	Length
+0x6	1	Logical Start bit
+0x7	1	Logical Stop bit
+0x8:0x9	2	Physical Start Address

EtherCAT Slave Controller (ECAT)

Table 15-11 Table 78: FMMU Register overview (cont'd)

Register Address Offset	Length (Byte)	Description
+0xA	1	Physical Start bit
+0xB	1	Type
+0xC	1	Activate
+0xD:0xF	3	Reserved

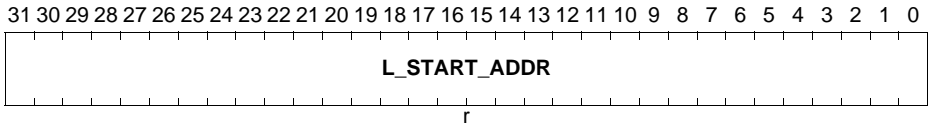
EtherCAT Slave Controller (ECAT)

Register Logical Start address FMMU y

The FMMU_L_START_ADR_y register is used to setup the logical start address of the FMMU_y within the EtherCAT Address Space.

ECAT0_FMMU_L_START_ADR_y (y = 0-7)

Logical Start address FMMU (0600_H+y*10H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
L_START_ADDR R	[31:0]	r	Logical start address within the EtherCAT Address Space

EtherCAT Slave Controller (ECAT)

Register Length FMMU y

The FMMU_LEN_y register is used to setup the Offset from the first logical FMMU Byte to the last FMMU Byte + 1.

ECAT0_FMMU_LEN_y (y = 0-7)

Length FMMU y (0604_H+y*10_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
OFFSET	[15:0]	r	Offset from the first logical FMMU Byte to the last FMMU Byte + 1 e.g., if two bytes are used then this parameter shall contain 2

Register Start bit FMMU y in logical address space

The FMMU_L_START_BIT_y register is used to setup the Logical starting bit that shall be mapped.

ECAT0_FMMU_L_START_BIT_y (y = 0-7)

Start bit FMMU y in logical address space (0606_H+y*10_H) **Reset Value: 00_H**



Field	Bits	Type	Description
L_START_BIT	[2:0]	r	Logical starting bit that shall be mapped bits are counted from least significant bit (=0) to most significant bit(=7)
Res	[7:3]	r	Reserved Read as 0; should be written with 0.

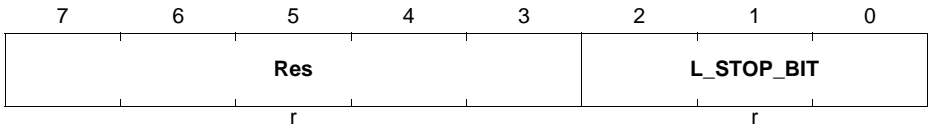
EtherCAT Slave Controller (ECAT)

Register Stop bit FMMU y in logical address space

The FMMU_L_STOP_BITy register is used to setup the last logical bit that shall be mapped.

ECAT0_FMMU_L_STOP_BITy (y = 0-7)

Stop bit FMMU y in logical address space (0607_H+y*10_H) **Reset Value: 00_H**



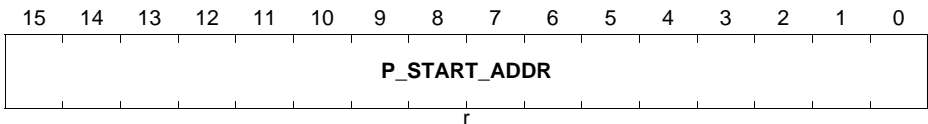
Field	Bits	Type	Description
L_STOP_BIT	[2:0]	r	Last logical bit that shall be mapped bits are counted from least significant bit (=0) to most significant bit(=7)
Res	[7:3]	r	Reserved Read as 0; should be written with 0.

Register Physical Start address FMMU y

The FMMU_P_START_ADDRy register is used to setup the physical start address.

ECAT0_FMMU_P_START_ADDRy (y = 0-7)

Physical Start address FMMU y (0608_H+y*10_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
P_START_ADDR	[15:0]	r	Physical Start Address mapped to logical Start address

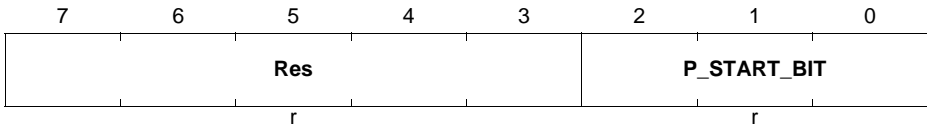
Register Physical Start bit FMMU y

The FMMU_P_START_BITy register is used to setup the Physical starting bit as target of logical start bit mapping.

ECAT0_FMMU_P_START_BITy (y = 0-7)

Physical Start bit FMMU y (060A_H+y*10H)

Reset Value: 00_H



Field	Bits	Type	Description
P_START_BIT	[2:0]	r	Physical starting bit as target of logical start bit mapping bits are counted from least significant bit (=0) to most significant bit(=7)
Res	[7:3]	r	Reserved Read as 0; should be written with 0.

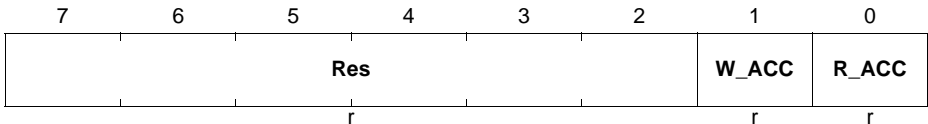
EtherCAT Slave Controller (ECAT)

Register Type FMMU y

The FMMU_TYPEy register is used to select if the mapping is used for read/write access.

ECAT0_FMMU_TYPEy (y = 0-7)

Type FMMU y (060B_H+y*10H) Reset Value: 00_H



Field	Bits	Type	Description
R_ACC	0	r	Read Access 0 _B Ignore mapping for read accesses 1 _B Use mapping for read accesses
W_ACC	1	r	Write Access 0 _B Ignore mapping for write accesses 1 _B Use mapping for write accesses
Res	[7:2]	r	Reserved Read as 0; should be written with 0.

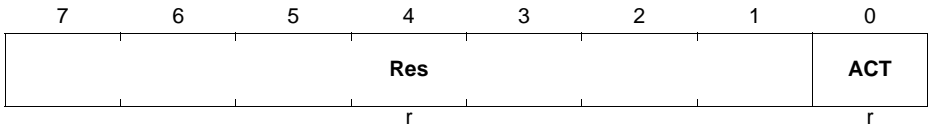
EtherCAT Slave Controller (ECAT)

Register Activate FMMU y

The FMMU_ACTIVATEy register shows if the FMMU is enabled by the EtherCAT master.

ECAT0_FMMU_ACTy (y = 0-7)

Activate FMMU y (060C_H+y*10H) **Reset Value: 00_H**



Field	Bits	Type	Description
ACT	0	r	FMMU Activation 0 _B FMMU deactivated. 1 _B FMMU activated. FMMU checks logical addressed blocks to be mapped according to mapping configured
Res	[7:1]	r	Reserved Read as 0; should be written with 0.

15.15.13 SyncManager

SyncManager (0x0800:0x087F)

SyncManager registers are mapped from 0x0800:0x0807 to 0x0818:0x087F. x specifies SyncManager (x=0 to 15).

Table 15-12 SyncManager Register overview

Register Address Offset	Length (Byte)	Description
+0x0:0x1	2	Physical Start Address
+0x2:0x3	2	Length
+0x4	1	Control Register
+0x5	1	Status Register
+0x6	1	Activate
+0x7	1	PDI Control

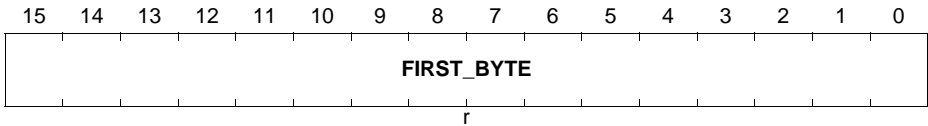
EtherCAT Slave Controller (ECAT)

Register physical Start Address SyncManager x

The SM_P_START_ADRx register is used to setup the first byte that will be handled by SyncManager.

ECAT0_SM_P_START_ADRx (x = 0-7)

Physical Start Address SyncManager x (0800_H+x*8_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
FIRST_BYTE	[15:0]	r	Specifies first byte that will be handled by SyncManager

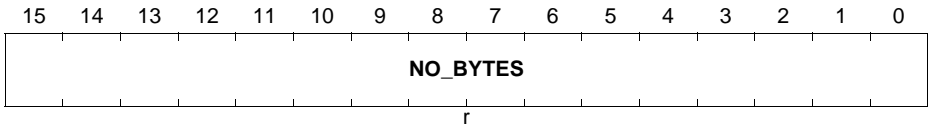
EtherCAT Slave Controller (ECAT)

Register Length SyncManager x

The SM_LENx register is used to setup the number of bytes assigned to the SyncManager.

ECAT0_SM_LENx (x = 0-7)

Length SyncManager x (0802_H+x*8_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
NO_BYTES	[15:0]	r	Number of bytes assigned to SyncManager shall be greater 1, otherwise SyncManager is not activated. If set to 1, only Watchdog Trigger is generated if configured

EtherCAT Slave Controller (ECAT)

Register Control Register SyncManager x

The SM_CONTROLx register is used to configure the SyncManager.

ECAT0_SM_CONTROLx (x = 0-7)

Control Register SyncManager x (0804_H+x*8_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
Res	WD_TRG	INT_PDI	INT_ECAT	DIR		OP_MODE	
r	r	r	r	r		r	

Field	Bits	Type	Description
OP_MODE	[1:0]	r	Operation Mode 00 _B Buffered (3 buffer mode) 01 _B Reserved 10 _B Mailbox (Single buffer mode) 11 _B Reserved
DIR	[3:2]	r	Direction 00 _B Read: ECAT read access, PDI write access 01 _B Write: ECAT write access, PDI read access 10 _B Reserved 11 _B Reserved
INT_ECAT	4	r	Interrupt in ECAT Event Request Register 0 _B Disabled 1 _B Enabled
INT_PDI	5	r	Interrupt in PDI Event Request Register 0 _B Disabled 1 _B Enabled
WD_TRG	6	r	Watchdog Trigger Enable 0 _B Disabled 1 _B Enabled
Res	7	r	Reserved Read as 0; should be written with 0.

Register Status Register SyncManager x

The SM_STATUSx register show the status of the SyncManager.

ECAT0_SM_STATUSx (x = 0-7)

Status Register SyncManager x (0805_H+x*8_H)

Reset Value: 30_H

7	6	5	4	3	2	1	0
W_BUF_IU	R_BUF_IU	BUF_STATUS		MB_STAT US	Res	INT_R	INT_W
r	r			r	r	r	r

Field	Bits	Type	Description
INT_W	0	r	<p>Interrupt Write</p> <p>0_B Interrupt cleared after first byte of buffer was read</p> <p>1_B Interrupt after buffer was completely and successfully written</p> <p><i>Note: This interrupt is signaled to the reading side if enabled in the SM Control register.</i></p>
INT_R	1	r	<p>Interrupt Read</p> <p>0_B Interrupt cleared after first byte of buffer was written</p> <p>1_B Interrupt after buffer was completely and successful read</p> <p><i>Note: This interrupt is signaled to the writing side if enabled in the SM Control register.</i></p>
Res	2	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>
MB_STATUS	3	r	<p>Mailbox mode: mailbox status</p> <p>0_B Mailbox empty</p> <p>1_B Mailbox full</p> <p>Buffered mode: reserved</p>
BUF_STATUS	[5:4]	r	<p>Buffered mode: buffer status (last written buffer)</p> <p>00_B 1. buffer</p> <p>01_B 2. buffer</p> <p>10_B 3. buffer</p> <p>11_B (no buffer written)</p> <p>Mailbox mode: reserved</p>

EtherCAT Slave Controller (ECAT)

Field	Bits	Type	Description
R_BUF_IU	6	r	Read buffer in use (opened) 0 _B buffer not in use 1 _B buffer in use
W_BUF_IU	7	r	Write buffer in use (opened) 0 _B buffer not in use 1 _B buffer in use

EtherCAT Slave Controller (ECAT)

Register Activate SyncManager x

The SM_ACTIVATEx register show the activated state of the SyncManager.

ECAT0_SM_ACTx (x = 0-7)

Activate SyncManager x (0806_H+x*8_H) **Reset Value: 00_H**

7	6	5	4	3	2	1	0
LE_PDI	LE_ECAC	Res				REP_REQ	SM_EN
r	r	r				r	r

Field	Bits	Type	Description
SM_EN	0	r	SyncManager Enable/Disable 0 _B Disable: Access to Memory without SyncManager control 1 _B Enable: SyncManager is active and controls Memory area set in configuration
REP_REQ	1	r	Repeat Request A toggle of Repeat Request means that a mailbox retry is needed (primarily used in conjunction with ECAT Read Mailbox
Res	[5:2]	r	Reserved Read as 0; should be written with 0.
LE_ECAC	6	r	LatchEvent ECAT 0 _B No 1 _B Generate Latch event if EtherCAT master issues a buffer exchange
LE_PDI	7	r	LatchEvent PDI 0 _B No 1 _B Generate Latch events if PDI issues a buffer exchange or if PDI accesses buffer start address

PDI register function acknowledge by Write command is disabled: Reading this register from PDI in all SyncManagers which have changed activation clears AL Event Request 0x0220[4]. Writing to this register from PDI is not possible.

PDI register function acknowledge by Write command is enabled: Writing this register from PDI in all SyncManagers which have changed activation clears AL Event Request 0x0220[4]. Writing to this register from PDI is possible; write value is ignored (write 0).

EtherCAT Slave Controller (ECAT)

Register PDI Control SyncManager x

The SM_PDI_CTRx register is used for the PDI to control the SyncManager.

ECAT0_SM_PDI_CTRx (x = 0-7)

PDI Control SyncManager x (0807_H+x*8_H) **Reset Value: 00_H**

	7	6	5	4	3	2	1	0
	Res						REP_ACK	DEACT
	r						rw	rw

Field	Bits	Type	Description
DEACT	0	rw	<p>Deactivate SyncManager</p> <p>0_B Read 0 for Normal operation, SyncManager activated, write 0 for Activate SyncManager</p> <p>1_B Read 1 for SyncManager deactivated and reset SyncManager locks access to Memory area, write 1 for Request SyncManager deactivation</p> <p><i>Note: Writing 1 is delayed until the end of a frame which is currently processed.</i></p>
REP_ACK	1	rw	<p>Repeat Ack</p> <p>If this is set to the same value as set by Repeat Request, the PDI acknowledges the execution of a previous set Repeat request.</p>
Res	[7:2]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

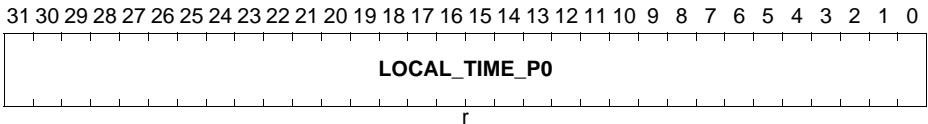
15.15.14 DC – Receive Times

Register Receive Time Port 0 (DC_RCV_TIME_PORT0)

The DC_RCV_TIME_PORT register is used to latch the local time of the beginning of the last receive frame.

ECAT0_DC_RCV_TIME_PORT0

Receive Time Port 0 (0900_H) Reset Value: XXXX XXXX_H



Field	Bits	Type	Description
LOCAL_TIME_P0	[31:0]	r	<p>Write by EtherCAT master A write access to register 0x0900 with BWR or FPWR latches the local time of the beginning of the receive frame (start first bit of preamble) at each port.</p> <p>Read Local time of the beginning of the last receive frame containing a write access to this register.</p>

Note: The time stamps cannot be read in the same frame in which this register was written.

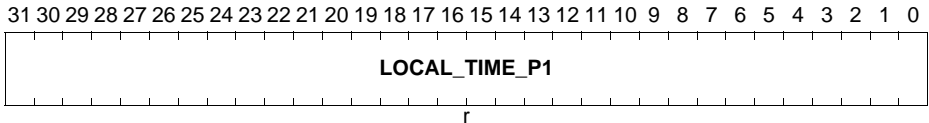
EtherCAT Slave Controller (ECAT)

Register Receive Time Port 1 (DC_RCV_TIME_PORT1)

The DC_RCV_TIME_PORT register show the Local time of the beginning of a frame received at port 1.

ECAT0_DC_RCV_TIME_PORT1

Receive Time Port 1 (0904_H) Reset Value: XXXX XXXX_H



Field	Bits	Type	Description
LOCAL_TIME_P1	[31:0]	r	Local time of the beginning of a frame (start first bit of preamble) received at port 1 containing a BWR or FPWR to Register 0x0900.

15.15.15 Time Loop Control Unit

Time Loop Control unit is usually assigned to ECAT. Write access to Time Loop Control registers by PDI (and not ECAT) is only possible with explicit IP Core configuration.

Register Receive Time ECAT Processing Unit (RECEIVE_TIME_PU)

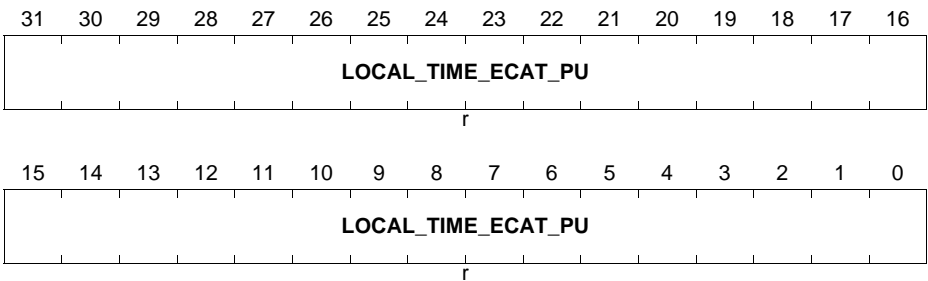
The xxx register is used show the Local time of the beginning of a frame received at the ECAT Processing Unit.

ECAT0_RECEIVE_TIME_PUx (x = 0-1)

Receive Time PU x

(918H + x *4)

Reset Value: XXXX XXXX_H



Field	Bits	Type	Description
LOCAL_TIME_ECAT_PU	[31:0]	r	Local time of the beginning of a frame (start first bit of preamble) received at the ECAT Processing Unit containing a write access to Register 0x0900

Register System Time - Read

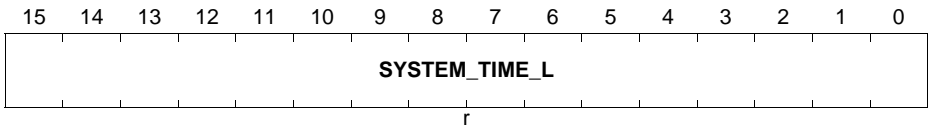
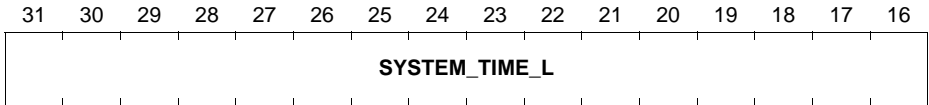
The DC_SYS_TIME register is used as a local copy of the actual system time latched by reading the first byte.

ECAT0_DC_SYS_TIME_x (x = 0-1)

System Time [READ Mode] x

(0910H + x *4)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SYSTEM_TIM E_L	[31:0]	r	System Time read access Local copy of the System Time. Time latched when reading first byte of this register.

Note: E.g., if port 0 is open, this registers reflects the Receive Time Port 0 as a 64-bit value.

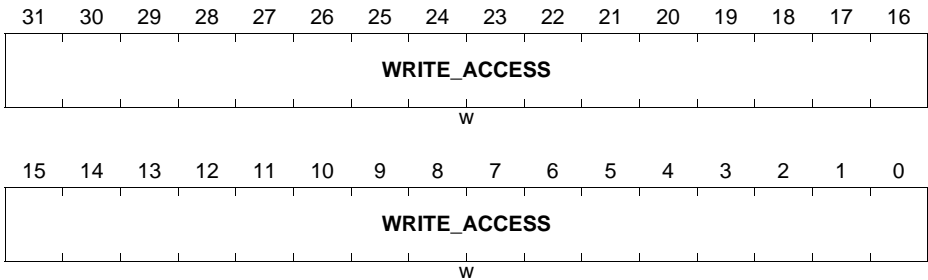
EtherCAT Slave Controller (ECAT)

Register System Time - Write

The DC_SYS_TIME register is used to compared with Latch0 Time Positive Edge time.

ECAT0_DC_SYS_TIME

System Time [WRITE Mode] (0910_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
WRITE_ACCESS	[31:0]	w	<p>Write access</p> <p>Written value will be compared with Latch0 Time Positive Edge time. The result is an input to the time control loop.</p> <p><i>Note: written value will be compared at the end of the access with Latch0 Time Positive Edge (0x09B0:0x09B3) if at least the last byte (0x0913) was written.</i></p>

Notes

1. Write access to this register depends upon ESC configuration (typically ECAT, PDI only with explicit ESC configuration: System Time PDI controlled).
2. Register 0x0918:0x091F is described in the previous chapter.

Register System Time Offset (DC_SYS_TIME_OFFSET)

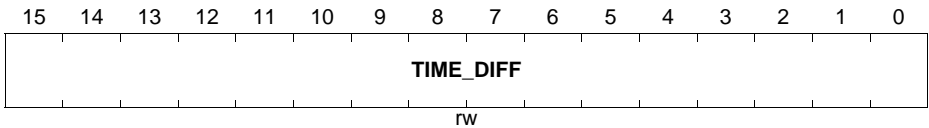
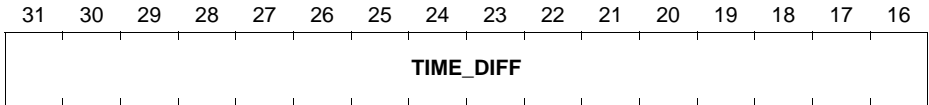
The DC_SYS_TIME_OFFSET register is used to setup the difference between local time and System Time.

ECAT0_DC_SYS_TIME_OFFSET_x (x = 0-1)

System Time Offset x

(920H + x *4)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TIME_DIFF	[31:0]	rw	Difference between local time and System Time Offset is added to the local time.

Note: Write access to this register depends upon ESC configuration (typically ECAT, PDI only with explicit ESC configuration: System Time PDI controlled). Reset internal system time difference filter and speed counter filter by writing Speed Counter Start (0x0930:0x0931) after changing this value.

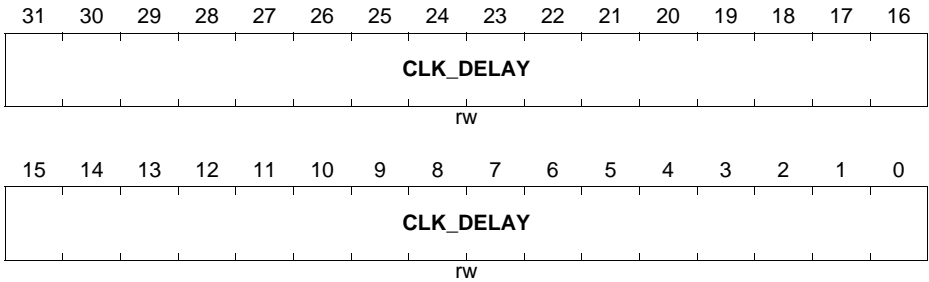
EtherCAT Slave Controller (ECAT)

Register System Time Delay (DC_SYS_TIME_DELAY)

The DC_SYS_TIME_DELAY register is used to setup the delay between Reference Clock and the ESC.

ECAT0_DC_SYS_TIME_DELAY

System Time Delay (0928_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
CLK_DELAY	[31:0]	rw	Delay between Reference Clock and the ESC

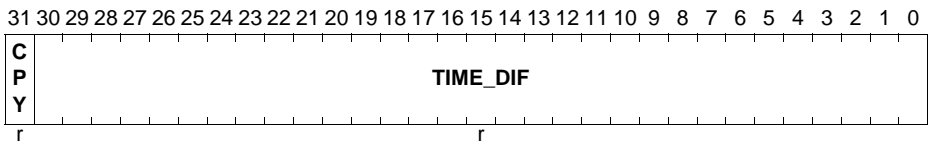
Note: Write access to this register depends upon ESC configuration (typically ECAT, PDI only with explicit ESC configuration: System Time PDI controlled). Reset internal system time difference filter and speed counter filter by writing Speed Counter Start (0x0930:0x0931) after changing this value.

Register System Time Difference (DC_SYS_TIME_DIFF)

The DC_SYS_TIME_DIFF register show the difference between local copy of System Time and received System Time values.

ECAT0_DC_SYS_TIME_DIFF

System Time Difference (092C_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
TIME_DIF	[30:0]	r	Mean difference between local copy of System Time and received System Time values
CPY	31	r	Local copy of System Time 0 _B Greater than or equal received System Time 1 _B Smaller than received System Time

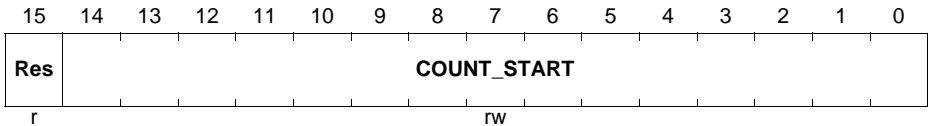
EtherCAT Slave Controller (ECAT)

Register Speed Counter Start (DC_SPEED_COUNT_START)

The DC_SPEED_COUNT_START register is used for Bandwidth for adjustment of local copy of System Time.

ECAT0_DC_SPEED_COUNT_START

Speed Counter Start (0930_H) **Reset Value: 1000_H**



Field	Bits	Type	Description
COUNT_START	[14:0]	rw	Bandwidth for adjustment of local copy of System Time (larger values → smaller bandwidth and smoother adjustment) A write access resets System Time Difference (0x092C:0x092F) and Speed Counter Diff (0x0932:0x0933). Minimum value: 0x0080 to 0x3FFF
Res	15	r	Reserved Read as 0; should be written with 0.

Note: Write access to this register depends upon ESC configuration (typically ECAT, PDI only with explicit ESC configuration: System Time PDI controlled).

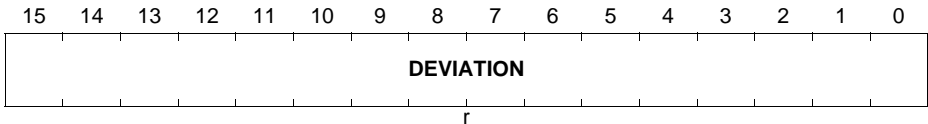
EtherCAT Slave Controller (ECAT)

Register Speed Counter Diff (DC_SPEED_COUNT_DIFF)

The DC_SPEED_COUNT_DIFF register is show the deviation between local clock period and Reference Clock's clock period.

ECAT0_DC_SPEED_COUNT_DIFF

Speed Counter Diff (0932_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
DEVIATION	[15:0]	r	Representation of the deviation between local clock period and Reference Clock's clock period (representation: two' s complement) Range: ± (Speed Counter Start – 0x7F)

Note: Calculate the clock deviation after System Time Difference has settled at a low value as follows:

$$\text{Deviation} = \frac{\text{Speed Counter Diff}}{5 * (\text{Speed Counter Start} + \text{Speed Counter Diff} + 2) * (\text{Speed Counter Start} - \text{Speed Counter Diff} + 2)}$$

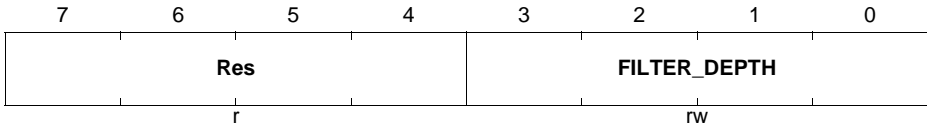
EtherCAT Slave Controller (ECAT)

Register Speed Counter Filter Depth (DC_SPEED_COUNT_FIL_DEPTH)

The DC_SPEED_COUNT_FIL_DEPTH register is used to setup the Filter depth for averaging the clock period deviation.

ECAT0_DC_SPEED_COUNT_FIL_DEPTH

Speed Counter Filter Depth (0935_H) **Reset Value: 0C_H**



Field	Bits	Type	Description
FILTER_DEPTH	[3:0]	rw	Filter depth for averaging the clock period deviation
Res	[7:4]	r	Reserved Read as 0; should be written with 0.

Notes

1. Write access to this register depends upon ESC configuration (typically ECAT, PDI only with explicit ESC configuration: System Time PDI controlled) .
2. Reset internal speed counter filter by writing Speed Counter Start (0x0930:0x0931) after changing this value.

EtherCAT Slave Controller (ECAT)

Register Cyclic Unit Control

The DC_CYC_CONT register show the control setup for the Sync and Latch Units.

ECAT0_DC_CYC_CONT

Cyclic Unit Control

(0980_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
Res	LATCH_U 1	LATCH_U 0			Res		SYNC
r	r	r			r		r

Field	Bits	Type	Description
SYNC	0	r	SYNC out unit control 0 _B ECAT controlled 1 _B PDI controlled
Res	[3:1]	r	Reserved Read as 0; should be written with 0.
LATCH_U0	4	r	Latch In unit 0 0 _B ECAT controlled 1 _B PDI controlled <i>Note: Always 1 (PDI controlled) if System Time is PDI controlled. Latch interrupt is routed to ECAT/PDI depending on this setting</i>
LATCH_U1	5	r	Latch In unit 1 0 _B ECAT controlled 1 _B PDI controlled <i>Note: Latch interrupt is routed to ECAT/PDI depending on this setting</i>
Res	[7:6]	r	Reserved Read as 0; should be written with 0.

EtherCAT Slave Controller (ECAT)

Register Activation (DC_ACT)

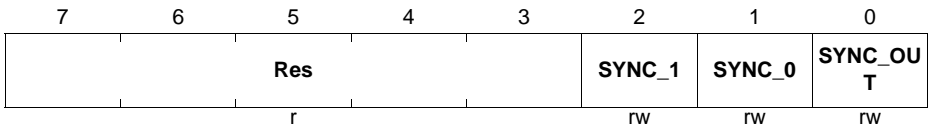
The DC_ACT register is used to setup the SYNC Unit.

ECAT0_DC_ACT

Activation register

(0981_H)

Reset Value: 00_H



Field	Bits	Type	Description
SYNC_OUT	0	rw	Sync Out Unit activation 0 _B Deactivated 1 _B Activated
SYNC_0	1	rw	SYNC0 generation 0 _B Deactivated 1 _B SYNC0 pulse is generated
SYNC_1	2	rw	SYNC1 generation 0 _B Deactivated 1 _B SYNC1 pulse is generated
Res	[7:3]	r	Reserved Read as 0; should be written with 0.

Note: Write to this register depends upon setting of .SYNC

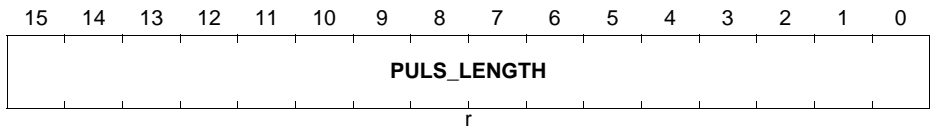
EtherCAT Slave Controller (ECAT)

Register Pulse Length of SyncSignals (DC_PULSE_LEN)

The DC_PULSE_LEN register is show the used Puls length of the SYNC signals. The value is configured by EEPROM

ECAT0_DC_PULSE_LEN

Pulse Length of SyncSignals (0982_H) **Reset Value: XXXX_H**



Field	Bits	Type	Description
PULS_LENGTH	[15:0]	r	Pulse length of SyncSignals (in Units of 10 ns) loaded by EEPROM ADR 0x0002 0 _B Acknowledge mode: SyncSignal will be cleared by reading SYNC[1:0] Status register

EtherCAT Slave Controller (ECAT)

Register Activation Status (DC_ACT_STAT)

The DC_SYNC0_STAT register shows the activation status.

ECAT0_DC_ACT_STAT

Activation Status

(0984_H)

Reset Value: 00_H

7	6	5	4	3	2	1	0
Res					S_TIME	S1_ACK_STATE	S0_ACK_STATE
r					r	r	r

Field	Bits	Type	Description
S0_ACK_STATE	0	r	SYNC0 activation state 0 _B First SYNC0 pulse is not pending 1 _B First SYNC0 pulse is pending
S1_ACK_STATE	1	r	SYNC1 activation state SYNC0 in Acknowledge mode is cleared by reading this register from PDI, use only in Acknowledge mode 0 _B First SYNC1 pulse is not pending 1 _B First SYNC1 pulse is pending
S_TIME	2	r	Start Time Cyclic Operation (0x0990:0x0997) plausibility check result when Sync Out Unit was activated 0 _B Start Time was within near future 1 _B Start Time was out of near future (0x0981.6)
Res	[7:3]	r	Reserved Read as 0; should be written with 0.

Notes

1. PDI register function acknowledge by Write command is disabled: Reading this register from PDI clears AL Event Request 0x0220[2]. Writing to this register from PDI is not possible.
2. PDI register function acknowledge by Write command is enabled: Writing this register from PDI clears AL Event Request 0x0220[2]. Writing to this register from PDI is possible; write value is ignored (write 0).

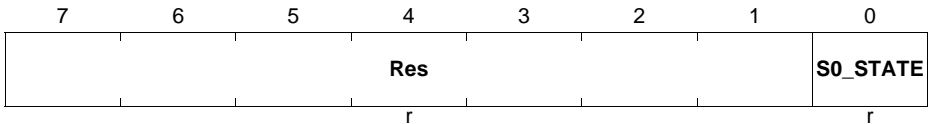
EtherCAT Slave Controller (ECAT)

Register SYNC0 Status (DC_SYNC0_STAT)

The DC_SYNC0_STAT register show the SYNC0 status.

ECAT0_DC_SYNC0_STAT

SYNC0 Status (098E_H) **Reset Value: 00_H**



Field	Bits	Type	Description
S0_STATE	0	r	SYNC0 state for Acknowledge mode SYNC0 in Acknowledge mode is cleared by reading this register from PDI, use only in Acknowledge mode
Res	[7:1]	r	Reserved Read as 0; should be written with 0.

Notes

1. *PDI register function acknowledge by Write command is disabled: Reading this register from PDI clears AL Event Request 0x0220[2]. Writing to this register from PDI is not possible.*
2. *PDI register function acknowledge by Write command is enabled: Writing this register from PDI clears AL Event Request 0x0220[2]. Writing to this register from PDI is possible; write value is ignored (write 0).*

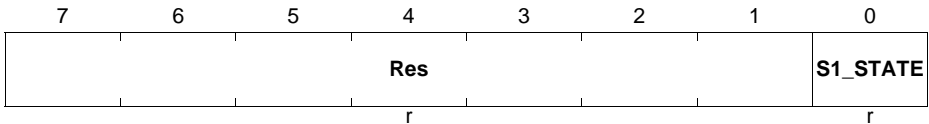
EtherCAT Slave Controller (ECAT)

Register SYNC1 Status (DC_SYNC1_STAT)

The DC_SYNC0_STAT register show the SYNC1 status.

ECAT0_DC_SYNC1_STAT

SYNC1 Status (098F_H) **Reset Value: 00_H**



Field	Bits	Type	Description
S1_STATE	0	r	SYNC1 state for Acknowledge mode SYNC1 in Acknowledge mode is cleared by reading this register from PDI, use only in Acknowledge mode
Res	[7:1]	r	Reserved Read as 0; should be written with 0.

Notes

1. *PDI register function acknowledge by Write command is disabled: Reading this register from PDI clears AL Event Request 0x0220[3]. Writing to this register from PDI is not possible.*
2. *PDI register function acknowledge by Write command is enabled: Writing this register from PDI clears AL Event Request 0x0220[3]. Writing to this register from PDI is possible; write value is ignored (write 0).*

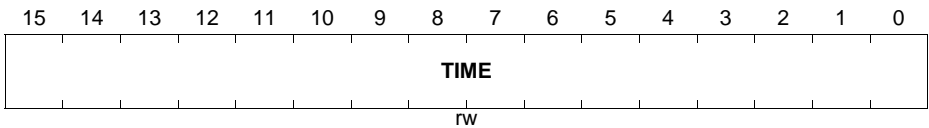
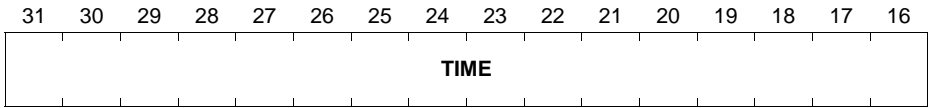
Register Start Time Cyclic Operation (DC_CYC_START_TIME)

The DC_CYC_START_TIME register is used to setup cyclic operations in relation to the system time.

EtherCAT Slave Controller (ECAT)

ECAT0_DC_CYC_START_TIME_x (x = 0-1)

Start Time Cyclic Operation x (0990H + x *4) Reset Value: 0000 0000_H



Field	Bits	Type	Description
TIME	[31:0]	rw	Write Start time (System time) of cyclic operation in ns. Read System time of next SYNC0 pulse in ns.

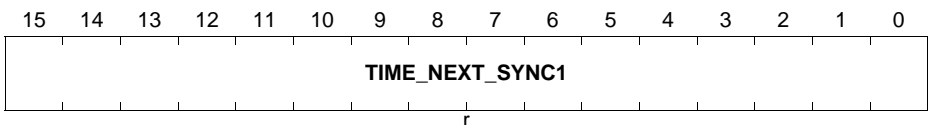
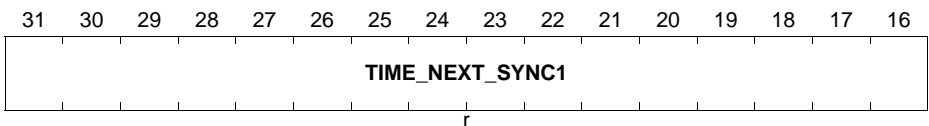
Note: Write to this register depends upon setting of 0x0980.0. Only writable if 0x0981.0=0. Auto-activation (0x0981.3=1): upper 32 bits are automatically extended if only lower 32 bits are written within one frame.

Register Next SYNC1 Pulse (DC_NEXT_SYNC1_PULSE)

The DC_NEXT_SYNC1_PULSE register show the system time for the next SYNC1 puls

ECAT0_DC_NEXT_SYNC1_PULSE_x (x = 0-1)

Next SYNC1 Pulse x (0998H + x *4) Reset Value: 0000 0000_H



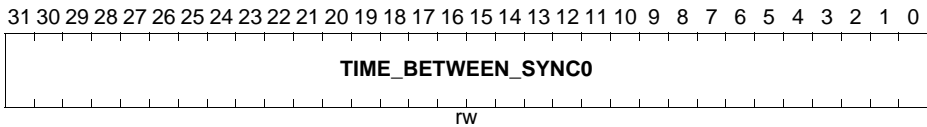
Field	Bits	Type	Description
TIME_NEXT_SYNC1	[31:0]	r	System time of next SYNC1 pulse in ns

Register SYNC0 Cycle Time (DC_SYNC0_CYC_TIME)

The DC_SYNC0_CYC_TIME register is to setup the Time between two consecutive SYNC0 pulses.

ECAT0_DC_SYNC0_CYC_TIME

SYNC0 Cycle Time (09A0_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
TIME_BETWEEN_SYNC0	[31:0]	rw	Time between two consecutive SYNC0 pulses in ns 0 _B Single shot mode, generate only one SYNC0 pulse

Note: Write to this register depends upon setting of 0x0980.0.

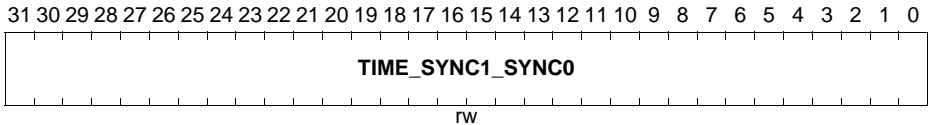
EtherCAT Slave Controller (ECAT)

Register SYNC1 Cycle Time (DC_SYNC1_CYC_TIME)

The DC_SYNC1_CYC_TIME register is used to setup the Time between SYNC1 pulses and SYNC0 pulse.

ECAT0_DC_SYNC1_CYC_TIME

SYNC1 Cycle Time (09A4_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TIME_SYNC1_SYNC0	[31:0]	rw	Time between SYNC1 pulses and SYNC0 pulse in ns

Note: Write to this register depends upon setting of 0x0980.0.

EtherCAT Slave Controller (ECAT)

Register Latch0 Control (DC_LATCH0_CONT)

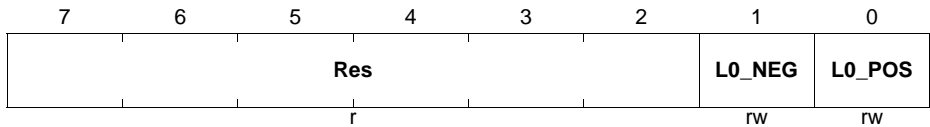
The DC_LATCH0_CONT register is used to setup LATCH0.

ECAT0_DC_LATCH0_CONT

Latch0 Control

(09A8_H)

Reset Value: 00_H



Field	Bits	Type	Description
L0_POS	0	rw	Latch0 positive edge 0 _B Continuous Latch active 1 _B Single event (only first event active)
L0_NEG	1	rw	Latch0 negative edge 0 _B Continuous Latch active 1 _B Single event (only first event active)
Res	[7:2]	r	Reserved Read as 0; should be written with 0.

Note: Write access depends upon setting of 0x0980.4

EtherCAT Slave Controller (ECAT)

Register Latch1 Control (DC_LATCH1_CONT)

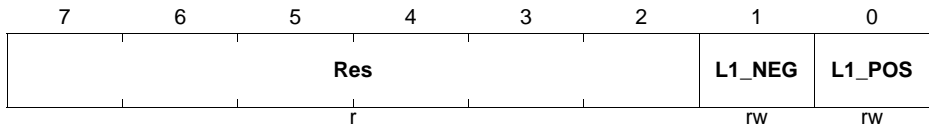
The DC_LATCH1_CONT register is used to setup LATCH1.

ECAT0_DC_LATCH1_CONT

Latch1 Control

(09A9_H)

Reset Value: 00_H



Field	Bits	Type	Description
L1_POS	0	rw	Latch1 positive edge 0 _B Continuous Latch active 1 _B Single event (only first event active)
L1_NEG	1	rw	Latch1 negative edge 0 _B Continuous Latch active 1 _B Single event (only first event active)
Res	[7:2]	r	Reserved Read as 0; should be written with 0.

Note: Write access depends upon setting of 0x0980.5

EtherCAT Slave Controller (ECAT)

Register Latch0 Status (DC_LATCH0_STAT)

The DC_LATCH0_STAT register show the status of LATCH0.

ECAT0_DC_LATCH0_STAT

Latch0 Status

(09AE_H)

Reset Value: XX_H

7	6	5	4	3	2	1	0
Res					L0_PIN	EV_L0_NEG	EV_L0_POS
r					r	r	r

Field	Bits	Type	Description
EV_L0_POS	0	r	Event Latch0 positive edge 0 _B Positive edge not detected or continuous mode 1 _B Positive edge detected in single event mode only Flag cleared by reading out Latch0 Time Positive Edge.
EV_L0_NEG	1	r	Event Latch0 negative edge 0 _B Negative edge not detected or continuous mode 1 _B Negative edge detected in single event mode only Flag cleared by reading out Latch0 Time Negative Edge.
L0_PIN	2	r	Latch0 pin state
Res	[7:3]	r	Reserved Read as 0; should be written with 0.

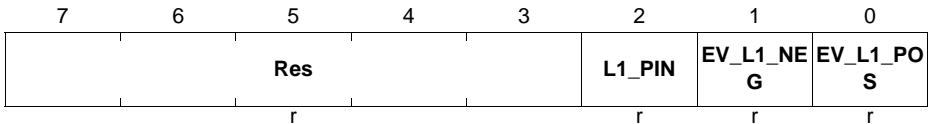
EtherCAT Slave Controller (ECAT)

Register Latch1 Status (DC_LATCH1_STAT)

The DC_LATCH1_STAT register is used show the status of LATCH1.

ECAT0_DC_LATCH1_STAT

Latch1 Status (09AF_H) **Reset Value: XX_H**



Field	Bits	Type	Description
EV_L1_POS	0	r	Event Latch1 positive edge 0 _B Positive edge not detected or continuous mode 1 _B Positive edge detected in single event mode only Flag cleared by reading out Latch1 Time Positive Edge.
EV_L1_NEG	1	r	Event Latch1 negative edge 0 _B Negative edge not detected or continuous mode 1 _B Negative edge detected in single event mode only Flag cleared by reading out Latch1 Time Negative Edge.
L1_PIN	2	r	Latch1 pin state
Res	[7:3]	r	Reserved Read as 0; should be written with 0.

Register Latch0 Time Positive Edge (DC_LATCH0_TIME_POS)

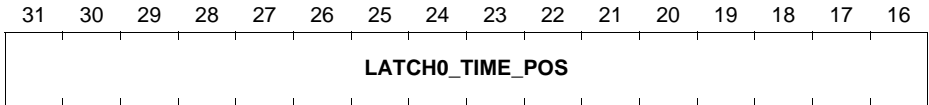
The DC_LATCH0_TIME_POS register show the captures System time at the positive edge of the Latch0 signal.

ECAT0_DC_LATCH0_TIME_POS_x (x = 0-1)

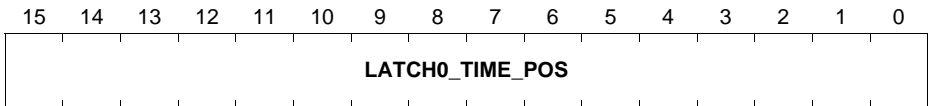
Latch0 Time Positive Edge x

(09B0_H + x *4)

Reset Value: 0000 0000_H



rw



rw

Field	Bits	Type	Description
LATCH0_TIM E_POS	[31:0]	rw	Register captures System time at the positive edge of the Latch0 signal

Note: Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value. Reading this register from ECAT clears Latch0 Status 0x09AE[0] if 0x0980.4=0. Writing to this register from ECAT is not possible.

Register Latch0 Time Negative Edge (DC_LATCH0_TIME_NEG)

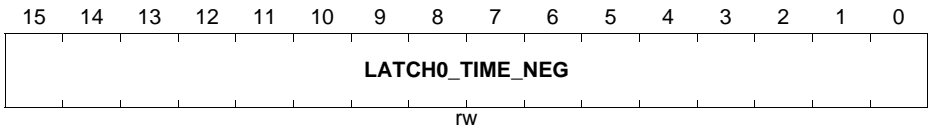
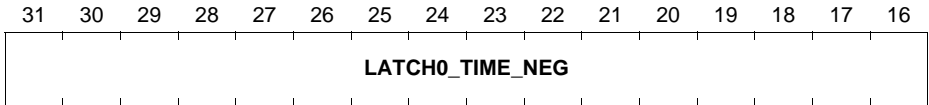
The DC_LATCH0_TIME_POS register show the captures System time at the negative edge of the Latch0 signal.

ECAT0_DC_LATCH0_TIME_NEG_x (x = 0-1)

Latch0 Time Negative Edge x

(09B8_H + x *4)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
LATCH0_TIME_NEG	[31:0]	rw	Register captures System time at the negative edge of the Latch0 signal

Note: Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value. Reading this register from ECAT clears Latch0 Status 0x09AE[0] if 0x0980.4=0. Writing to this register from ECAT is not possible.

Register Latch1 Time Positive Edge (DC_LATCH1_TIME_POS)

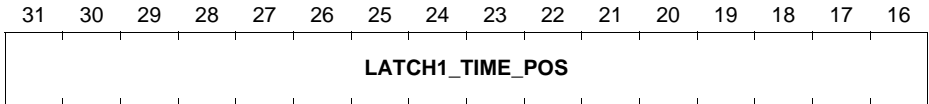
The DC_LATCH1_TIME_POS register show the captures System time at the positive edge of the Latch1 signal.

ECAT0_DC_LATCH1_TIME_POS_x (x = 0-1)

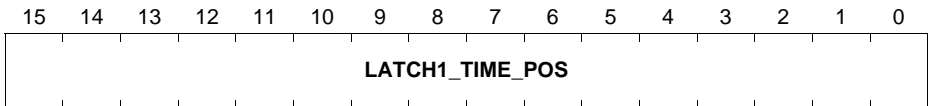
Latch1 Time Positive Edge x

(09C0_H + x *4)

Reset Value: 0000 0000_H



rw



rw

Field	Bits	Type	Description
LATCH1_TIM E_POS	[31:0]	rw	Register captures System time at the positive edge of the Latch1 signal

Note: Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value. Reading this register from ECAT clears Latch1 Status 0x09AF[0] if 0x0980.5=0. Writing to this register from ECAT is not possible.

Register Latch1 Time Negative Edge (DC_LATCH1_TIME_NEG)

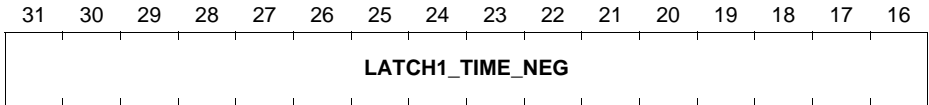
The DC_LATCH1_TIME_NEG register show the captures System time at the negative edge of the Latch1 signal.

ECAT0_DC_LATCH1_TIME_NEG_x (x = 0-1)

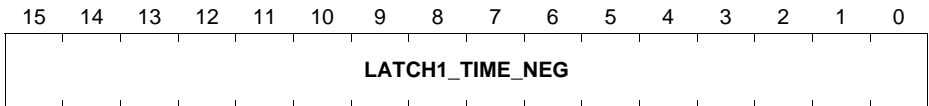
Latch1 Time Negative Edge x

(09C8_H + x *4)

Reset Value: 0000 0000_H



rw



rw

Field	Bits	Type	Description
LATCH1_TIM E_NEG	[31:0]	rw	Register captures System time at the negative edge of the Latch1 signal

Note: Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value. Reading this register from ECAT clears Latch1 Status 0x09AF[0] if 0x0980.5=0. Writing to this register from ECAT is not possible.

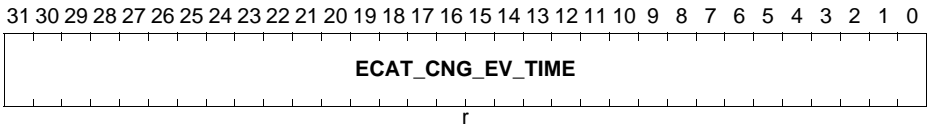
15.15.16 SyncManager Event Times

Register EtherCAT Buffer Change Event Time (DC_EC_CNG_EV_TIME)

The DC_EC_CNG_EV_TIME register show the captured local time of the beginning of the frame which causes at least one SyncManager to assert an ECAT event.

ECAT0_DC_EC_CNG_EV_TIME

EtherCAT Buffer Change Event Time (09F0_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ECAT_CNG_EV_TIME	[31:0]	r	Register captures local time of the beginning of the frame which causes at least one SyncManager to assert an ECAT event

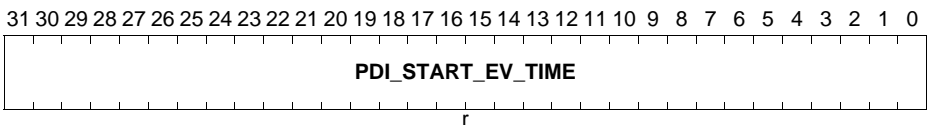
Note: Register bits [31:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value.

Register PDI Buffer Start Event Time (DC_PDI_START_EV_TIME)

The DC_PDI_START_EV_TIME register show the captured local time when at least one SyncManager asserts an PDI buffer start event.

ECAT0_DC_PDI_START_EV_TIME

PDI Buffer Start Event Time (09F8_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
PDI_START_EV_TIME	[31:0]	r	Register captures local time when at least one SyncManager asserts an PDI buffer start event

EtherCAT Slave Controller (ECAT)

Note: Register bits [31:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value.

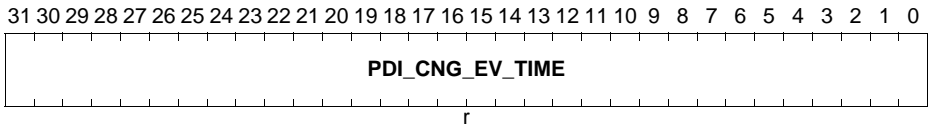
EtherCAT Slave Controller (ECAT)

Register PDI Buffer Change Event Time (DC_PDI_CNG_EV_TIME)

The DC_PDI_CNG_EV_TIME register show the captured local time when at least one SyncManager asserts an PDI buffer change event.

ECAT0_DC_PDI_CNG_EV_TIME

PDI Buffer Change Event Time (09FC_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
PDI_CNG_EV_TIME	[31:0]	r	Register captures local time when at least one SyncManager asserts an PDI buffer change event

Note: Register bits [31:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value.

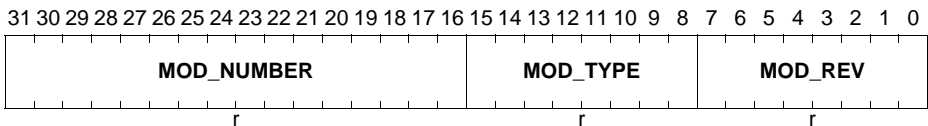
15.15.17 ESC specific registers (0x0E00:0x0E01)

Register Module Identification (ID)

The module identification registers indicate the function and the design step of the ECAT module.

ECAT0_ID

ECAT0 Module ID (0E00_H) **Reset Value: 0093C001_H**



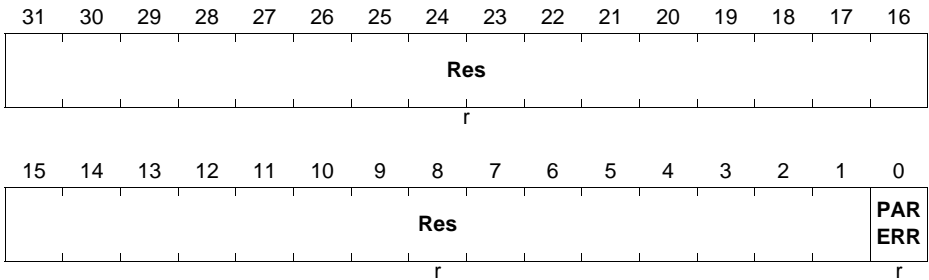
Field	Bits	Type	Description
MOD_REV	[7:0]	r	Module Revision Number MOD_REV defines the revision number. The value of a module revision starts with 01H (first revision).
MOD_TYPE	[15:8]	r	Module Type This bit field is C0H. It defines the module as a 32-bit module.
MOD_NUMBE R	[31:16]	r	Module Number Value This bit field defines the USIC module identification number (0093H = EtherCAT).

Register ECAT0 Status (STATUS)

The register shows specific status information of the ECAT module.

ECAT0_STATUS

ECAT0 Status (0E08_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
PARERR	0	r	PARITY ERROR 0 _B No Error 1 _B Parity Error in User or Process RAM
Res	[31:1]	r	Reserved Read as 0; should be written with 0.

15.16 Interconnects

EtherCAT Slave Controller (ECAT)

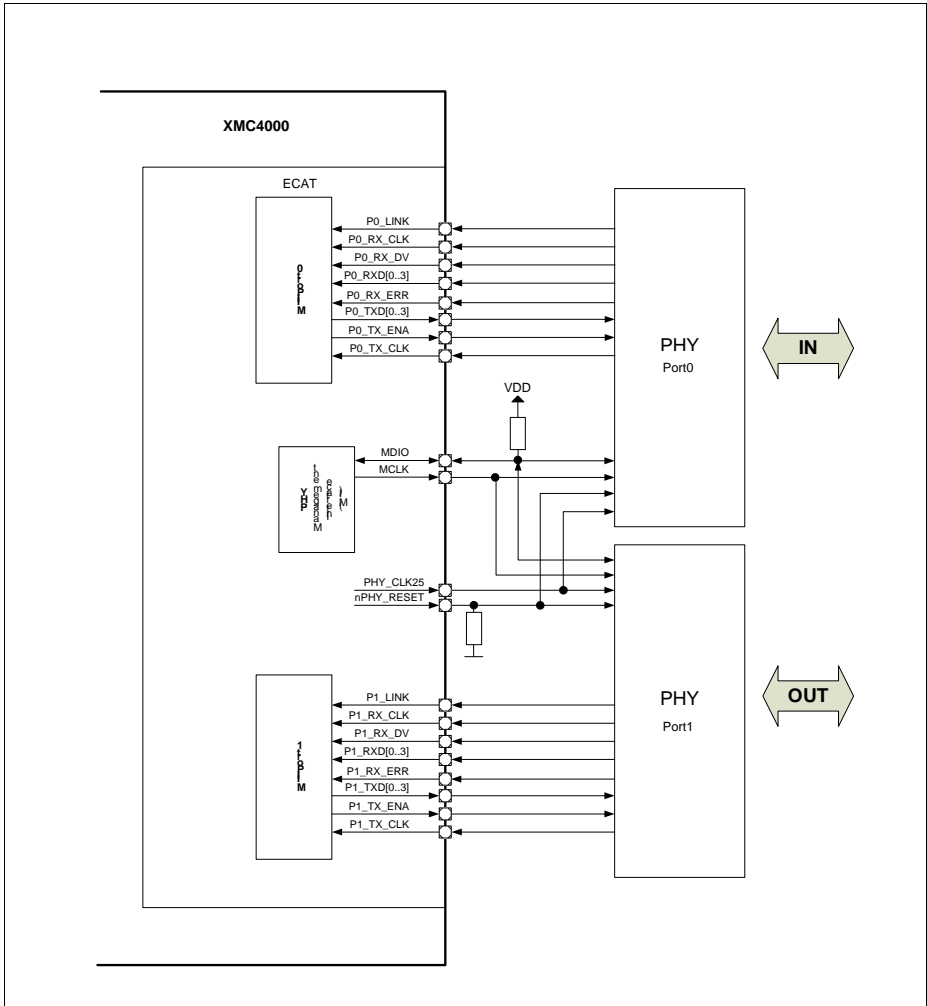


Figure 15-10 EtherCAT Slave Controller PHY Interconnects

Conventions for signal notation:

- Dot "." is the hierarchy delimiter.
- Square brackets "["]" are the array delimiters.

Table 15-13 ECAT Interconnects

Input/Output	I/O	Connected To	Description
MII Port 0			
P0_LINKA	I	PORT	
P0_LINKB	I	PORT	
P0_LINKC	I	PORT	
P0_LINKD	I	PORT	
P0_RX_CLKA	I	PORT	
P0_RX_CLKB	I	PORT	
P0_RX_CLKC	I	PORT	
P0_RX_CLKD	I	PORT	
P0_RX_DVA	I	PORT	
P0_RX_DVB	I	PORT	
P0_RX_DVC	I	PORT	
P0_RX_DVD	I	PORT	
P0_RXD0A	I	PORT	
P0_RXD0B	I	PORT	
P0_RXD0C	I	PORT	
P0_RXD0D	I	PORT	
P0_RXD1A	I	PORT	
P0_RXD1B	I	PORT	
P0_RXD1C	I	PORT	
P0_RXD1D	I	PORT	
P0_RXD2A	I	PORT	
P0_RXD2B	I	PORT	
P0_RXD2C	I	PORT	
P0_RXD2D	I	PORT	
P0_RXD3A	I	PORT	
P0_RXD3B	I	PORT	
P0_RXD3C	I	PORT	
P0_RXD3D	I	PORT	
P0_RX_ERRA	I	PORT	

EtherCAT Slave Controller (ECAT)

Table 15-13 ECAT Interconnects

Input/Output	I/O	Connected To	Description
P0_RX_ERRB	I	PORT	
P0_RX_ERRC	I	PORT	
P0_RX_ERRD	I	PORT	
P0_TXD0	O	PORT	
P0_TXD1	O	PORT	
P0_TXD2	O	PORT	
P0_TXD3	O	PORT	
P0_TX_ENA	O	PORT	
P0_TX_CLKA	I	PORT	
P0_TX_CLKB	I	PORT	
P0_TX_CLKC	I	PORT	
P0_TX_CLKD	I	PORT	

MII Port 1

P1_LINKA	I	PORT	
P1_LINKB	I	PORT	
P1_LINKC	I	PORT	
P1_LINKD	I	PORT	
P1_RX_CLKA	I	PORT	
P1_RX_CLKB	I	PORT	
P1_RX_CLKC	I	PORT	
P1_RX_CLKD	I	PORT	
P1_RX_DVA	I	PORT	
P1_RX_DVB	I	PORT	
P1_RX_DVC	I	PORT	
P1_RX_DVD	I	PORT	
P1_RXD0A	I	PORT	
P1_RXD0B	I	PORT	
P1_RXD0C	I	PORT	
P1_RXD0D	I	PORT	

EtherCAT Slave Controller (ECAT)

Table 15-13 ECAT Interconnects

Input/Output	I/O	Connected To	Description
P1_RXD1A	I	PORT	
P1_RXD1B	I	PORT	
P1_RXD1C	I	PORT	
P1_RXD1D	I	PORT	
P1_RXD2A	I	PORT	
P1_RXD2B	I	PORT	
P1_RXD2C	I	PORT	
P1_RXD2D	I	PORT	
P1_RXD3A	I	PORT	
P1_RXD3B	I	PORT	
P1_RXD3C	I	PORT	
P1_RXD3D	I	PORT	
P1_RX_ERRA	I	PORT	
P1_RX_ERRB	I	PORT	
P1_RX_ERRC	I	PORT	
P1_RX_ERRD	I	PORT	
P1_TXD0	O	PORT	
P1_TXD1	O	PORT	
P1_TXD2	O	PORT	
P1_TXD3	O	PORT	
P1_TX_ENA	O	PORT	
P1_TX_CLKA	I	PORT	
P1_TX_CLKB	I	PORT	
P1_TX_CLKC	I	PORT	
P1_TX_CLKD	I	PORT	
MII Management Interface (MI)			
MCLK	O	PORT	
MDO	O	PORT	This signal is used by HW Pin control
MDIA	I	PORT	This signal is used by HW Pin control

EtherCAT Slave Controller (ECAT)

Table 15-13 ECAT Interconnects

Input/Output	I/O	Connected To	Description
MDIB	I	PORT	This signal is used by HW Pin control
MDIC	I	PORT	This signal is used by HW Pin control
MDID	I	PORT	This signal is used by HW Pin control
LED			
LED_RUN	O	PORT	
LED_ERR	O	PORT	
LED_STATE_RUN	O	PORT	
LED_LINK_ACT_P0	O	PORT	
LED_LINK_ACT_P1	O	PORT	
Sync/Latch			
SYNC0	O	PORT ERU1.2B3 VADC.GxREQ TRG VADC.BGREQ TRG	
SYNC1	O	PORT ERU1.3A3 VADC.GxREQ TRH VADC.BGREQ TRH	
LATCH0A	I	PORT	
LATCH0B	I	PORT	
LATCH0C	I	ERU0.PDOUT0	
LATCH0D	I	ERU1.PDOUT0	
LATCH1A	I	PORT	
LATCH1B	I	PORT	
LATCH1C	I	ERU0.PDOUT1	
LATCH1D	I	ERU1.PDOUT1	
Misc			
PHY_RESET	O	PORT	

Table 15-13 ECAT Interconnects

Input/Output	I/O	Connected To	Description
PHY_CLK25	O	PORT	
...	...		

16 Universal Serial Bus (USB)

The USB module is a Dual-Role Device (DRD) controller that supports both device and host functions and complies fully with the On-The-Go Supplement to the USB 2.0 Specification, Revision 1.3. It can also be configured as a host-only or device-only controller, fully compliant with the USB 2.0 Specification.

The USB core's USB 2.0 configurations support full-speed (12-Mbps) transfers.

The USB core is optimized for the following applications and systems:

- Portable electronic devices
- Point-to-point applications (direct connection to FS device)

References:

[18] USB 2.0 specification (April 27, 2000).

[19] On-The-Go Supplement to the USB 2.0 specification (Revision 1.3, December 5, 2006).

Table 16-1 Abbreviations

DWORD	32-bit Data Word
DRD	Dual-Role Device
FS	Full Speed
MAC	Media Access Controller
OTG	On-The-Go
PHY	Physical Layer
SOF	Start of Frame

16.1 Overview

This section describes the features and provides a block diagram of the USB module.

16.1.1 Features

The USB module includes the following features:

- Complies with the USB 2.0 Specification
- Complies with the On-The-Go Supplement to the USB 2.0 Specification (Revision 1.3)
- Configurable as Device only, Host only or as an OTG Dual Role Device
- Support for the Full-Speed (12-Mbps) mode
- Provides a USB OTG FS PHY interface
- Supports up to 7 bidirectional endpoints, including control endpoint 0

- Supports up to 14 Host channels
- Supports Session Request Protocol (SRP).
- Supports Host Negotiation Protocol (HNP).
- Supports SOFs in Full-Speed modes.
- Supports clock gating for power saving
- Supports USB suspend/resume
- Supports USB soft disconnect
- Supports DMA mode in:
 - Descriptor-Based Scatter/Gather DMA operation
 - Buffer DMA operation
- 2 Kbytes of RAM for data FIFO consisting of 512 DWORDs
- Dedicated transmit FIFO for each of the device IN endpoints in Slave and DMA modes. Each FIFO can hold multiple packets.
- Supports packet-based, Dynamic FIFO memory allocation for endpoints for small FIFOs and flexible, efficient use of RAM.
- Provides support to change an endpoint's FIFO memory size during transfers.
- Supports endpoint FIFO sizes that are not powers of 2, to allow the use of contiguous memory locations.

16.1.2 Block Diagram

Figure 16-1 shows the USB module block diagram.

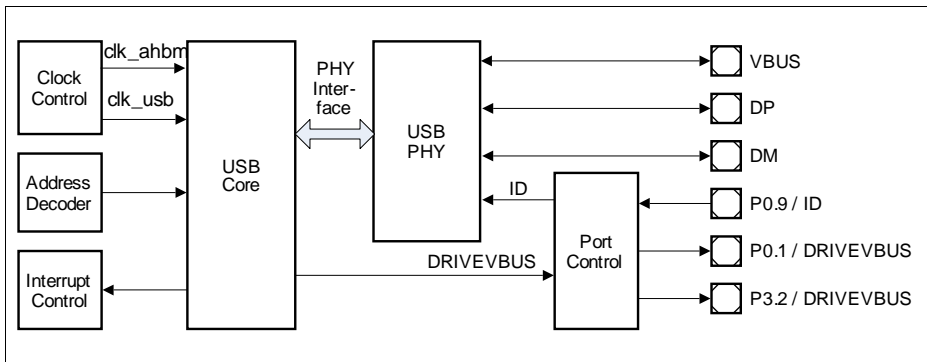


Figure 16-1 USB Module Block Diagram

16.2 Functional Description

This chapter describes the operation modes and the FIFO architecture of the USB module.

16.2.1 OTG Dual-Role Device (DRD)

The OTG DRD provides both Host and Device functions and supports the Host Negotiation Protocol (HNP) and Session Request Protocol (SRP). It is able to detect whether an A- or B-device is connected by sampling the ID input signal.

To drive the VBUS as an A-device, the OTG DRD requires an external charge pump, which is enabled through the output signal DRIVEBUS.

Figure 16-2 shows the connections of the DRD.

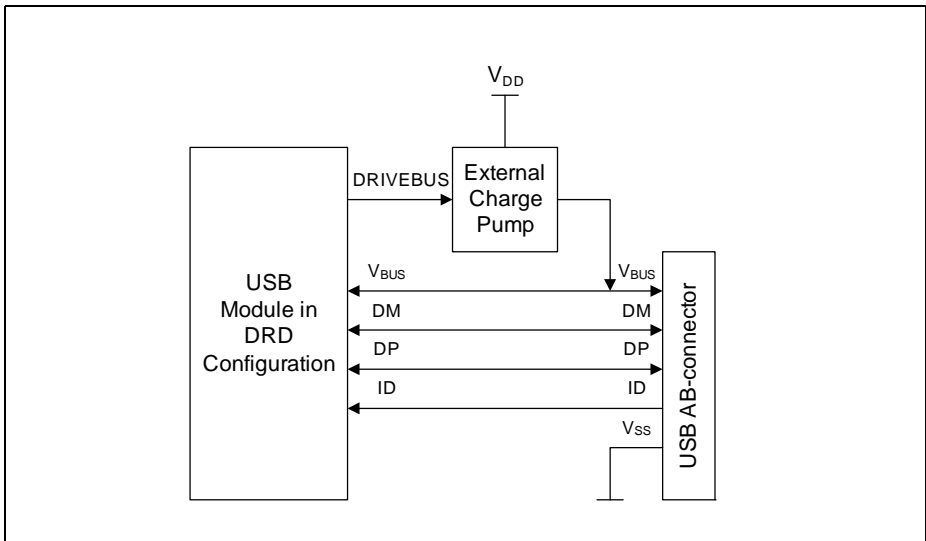


Figure 16-2 OTG DRD Connections

16.2.2 USB Host

The USB Host supports up to 14 Host channels, each configurable for the transfer type (Control, Bulk, Interrupt, or Isochronous) and direction (IN or OUT). To drive the VBUS, it requires an external charge pump, which is enabled through the output signal DRIVEBUS.

Figure 16-3 shows the connections of the USB Host.

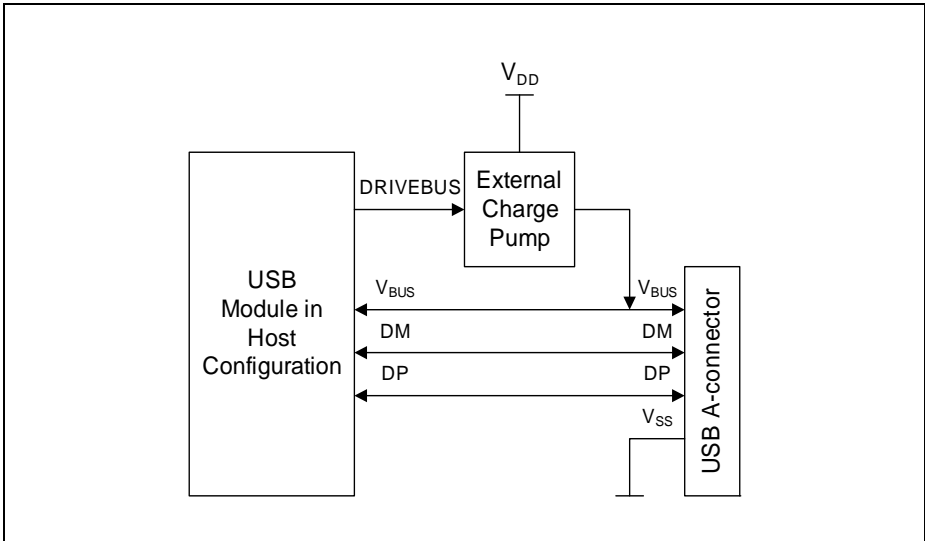


Figure 16-3 USB Host Connections

16.2.3 USB Device

The USB Device supports Control transfers through the bidirectional endpoint 0, and Bulk, Interrupt or Isochronous transfers configurable from within the other 6 bidirectional endpoints. Being a self-powered device, it does not require an additional external voltage regulator.

Software can disconnect the USB Device from the USB through the DCTL.SftDiscon bit.

Figure 16-4 shows the connections of the USB Device.

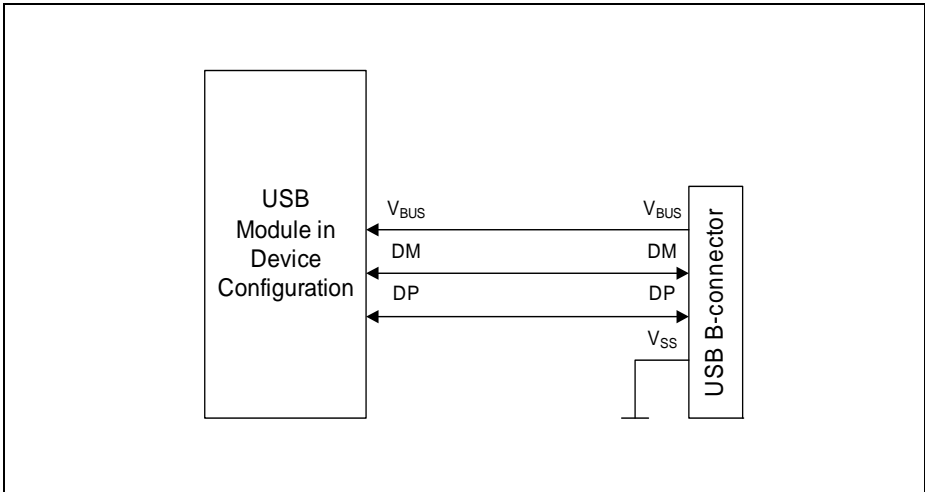


Figure 16-4 USB Device Connections

16.2.4 FIFO Architecture

This section describes the FIFO architecture in a USB Host and Device.

16.2.4.1 Host FIFO Architecture

The host uses one transmit FIFO for all non-periodic OUT transactions and one transmit FIFO for all periodic OUT transactions (periodic FIFOs 2 to n are only used in Device mode, where n is number of periodic IN endpoints in Device mode). These transmit FIFOs are used as transmit buffers to hold the data (payload of the transmit packet) to be transmitted over USB. The host pipes the USB transactions through Request queues (one for periodic and one for non-periodic). Each entry in the Request queue holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the requests are written into the queue determines the sequence of transactions on the USB. The host processes the periodic Request queue first, followed by the non-periodic Request queue, at the beginning of each frame.

The host uses one receive FIFO for all periodic and non-periodic transactions. The FIFO is used as a receive buffer to hold the received data (payload of the received packet) from the USB until it is transferred to the system memory. The status of each packet received also goes into the FIFO. The status entry holds the IN channel number along with other information, such as received byte count and validity status, to perform a transaction on the AHB.

16.2.4.2 Device FIFO Architecture

This section describes the USB device FIFO architecture.

Dedicated Transmit FIFO Operation

The core uses individual transmit FIFOs for each IN endpoint.

The core internally handles underrun condition during transmit and corrupts the packet (inverts the CRC) on the USB. If packet transmission results in an underrun condition (eventually resulting in packet corruption on the USB), the host can time out the endpoint after three consecutive errors.

Single Receive FIFO

The USB device uses a single receive FIFO to receive the data for all the OUT endpoints. The receive FIFO holds the status of the received data packet, such as byte count, data PID and the validity of the received data. The DMA or the application reads the data out of the receive FIFO as it is received.

16.3 Programming Overview

This section provides an overview of the programming options available in the USB core in different modes of operation.

16.3.1 Programming Options on DMA

The application can operate the core in either of two modes:

- In **DMA Mode** — the core fetches the data to be transmitted or updates the received data on the AHB.
- In **Slave Mode** — the application initiates the data transfers for data fetch and store.

The application cannot operate the core using a combination of DMA and Slave simultaneously. The application can operate the DMA in:

- Scatter/Gather mode (a Descriptor-Based mode).
- Buffer-pointer based mode.

16.3.1.1 DMA Mode

In this mode, the OTG host uses the AHB master Interface for transmit packet data fetch (AHB to USB) and receive data update (USB to AHB). The AHB master uses the programmed DMA address (HCDMAx register in host mode and DIEPDMAx/DOEPDMAx register in device mode) to access the data buffers.

Transfer-Level Operation

In DMA mode, the application is interrupted only after the programmed transfer size is transmitted or received (provided the USB core detects no NAK/NYET/Timeout/Error response in Host mode, or Timeout/CRC Error in Device mode). The application must handle all transaction errors. In Device mode with dedicated FIFOs, all the USB errors are handled by the core itself.

Transaction-Level Operation

This mode is similar to transfer-level operation with the programmed transfer size equal to one packet size (either maximum packet size, or a short packet size). When Scatter/Gather DMA is enabled, the transfer size is extracted from the descriptors.

16.3.1.2 Slave Mode

In Slave mode, the application can operate the USB core either in transaction-level (packet-level) operation or in pipelined transaction-level operation.

Transaction-Level Operation

The application handles one data packet at a time per channel/endpoint in transaction-level operations. Based on the handshake response received on the USB, the application determines whether to retry the transaction or proceed with the next, until the end of the transfer. The application is interrupted on completion of every packet. The application performs transaction-level operations for a channel/endpoint for a transmission (host: OUT/ device: IN) or reception (host: IN / device: OUT) as shown in [Figure 16-5](#) and [Figure 16-6](#).

Transaction-Level Operation: Host Mode

For an OUT transaction, the application enables the channel and writes the data packet into the corresponding (Periodic or Non-periodic) transmit FIFO. The USB core automatically writes the channel number into the corresponding (Periodic or Non-periodic) Request Queue, along with the last DWORD write of the packet.

For an IN transaction, the application enables the channel and the USB core automatically writes the channel number into the corresponding Request queue. The application must wait for the packet received interrupt, then empty the packet from the receive FIFO.

Transaction-Level Operation: Device Mode

For an IN transaction, the application enables the endpoint, writes the data packet into the corresponding transmit FIFO, and waits for the packet completion interrupt from the core.

For an OUT transaction, the application enables the endpoint, waits for the packet received interrupt from the core, then empties the packet from the receive FIFO.

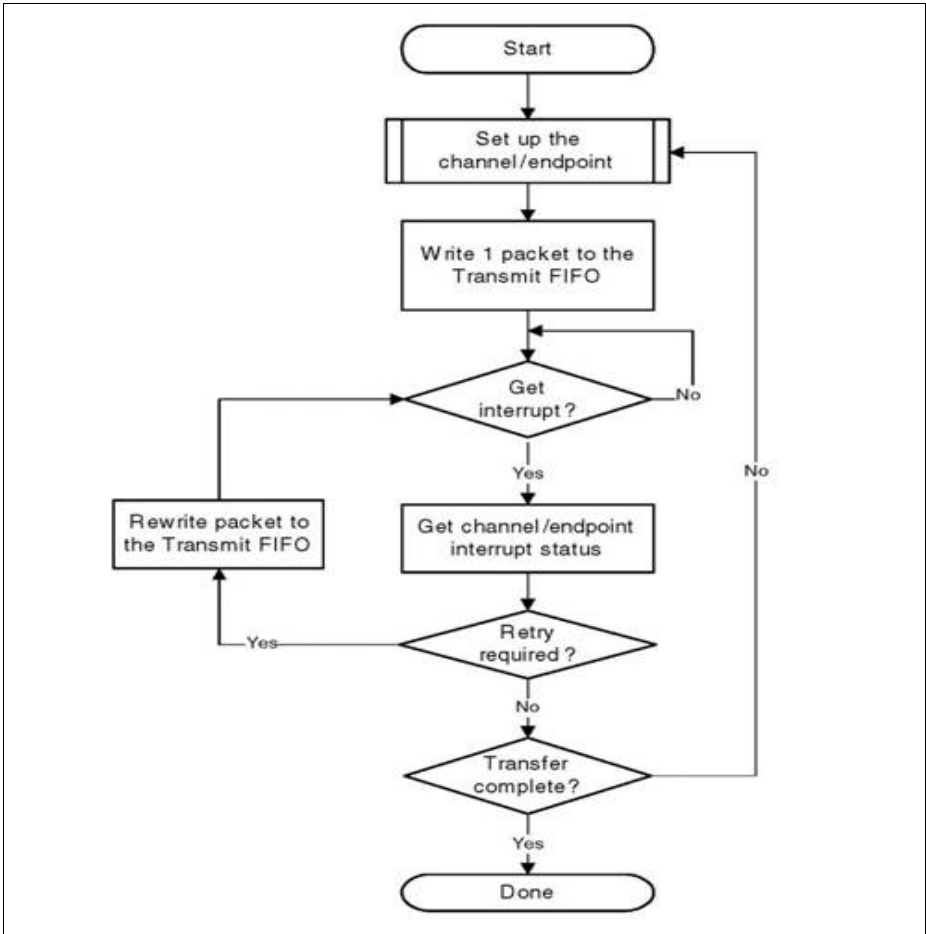


Figure 16-5 Transmit Transaction-Level Operation in Slave Mode

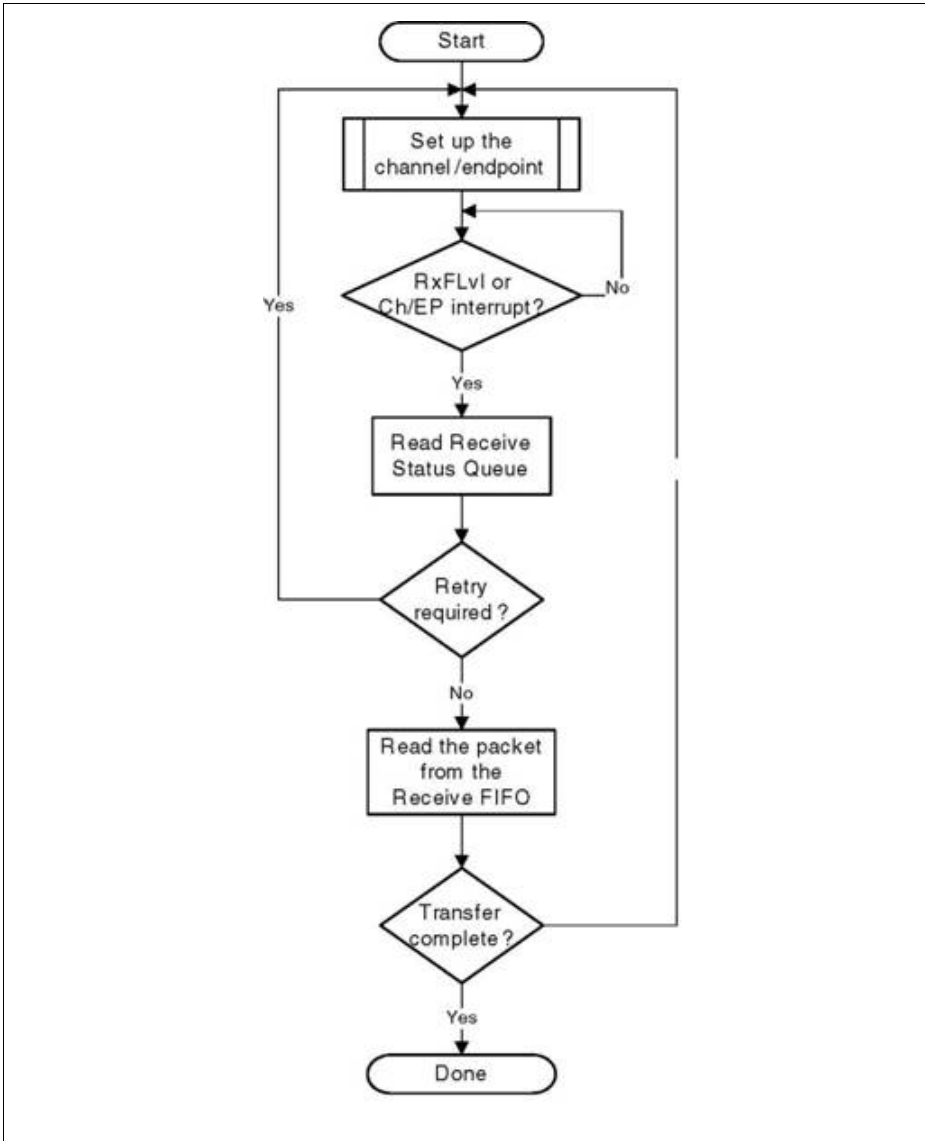


Figure 16-6 Receive Transaction-Level Operation in Slave Mode

Note: The application has to finish writing one complete packet before switching to a different channel/endpoint FIFO. Violating this rule results in an error.

Pipelined Transaction-Level Operation

The application can pipeline more than one transaction (IN or OUT) with pipelined transaction-level operation, which is analogous to Transfer mode in DMA mode. In pipelined transaction-level operation, the application can program the core to perform multiple transactions. The advantage of this mode compared to transaction-level operation is that the application is not interrupted on a packet basis.

Pipelined Transaction-Level Operation: Host Mode

For an OUT transaction, the application sets up a transfer and enables the channel. The application can write multiple packets back-to-back for the same channel into the transmit FIFO, based on the space availability. It can also pipeline OUT transactions for multiple channels by writing into the HCHARn register, followed by a packet write to that channel. The core writes the channel number, along with the last DWORD write for the packet, into the Request queue and schedules transactions on the USB in the same order.

For an IN transaction, the application sets up a transfer and enables the channel, and the USB core writes the channel number into the Request queue. The application can schedule IN transactions on multiple channels, provided space is available in the Request queue. The core initiates an IN token on the USB only when there is enough space to receive at least of one maximum-packet-size packet of the channel in the top of the Request queue.

Pipelined Transaction-Level Operation: Device Mode

For an IN transaction, the application sets up a transfer and enables the endpoint. The application can write multiple packets back-to-back for the same endpoint into the transmit FIFO, based on available space. It can also pipeline IN transactions for multiple channels by writing into the DIEPCTLx register followed by a packet write to that endpoint. The core writes the endpoint number, along with the last DWORD write for the packet into the Request queue. The core transmits the data in the transmit FIFO when an IN token is received on the USB.

For an OUT transaction, the application sets up a transfer and enables the endpoint. The core receives the OUT data into the receive FIFO, when it has available space. As the packets are received into the FIFO, the application must empty data from it.

From this point on in this chapter, the terms "Pipelined Transaction mode" and "Transfer mode" are used interchangeably.

16.3.2 Core Initialization

If the cable is connected during power-up, the Current Mode of Operation bit in the Core Interrupt register (GINTSTS.CurMod) reflects the mode. The USB core enters Host mode when an "A" plug is connected, or Device mode when a "B" plug is connected.

Universal Serial Bus (USB)

This section explains the initialization of the USB core after power-on. The application must follow the initialization sequence irrespective of Host or Device mode operation.

1. Program the following fields in the Global AHB Configuration (GAHBCFG) register.
 - a) DMA Mode bit
 - b) AHB Burst Length field
 - c) Global Interrupt Mask bit = 1
 - d) RxFIFO Non-Empty (GINTSTS.RxFLvl) (applicable only when the core is operating in Slave mode)
 - e) Non-periodic TxFIFO Empty Level (can be enabled only when the core is operating in Slave mode as a host.)
 - f) Periodic TxFIFO Empty Level (can be enabled only when the core is operating in Slave mode)
2. Program the following fields in GUSBCFG register.
 - a) HNP Capable bit
 - b) SRP Capable bit
 - c) FS Time-Out Calibration field
 - d) USB Turnaround Time field
3. The software must unmask the following bits in the GINTMSK register.
 - a) OTG Interrupt Mask
 - b) Mode Mismatch Interrupt Mask
4. The software can read the GINTSTS.CurMod bit to determine whether the USB core is operating in Host or Device mode. The software then follows either the **“Host Initialization” on Page 16-12** or **“Device Initialization” on Page 16-72** sequence.

Note: The core is designed to be interrupt-driven. Polling interrupt mechanism is not recommended: this may result in undefined resolutions.

Note: In device mode, just after Power On Reset or a Soft Reset, the GINTSTS.Sof bit is set to 1_B for debug purposes. This status must be cleared and can be ignored.

16.4 Host Programming Overview

This section discusses how to program the USB core when it is in Host mode.

16.4.1 Host Initialization

To initialize the core as host, the application must perform the following steps.

1. Program GINTMSK.PrtIntMsk to unmask.
2. Program the HCFG register to select full-speed host.
3. Program the HPRT.PrtPwr bit to 1_B. This drives VBUS on the USB.
4. Wait for the HPRT.PrtConnDet interrupt. This indicates that a device is connected to the port.
5. Program the HPRT.PrtRst bit to 1_B. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.

7. Program the HPRT.PrtRst bit to 0_B.
8. Wait for the HPRT.PrtEnChng interrupt.
9. Read the HPRT.PrtSpd field to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock.¹⁾
11. Program the GRXFSIZ register to select the size of the receive FIFO.
12. Program the GNPTXFSIZ register to select the size and the start address of the Non-periodic Transmit FIFO for non-periodic transactions.
13. Program the HPTXFSIZ register to select the size and start address of the Periodic Transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel as described in **“Channel Initialization in Buffer DMA or Slave Mode” on Page 16-14**. If the core is operating in Scatter/Gather DMA mode, see **“Channel Initialization in Scatter-Gather DMA Mode” on Page 16-64**.

16.4.2 Host Connection

The host connect flow is as follows:

1. When the USB cable is plugged to the host port, the core triggers GINTSTS.ConIDStsChng interrupt.
2. When the host application detects GINTSTS.ConIDStsChng interrupt, it can perform one of the following actions:
 - a) Turn on VBUS by setting HPRT.PrtPwr = 1_B, or
 - b) Wait for SRP Signaling from Device to turn on VBUS.
3. The PHY indicates VBUS power-on by asserting utmi_vbusvalid.
4. When the host core detects the device connection, it triggers the Host Port Interrupt (GINTSTS.PrtInt) to the application.
5. When GINTSTS.PrtInt is triggered, the application reads the HPRT register to check if the HPRT.Port Connect Detected (PrtConnDet) bit is set or not.

16.4.3 Host Disconnection

The host disconnect flow is as follows:

1. When the device is disconnected from the USB cable (but the cable is still connected to the USB host), the core triggers GINTSTS.DisconnInt (Disconnect Detected) interrupt.
 - a) If the USB cable is disconnected from the host port without removing the device, the core generates an additional interrupt - GINTSTS.ConIDStsChng (Connector ID Status Change).

1) At this point, the host is up and running and the port register begins to report device disconnects, etc. The port is active with SOFs occurring down the enabled port.

Universal Serial Bus (USB)

- The host application can choose to turn off the VBUS by programming $HPRT.PrtPwr = 0_B$.

16.4.4 Channel Initialization in Buffer DMA or Slave Mode

To communicate with devices, the application must initialize and enable at least one channel.

To initialize and enable a channel when the host core is in Buffer DMA or Slave mode, the application must perform the following steps.

- Program the GINTMSK register to unmask the following:
 - Non-periodic Transmit FIFO Empty for OUT transactions (applicable for Slave mode that operates in pipelined transaction-level with the Packet Count field programmed with more than one).
 - Non-periodic Transmit FIFO Half-Empty for OUT transactions (applicable for Slave mode that operates in pipelined transaction-level with the Packet Count field programmed with more than one).
- Program the HAINTMSK register to unmask the selected channels' interrupts.
- Program the HCINTMSK register to unmask the transaction-related interrupts of interest given in the Host Channel Interrupt register.
- Program the selected channel's HCTSIZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).
- Program the Transfer Size field so that the channel's transfer size is a multiple of the maximum packet size.


```
if (mps[epnum] mod 4) == 0
transfer size[epnum] = n * (mps[epnum]) //Dword Aligned
else
transfer size[epnum] = n * (mps[epnum] + 4 - (mps[epnum] mod 4))
//Non-Dword Aligned
packet count[epnum] = n
n > 0
```
- Program the HCCHARx register of the selected channel with the device's endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the Channel Enable bit to 1_B only when the application is ready to transmit or receive any packet).

Repeat steps 1-6 for other channels.

Note: De-allocate channel means after the transfer has completed, the channel is disabled. When the application is ready to start the next transfer, the application re-initializes the channel by following these steps.

16.4.5 Halting a Channel

The application can disable any channel by programming the HCCHARx register with the HCCHARx.ChDis and HCCHARx.ChEna bits set to 1_B. This enables the USB host to flush the posted requests (if any) and generates a Channel Halted interrupt. The application must wait for the HCINTx.ChHltd interrupt before reallocating the channel for other transactions. The USB host does not interrupt the transaction that has been already started on USB.

In Slave mode operation, before disabling a channel, the application must ensure that there is at least one free space available in the Non-periodic Request Queue (when disabling a non-periodic channel) or the Periodic Request Queue (when disabling a periodic channel). The application can simply flush the posted requests when the Request queue is full (before disabling the channel), by programming the HCCHARx register with the HCCHARx.ChDis bit set to 1_B, and the HCCHARx.ChEna bit reset to 0_B.

The core generates a RxFLvl interrupt when there is an entry in the queue. The application must read/pop the GRXSTSP register to generate the Channel Halted interrupt.

To disable a channel in DMA mode operation, the application need not check for space in the Request queue. The USB host checks for space in which to write the Disable request on the disabled channel's turn during arbitration. Meanwhile, all posted requests are dropped from the Request queue when the HCCHARx.ChDis bit is set to 1_B.

The application is expected to disable a channel under any of the following conditions:

1. When a HCINTx.XferCompl interrupt is received during a non-periodic IN transfer or high- bandwidth interrupt IN transfer (Slave mode only)
2. When a HCINTx.STALL, HCINTx.XactErr, HCINTx.BblErr, or HCINTx.DataTglErr interrupt is received for an IN or OUT channel (Slave mode only). For high-bandwidth interrupt INs in Slave mode, once the application has received a DataTglErr interrupt it must disable the channel and wait for a Channel Halted interrupt. The application must be able to receive other interrupts (DataTglErr, Nak, Data, XactErr, BabbleErr) for the same channel before receiving the halt.
3. When a GINTSTS.DisconnInt (Disconnect Device) interrupt is received. The application must check for the HPRT.PrtConnSts, because when the device directly connected to the host is disconnected, HPRT.PrtConnSts is reset. The software must issue a soft reset to ensure that all channels are cleared. When the device is reconnected, the host must issue a USB Reset.
4. When the application aborts a transfer before normal completion (Slave and DMA modes).

Note

In buffer DMA mode, the following guidelines must be considered:

Universal Serial Bus (USB)

- Channel disable must not be programmed for non-split periodic channels. At the end of the next frame (in the worst case), the core generates a channel halted and disables the channel automatically.
- For split enabled channels (both non-periodic and periodic), channel disable must not be programmed randomly. However, channel disable can be programmed for specific scenarios such as NAK and FrmOvrn as defined in the Host programming model.

16.4.6 Selecting the Queue Depth

Choose the Periodic and Non-periodic Request Queue depths carefully to match the number of periodic/non-periodic endpoints accessed.

The Non-periodic Request Queue depth affects the performance of non-periodic transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline non-periodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

The core's Periodic Request Queue depth is critical to performing periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a frame. In Slave mode, however, the application must also take into account the disable entry that must be put into the queue. So, if there are two non-high-bandwidth periodic endpoints, the Periodic Request Queue depth must be at least 4. If at least one high-bandwidth endpoint supported, the queue depth must be 8. If the Periodic Request Queue depth is smaller than the periodic transfers scheduled in a frame, a frame overrun condition results.

16.4.7 Handling Special Conditions

This section discusses how to handle certain special conditions.

16.4.7.1 Handling Babble Conditions

USB core handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

When USB core detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already-written data in the Rx buffer and generates a Babble interrupt to the application.

When USB core detects a port babble, it flushes the Rx FIFO and disables the port. The core then generates a Port Disabled Interrupt (GINTSTS.PrtInt, HPRT.PrtEnChng). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the Port Disabled interrupt) by checking

HPRT.PrtOvrCurrAct, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

16.4.7.2 Handling Disconnects

If the device is disconnected suddenly, a GINTSTS.DisconnInt interrupt is generated. When the application receives this interrupt, it must issue a soft reset by programming the GRSTCTL.CSftRst bit.

16.4.8 Host HFIR Functionality

The Host Frame Interval Register (HFIR) specifies the interval between two consecutive SOFs. This field contains the number of PHY clocks that constitute the required frame interval and is primarily used to regulate the SOF duration based on the phy_clk frequency.

16.4.8.1 HFIR Behaviour when HFIR.HFIRIdCtrl = 0_B

This section describes the core behavior when HFIR.HFIRIdCtrl = 0_B using the waveform shown in [Figure 16-7](#).

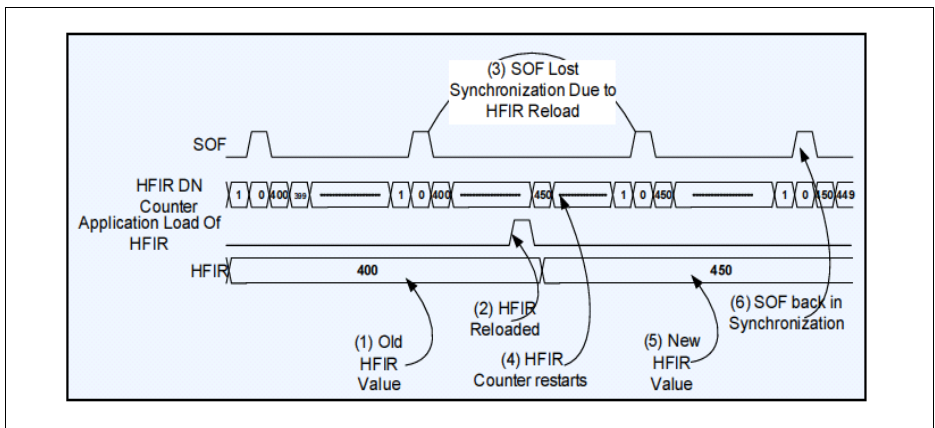


Figure 16-7 Functionality when HFIRIdCtrl = 0_B

The following numbered steps correspond to those in [Figure 16-7](#):

1. Depicts the present HFIR value programmed by the application after power on reset.
2. The application reloads the HFIR register with a new value.
3. Because the HFIR down counter is reloaded, it starts counting again immediately because of which the SOF synchronization is lost.
4. The HFIR counter is restarted.

5. The HFIR register receives the new programmed value.
6. After the first SOF is generated with the new HFIR functionality, SOF synchronization is regained.

16.4.8.2 HFIR Behaviour when HFIR.HFIRIdCtrl = 1_B

This section describes the core behavior when HFIR.HFIRIdCtrl = 1_B using the waveform shown in [Figure 16-8](#).

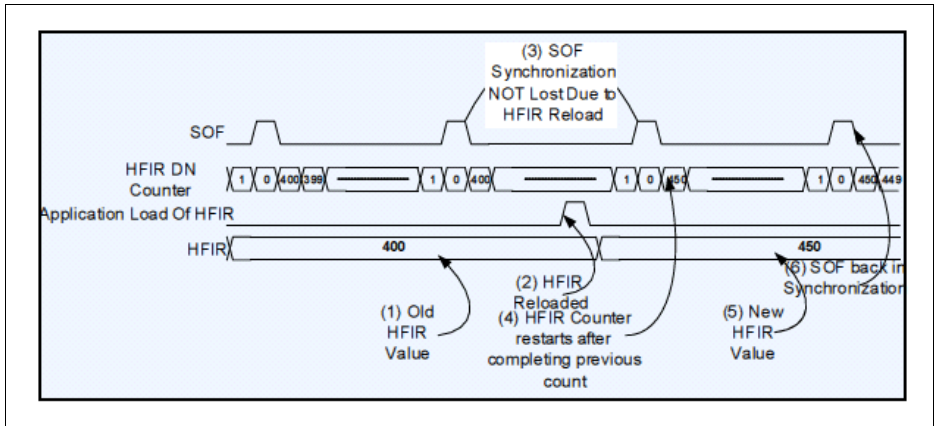


Figure 16-8 Functionality when HFIRIdCtrl = 1_B

The following numbered steps correspond to those in [Figure 16-8](#):

1. Depicts the present HFIR value programmed by the application after power on reset.
2. The application loads the new HFIR value; the HFIR counter does not take this new value and continues counting till the counter reaches zero.
3. The SOF is generated when the counter reaches zero with the old HFIR programmed value.
4. The HFIR counter takes the new value.
5. The new HFIR value takes effect.
6. The SOF is back in synchronization.

16.4.9 Host Programming for Various USB Transactions

[Table 16-2](#) provides links to the programming sequence for the different types of USB transactions.

Table 16-2 Host Programming Operations

Mode	IN	OUT/SETUP
Control		
Slave	“Bulk and Control IN Transactions in Slave Mode” on Page 16-23	“Bulk and Control OUT/SETUP Transactions in Slave Mode” on Page 16-25
Buffer DMA	“Bulk and Control IN Transactions in Buffer DMA Mode” on Page 16-38	“Bulk and Control OUT/SETUP Transactions in Buffer DMA Mode” on Page 16-40
Scatter Gather DMA Mode	“Asynchronous Transfers” on Page 16-65”	“Asynchronous Transfers” on Page 16-65
Bulk		
Slave	“Bulk and Control IN Transactions in Slave Mode” on Page 16-23	“Bulk and Control OUT/SETUP Transactions in Slave Mode” on Page 16-25
Buffer DMA	“Bulk and Control IN Transactions in Buffer DMA Mode” on Page 16-38	“Bulk and Control OUT/SETUP Transactions in Buffer DMA Mode” on Page 16-40
Scatter Gather DMA Mode	“Asynchronous Transfers” on Page 16-65”	“Asynchronous Transfers” on Page 16-65
Interrupt		
Slave	“Interrupt IN Transactions in Slave Mode” on Page 16-28	“Interrupt OUT Transactions in Slave Mode” on Page 16-31
Buffer DMA	“Interrupt IN Transactions in Buffer DMA Mode” on Page 16-45	“Interrupt OUT Transactions in Buffer DMA Mode” on Page 16-47
Scatter Gather DMA Mode	“Periodic Transfers” on Page 16-66	“Periodic Transfers” on Page 16-66
Isochronous		
Slave	“Isochronous IN Transactions in Slave Mode” on Page 16-34	“Isochronous OUT Transactions in Slave Mode” on Page 16-36

Table 16-2 Host Programming Operations (cont'd)

Mode	IN	OUT/SETUP
Buffer DMA	“Isochronous IN Transactions in Buffer DMA Mode” on Page 16-50	“Isochronous OUT Transactions in Buffer DMA Mode” on Page 16-52
Scatter Gather DMA Mode	“Periodic Transfers” on Page 16-66	“Periodic Transfers” on Page 16-66

16.5 Host Programming in Slave mode

This section discusses how to program the Host core when it is configured in Slave mode.

16.5.1 Writing the Transmit FIFO in Slave Mode

Figure 16-9 shows the flow diagram for writing to the transmit FIFO in Slave mode. The USB host automatically writes an entry (OUT request) to the Periodic/Non-periodic Request Queue, along with the last DWORD write of a packet. The application must ensure that at least one free space is available in the Periodic/Non-periodic Request Queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in DWORDs. If the packet size is non-DWORD aligned, the application must use padding. The USB host determines the actual packet size based on the programmed maximum packet size and transfer size.

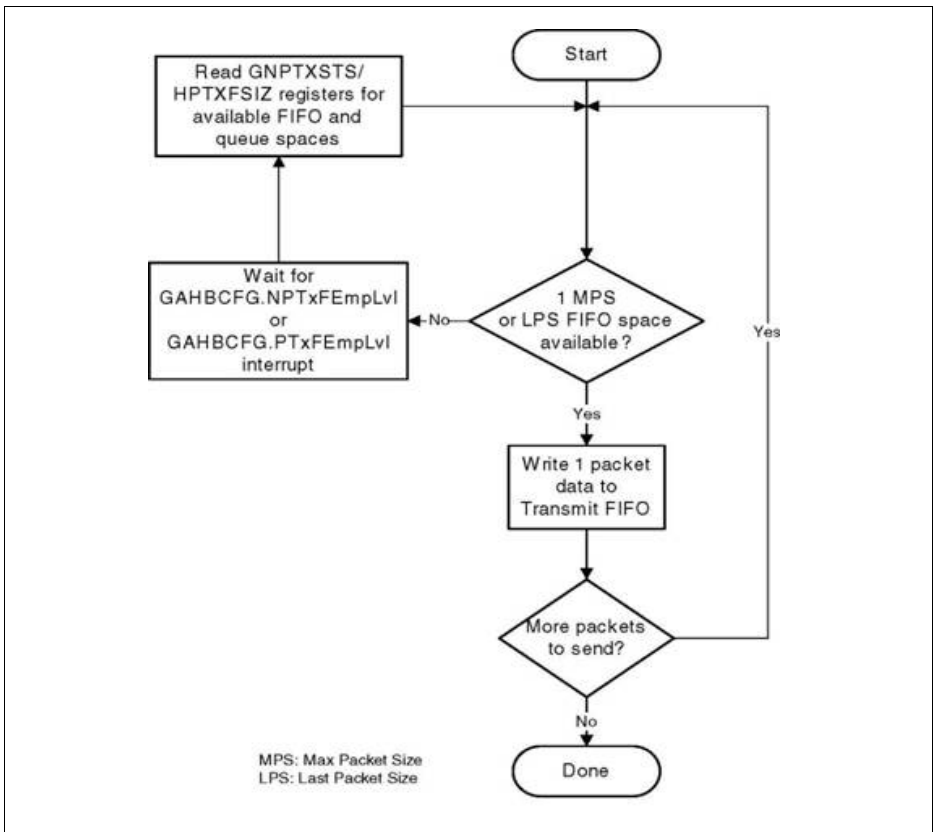


Figure 16-9 Transmit FIFO Write Task in Slave Mode

16.5.2 Reading the Receive FIFO in Slave Mode

Figure 16-10 shows the flow diagram for reading the receive FIFO in Slave mode. The application must ignore all packet statuses other than IN Data Packet (0010_B).

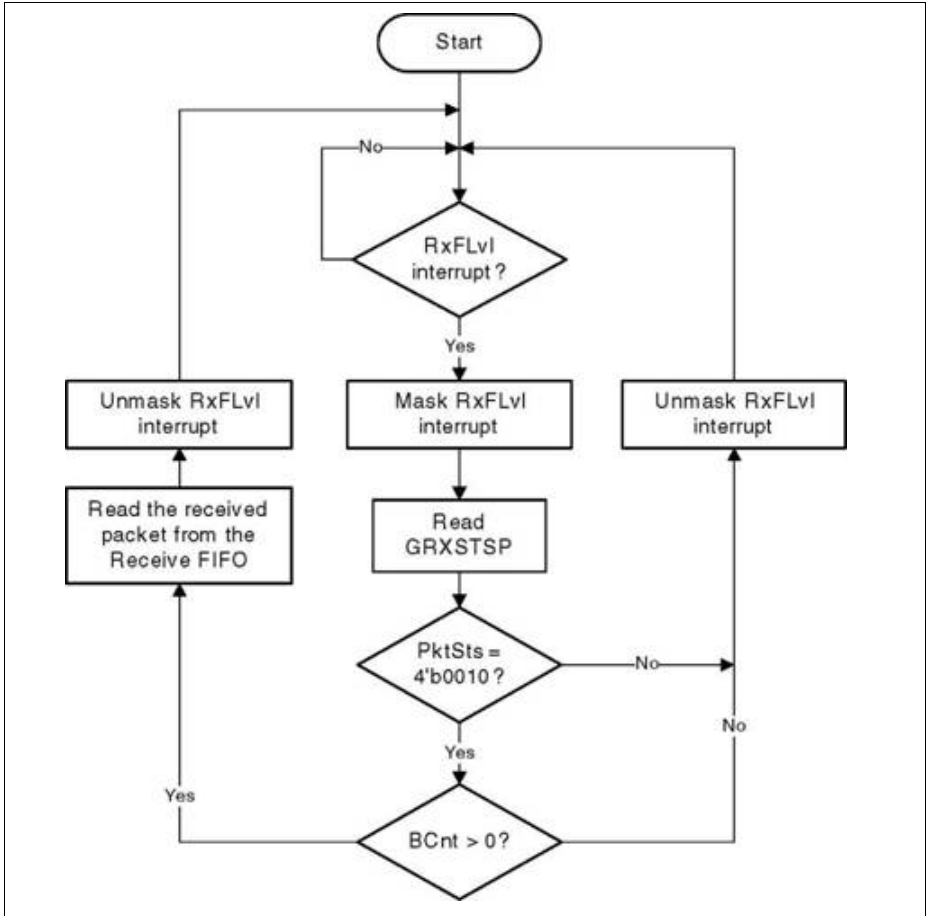


Figure 16-10 Receive FIFO Read Task in Slave Mode

16.5.3 Control Transactions in Slave Mode

Setup, Data, and Status stages of a control transfer must be performed as three separate transfers. Setup-, Data- or Status-stage OUT transactions are performed similarly to the bulk OUT transactions explained in **“Bulk and Control OUT/SETUP Transactions in Slave Mode” on Page 16-25**. Data- or Status-stage IN transactions are performed similarly to the bulk IN transactions explained in **“Bulk and Control IN Transactions in Slave Mode” on Page 16-23**. For all three stages, the application is expected to set the HCCHAR1.EPType field to Control. During the Setup stage, the application is expected to set the HCTSIZ1.PID field to SETUP.

16.5.4 Bulk and Control IN Transactions in Slave Mode

A typical bulk or control IN pipelined transaction-level operation in Slave mode is shown in **Figure 16-11**. See channel 2 (ch_2). The assumptions are:

- The application is attempting to receive two maximum-sized packets (transfer size = 1,024 bytes).
- The receive FIFO can contain at least one maximum-packet-size packet and two status DWORDs per packet (72 bytes for FS).
- The Non-periodic Request Queue depth = 4.

16.5.4.1 Normal Bulk and Control IN Operations

The sequence of operations in **Figure 16-11** (channel 2) is as follows:

1. Initialize channel 2 as explained in **“Channel Initialization in Buffer DMA or Slave Mode” on Page 16-14**.
2. Set the HCCHAR2.ChEna bit to write an IN request to the Non-periodic Request Queue.
3. The core attempts to send an IN token after completing the current OUT transaction.
4. The core generates an RxFLvl interrupt as soon as the received packet is written to the receive FIFO.
5. In response to the RxFLvl interrupt, mask the RxFLvl interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RxFLvl interrupt.
6. The core generates the RxFLvl interrupt for the transfer completion status entry in the receive FIFO.
7. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet ($\text{GRXSTSR.PktSts!} = 0010_{\text{B}}$).
8. The core generates the XferCompl interrupt as soon as the receive packet status is read.
9. In response to the XferCompl interrupt, disable the channel (see **“Halting a Channel” on Page 16-15**) and stop writing the HCCHAR2 register for further

Universal Serial Bus (USB)

requests. The core writes a channel disable request to the non-periodic request queue as soon as the HCCHAR2 register is written.

10. The core generates the RxFLvl interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The core generates a ChHltd interrupt as soon as the halt status is popped from the receive FIFO.
13. In response to the ChHltd interrupt, de-allocate the channel for other transfers.

Note: For Bulk/Control IN transfers, the application must write the requests when the Request queue space is available, and until the XferCompl interrupt is received.

16.5.4.2 Handling Interrupts

The channel-specific interrupt service routine for bulk and control IN transactions in Slave mode is shown in the following code samples.

Interrupt Service Routine for Bulk/Control IN Transactions in Slave Mode

```

Unmask (XactErr/XferCompl/BblErr/STALL/DataTglErr)
if (XferCompl)
{
    Reset Error Count
    Unmask ChHltd
    Disable Channel
    Reset Error Count
    Mask ACK
}
else if (XactErr or BblErr or STALL)
{
    Unmask ChHltd
    Disable Channel
    if (XactErr)
    {
        Increment Error Count
        Unmask ACK
    }
}
else if (ChHltd)
{
    Mask ChHltd
    if (Transfer Done or (Error_count == 3)
    {
        De-allocate Channel
    }
}

```

```

        }
    else
    {
        Re-initialize Channel
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}
else if (DataTglErr)
{
    Reset Error Count
}

```

16.5.5 Bulk and Control OUT/SETUP Transactions in Slave Mode

A typical bulk or control OUT/SETUP pipelined transaction-level operation in Slave mode is shown in [Figure 16-11](#). See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control SETUP transaction operates the same way but has only one packet. The assumptions are:

- The application is attempting to send two maximum-packet-size packets (transfer size = 1,024 bytes).
- The Non-periodic Transmit FIFO can hold two packets (128 bytes for FS).
- The Non-periodic Request Queue depth = 4.

16.5.5.1 Normal Bulk and Control OUT/SETUP Operations

The sequence of operations in [Figure 16-11](#) (channel 1) is as follows:

1. Initialize channel 1 as explained in [“Channel Initialization in Buffer DMA or Slave Mode” on Page 16-14](#).
2. Write the first packet for channel 1.
3. Along with the last DWORD write, the core writes an entry to the Non-periodic Request Queue.
4. As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame.
5. Write the second (last) packet for channel 1.
6. The core generates the XferCompl interrupt as soon as the last transaction is completed successfully.
7. In response to the XferCompl interrupt, de-allocate the channel for other transfers

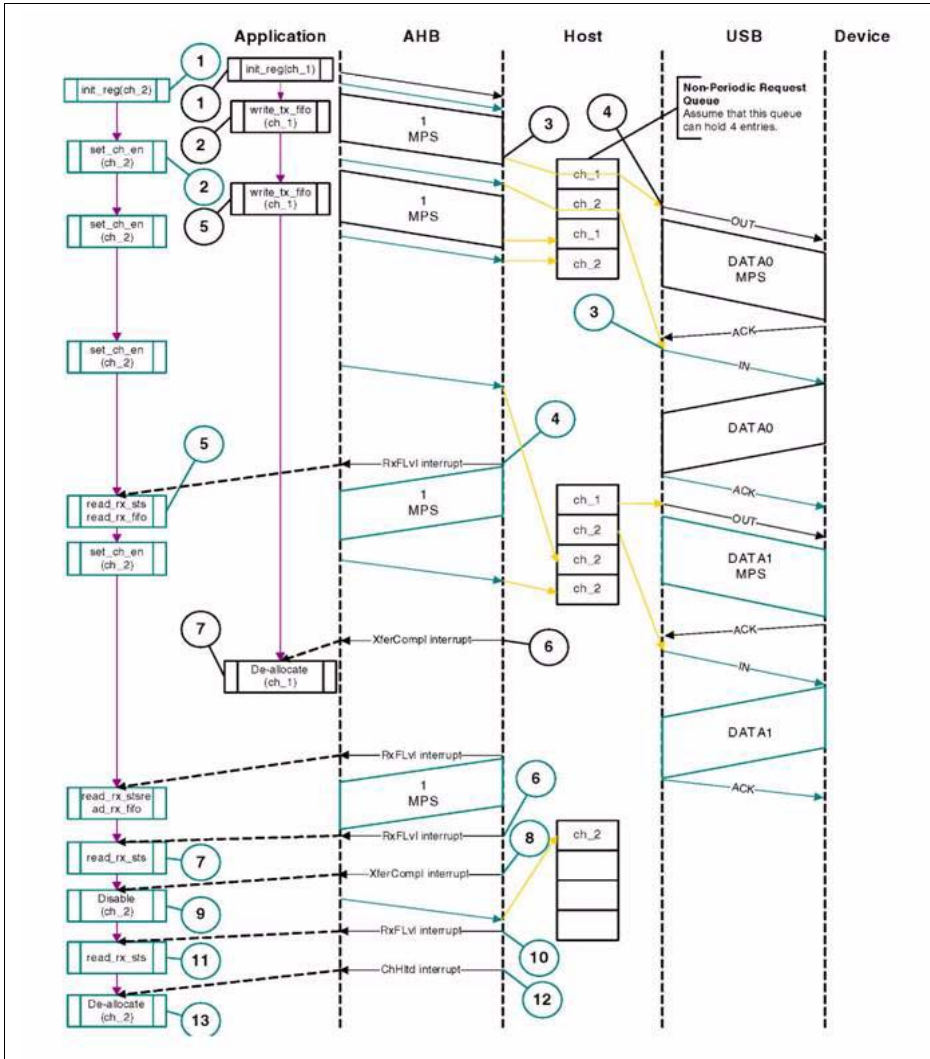


Figure 16-11 Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in Slave Mode

16.5.5.2 Handling Interrupts

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions in Slave mode is shown in the following code samples.

Interrupt Service Routine for Bulk/Control OUT/SETUP Transactions in Slave Mode

Bulk/Control OUT/SETUP

```
Unmask (NAK/XactErr/NYET/STALL/XferCompl)
if (XferCompl)
    {
        Reset Error Count
        Mask ACK
        De-allocate Channel
    }
else if (STALL)
    {
        Transfer Done = 1
        Unmask ChHltd
        Disable Channel
    }
else if (NAK or XactErr or NYET)
    {
        Rewind Buffer Pointers
        Unmask ChHltd
        Disable Channel
        if (XactErr)
            {
                Increment Error Count
                Unmask ACK
            }
        else
            {
                Reset Error Count
            }
    }
else if (ChHltd)
    {
        Mask ChHltd
        if (Transfer Done or (Error_count == 3))
```

```

        {
            De-allocate Channel
        }
    else
        {
            Re-initialize Channel
        }
    }
else if (ACK)
    {
        Reset Error Count
        Mask ACK
    }

```

The application is expected to write the data packets into the transmit FIFO when space is available in the transmit FIFO and the Request queue. The application can make use of GINTSTS.NPTxFEmp interrupt to find the transmit FIFO space.

The application is expected to write the requests as and when the Request queue space is available and until the XferCompl interrupt is received.

The application must clear and never modify the DoPIng bit after enabling the channel and until the XferCompl or ChHltd interrupt is received. The core uses the DoPIng bit to flush the excessive IN requests after receiving the last or short packet.

16.5.6 Interrupt IN Transactions in Slave Mode

A typical interrupt-IN operation in Slave mode is shown in [Figure 16-12](#). See channel 2 (ch_2). The assumptions are:

- The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd. (transfer size = 1,024 bytes).
- The receive FIFO can hold at least one maximum-packet-size packet and two status DWORDs per packet (1,031 bytes for FS).
- Periodic Request Queue depth = 4.

16.5.6.1 Normal Interrupt IN Operation

The sequence of operations in [Figure 16-12](#) (channel 2) is as follows:

1. Initialize channel 2 as explained in **“Channel Initialization in Buffer DMA or Slave Mode” on Page 16-14**. The application must set the HCCHAR2.OddFrm bit.
2. Set the HCCHAR2.ChEna bit to write an IN request to the Periodic Request Queue. For a high- bandwidth interrupt transfer, the application must write the HCCHAR2 register MC (maximum number of expected packets in the next frame) times before switching to another channel.

Universal Serial Bus (USB)

3. The USB host writes an IN request to the Periodic Request Queue for each HCCHAR2 register write with a ChEna bit set.
4. The USB host attempts to send an IN token in the next (odd) frame.
5. As soon as the IN packet is received and written to the receive FIFO, the USB host generates an RxFLvl interrupt.
6. In response to the RxFLvl interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RxFLvl interrupt before reading the receive FIFO, and unmask after reading the entire packet.
7. The core generates the RxFLvl interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (GRXSTSR.PktSts! = 0010_B).
8. The core generates an XferCompl interrupt as soon as the receive packet status is read.
9. In response to the XferCompl interrupt, read the HCTSIZ2.PktCnt field. If HCTSIZ2.PktCnt! = 0, disable the channel (as explained in **“Halting a Channel” on Page 16-15**) before re-initializing the channel for the next transfer, if any). If HCTSIZ2.PktCnt == 0, reinitialize the channel for the next transfer. This time, the application must reset the HCCHAR2.OddFrm bit.

16.5.6.2 Handling Interrupts

The channel-specific interrupt service routine for an interrupt IN transaction in Slave mode is as follows.

Interrupt IN

```

Unmask (NAK/XactErr/XferCompl/BblErr/STALL/FrmOvrn/DataTglErr)
if (XferCompl)
{
    Reset Error Count
    Mask ACK
    if (HCTSIZx.PktCnt == 0)
    {
        De-allocate Channel
    }
    else
    {
        Transfer Done = 1
        Unmask ChHltd
        Disable Channel
    }
}
else if (STALL or FrmOvrn or NAK or DataTglErr or BblErr)

```

```

    {
    Mask ACK
    Unmask ChHltd
    Disable Channel
    if (STALL or BblErr)
        {
        Reset Error Count
        Transfer Done = 1
        }
    else if (!FrmOvrn)
        {
        Reset Error Count
        }
    }
else if (XactErr)
    {
    Increment Error Count
    Unmask ACK
    Unmask ChHltd
    Disable Channel
    }
else if (ChHltd)
    {
    Mask ChHltd
    if (Transfer Done or (Error_count == 3))
        {
        De-allocate Channel
        }
    else Re-initialize Channel (in next b_interval - 1
uF/F)
    }
}
else if (ACK)
    {
    Reset Error Count
    Mask ACK
    }

```

The application is expected to write the requests for the same channel when the Request queue space is available up to the count specified in the MC field before switching to another channel (if any).

16.5.7 Interrupt OUT Transactions in Slave Mode

A typical interrupt OUT operation in Slave mode is shown in [Figure 16-12](#). See channel 1 (ch_1). The assumptions are:

- The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1,024 bytes).
- The Periodic Transmit FIFO can hold one packet (1 KB for FS).
- Periodic Request Queue depth = 4.

16.5.7.1 Normal Interrupt OUT Operation

The sequence of operations in [Figure 16-12](#) (channel 1) is as follows:

1. Initialize and enable channel 1 as explained in [“Channel Initialization in Buffer DMA or Slave Mode” on Page 16-14](#). The application must set the HCCHAR1.OddFrm bit.
2. Write the first packet for channel 1. For a high-bandwidth interrupt transfer, the application must write the subsequent packets up to MC (maximum number of packets to be transmitted in the next frame times before switching to another channel).
3. Along with the last DWORD write of each packet, the USB host writes an entry to the Periodic Request Queue.
4. The USB host attempts to send an OUT token in the next (odd) frame.
5. The USB host generates an XferCompl interrupt as soon as the last packet is transmitted successfully.
6. In response to the XferCompl interrupt, reinitialize the channel for the next transfer.

16.5.7.2 Handling Interrupts

The channel-specific interrupt service routine for Interrupt OUT transactions in Slave mode is shown in the following flow:

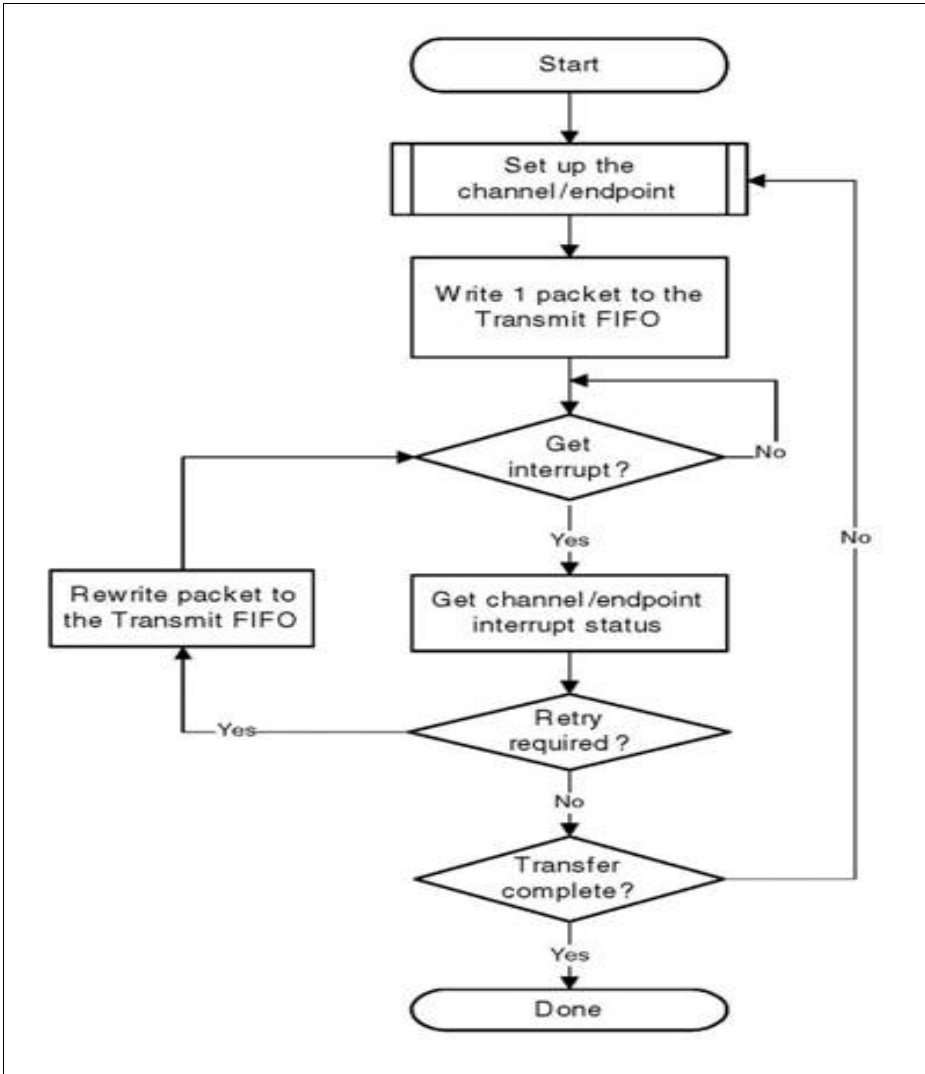


Figure 16-12 Normal Interrupt OUT/IN Transactions in Slave Mode

Interrupt Service Routine for Interrupt OUT Transactions in Slave Mode

Interrupt OUT

```

Unmask (NAK/XactErr/STALL/XferCompl/FrmOvrn)
if (XferCompl)
    {
        Reset Error Count
        Mask ACK
        De-allocate Channel
    }
else if (STALL or FrmOvrn)
    {
        Mask ACK
        Unmask ChHltd
        Disable Channel
        if ( STALL)
            {
                Transfer Done = 1
            }
    }
else if (NAK or XactErr)
    {
        Rewind Buffer Pointers
        Reset Error Count
        Mask ACK
        Unmask ChHltd
        Disable Channel
    }
else if (ChHltd)
    {
        Mask ChHltd
        if (Transfer Done or (Error_count == 3))
            {
                De-allocate Channel
            }
    }
else
    {
        Re-initialize Channel (in next b_interval - 1
uF/F)
    }
    }
else if (ACK)
    {
        Reset Error Count
    }

```



```
Mask ACK
}
```

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the Request queue up to the count specified in the MC field before switching to another channel. The application uses the GINTSTS.NPTxFEmp interrupt to find the transmit FIFO space.

16.5.8 Isochronous IN Transactions in Slave Mode

A typical isochronous IN operation in Slave mode is shown in [Figure 16-13](#). See channel 2 (ch_2). The assumptions are:

- The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame. (transfer size = 1,024 bytes).
- The receive FIFO can hold at least one maximum-packet-size packet and two status DWORDs per packet (1,031 bytes for FS).
- Periodic Request Queue depth = 4.

16.5.8.1 Normal Isochronous IN Operation

The sequence of operations in [Figure 16-13](#) (channel 2) is as follows:

1. Initialize channel 2 as explained in [“Channel Initialization in Buffer DMA or Slave Mode” on Page 16-14](#). The application must set the HCCHAR2.OddFrm bit.
2. Set the HCCHAR2.ChEna bit to write an IN request to the Periodic Request Queue. For a high- bandwidth isochronous transfer, the application must write the HCCHAR2 register MC (maximum number of expected packets in the next frame) times before switching to another channel.
3. The USB host writes an IN request to the Periodic Request Queue for each HCCHAR2 register write with the ChEna bit set.
4. The USB host attempts to send an IN token in the next odd frame.
5. As soon as the IN packet is received and written to the receive FIFO, the USB host generates an RxFLvl interrupt.
6. In response to the RxFLvl interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RxFLvl interrupt before reading the receive FIFO, and unmask it after reading the entire packet.
7. The core generates an RxFLvl interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (GRXSTSR.PktSts!= 0010_B).
8. The core generates an XferCompl interrupt as soon as the receive packet status is read.
9. In response to the XferCompl interrupt, read the HCTSIZ2.PktCnt field. If HCTSIZ2.PktCnt!= 0, disable the channel (as explained in [“Halting a Channel” on](#)

Page 16-15) before re-initializing the channel for the next transfer, if any. If HCTSIZ2.PktCnt == 0, reinitialize the channel for the next transfer. This time, the application must reset the HCCHAR2.OddFrm bit.

16.5.8.2 Handling Interrupts

The channel-specific interrupt service routine for an isochronous IN transaction in Slave mode is as follows.

Isochronous IN

```

Unmask (XactErr/XferCompl/FrmOvrn/BblErr)
if ( XferCompl or FrmOvrn)
    {
    if (XferCompl and (HCTSIZx.PktCnt == 0))
        {
        Reset Error Count
        De-allocate Channel
        }
    else
        {
        Unmask ChHltd
        Disable Channel
        }
    }
else if (XactErr or BblErr)
    {
    Increment Error Count
    Unmask ChHltd
    Disable Channel
    }
else if (ChHltd)
    {
    Mask ChHltd
    if (Transfer Done or (Error_count == 3))
        {
        De-allocate Channel
        }
    else
        {
        Re-initialize Channel
        }
    }
}

```

16.5.9 Isochronous OUT Transactions in Slave Mode

A typical isochronous OUT operation in Slave mode is shown in [Figure 16-13](#). See channel 1 (ch_1). The assumptions are:

- The application is attempting to send one packet every frame (up to 1 maximum packet size), starting with an odd frame. (transfer size = 1,024 bytes).
- The Periodic Transmit FIFO can hold one packet (1 KB for FS).
- Periodic Request Queue depth = 4.

16.5.9.1 Normal Isochronous OUT Operation

The sequence of operations in [Figure 16-13](#) (channel 1) is as follows:

1. Initialize and enable channel 1 as explained in [“Channel Initialization in Buffer DMA or Slave Mode” on Page 16-14](#). The application must set the HCCHAR1.OddFrm bit.
2. Write the first packet for channel 1. For a high-bandwidth isochronous transfer, the application must write the subsequent packets up to MC (maximum number of packets to be transmitted in the next frame) times before switching to another channel.
3. Along with the last DWORD write of each packet, the USB host writes an entry to the Periodic Request Queue.
4. The USB host attempts to send the OUT token in the next frame (odd).
5. The USB host generates the XferCompl interrupt as soon as the last packet is transmitted successfully.
6. In response to the XferCompl interrupt, reinitialize the channel for the next transfer.

16.5.9.2 Handling Interrupts

The channel-specific interrupt service routine for isochronous OUT transactions in Slave mode is shown in the following flow:

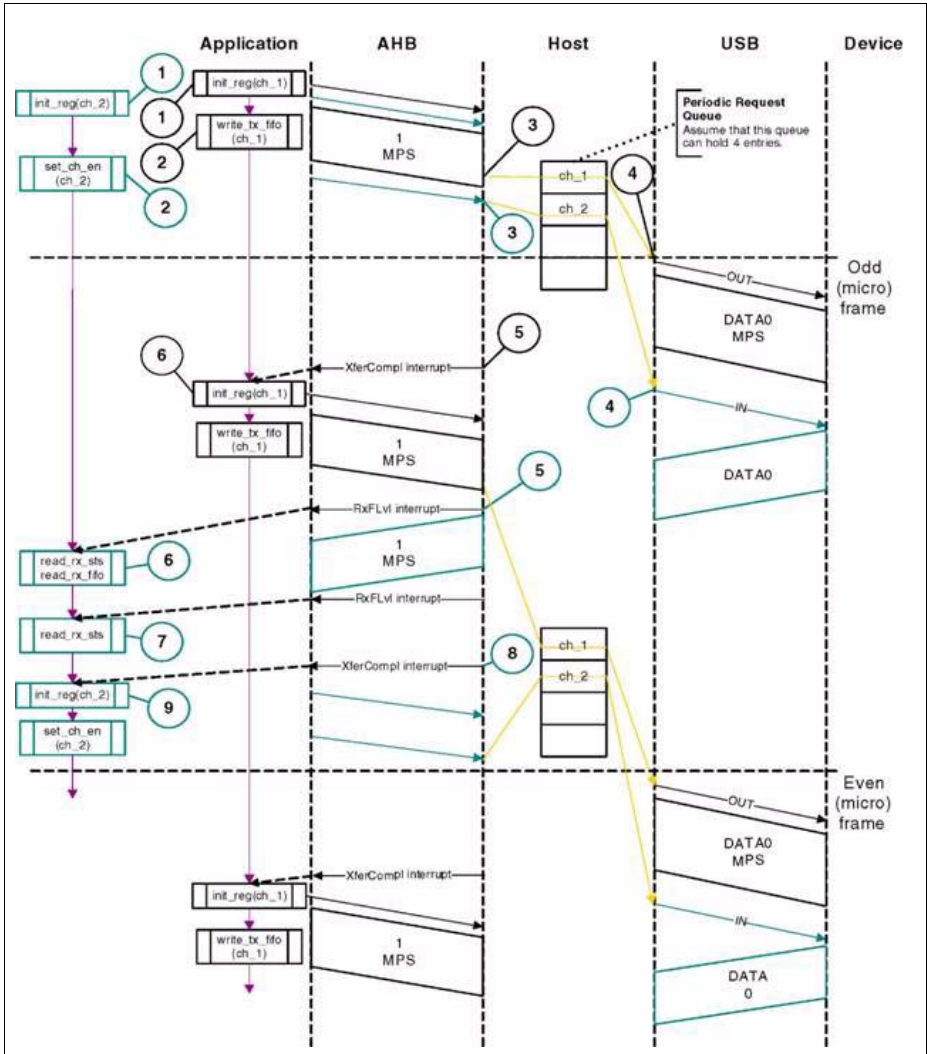


Figure 16-13 Normal Isochronous OUT/IN Transactions in Slave Mode

Interrupt Service Routine for Isochronous OUT Transactions in Slave Mode**Isochronous OUT**

```
Unmask (FrmOvrn/XferCompl)
if (XferCompl)
    {
        De-allocate Channel
    }
else if (FrmOvrn)
    {
        Unmask ChHltd
        Disable Channel
    }
else if (ChHltd)
    {
        Mask ChHltd
        De-allocate Channel
    }
```

16.6 Host Programming in Buffer DMA Mode

This section discusses how to program the Host core when it is in Buffer DMA mode.

16.6.1 Control Transactions in Buffer DMA Mode

Setup, Data, and Status stages of a control transfer must be performed as three separate transfers. Setup- and Data- or Status-stage OUT transactions are performed similarly to the bulk OUT transactions explained in [“Bulk and Control OUT/SETUP Transactions in Buffer DMA Mode” on Page 16-40](#). Data- or Status-stage IN transactions are performed similarly to the bulk IN transactions explained in [“Bulk and Control IN Transactions in Buffer DMA Mode” on Page 16-38](#). For all three stages, the application is expected to set the HCCHAR1.EPType field to Control. During the Setup stage, the application is expected to set the HCTSIZ1.PID field to SETUP.

16.6.2 Bulk and Control IN Transactions in Buffer DMA Mode

A typical bulk or control IN operation in DMA mode is shown in [Figure 16-14](#). See channel 2 (ch_2).

The assumptions are:

- The application is attempting to receive two maximum-packet-size packets (transfer size = 1,024 bytes).

Universal Serial Bus (USB)

- The receive FIFO can hold at least one maximum-packet-size packet and two status DWORDs per packet (72 bytes for FS).
- The Non-periodic Request Queue depth = 4.

16.6.2.1 Normal Bulk and Control IN Operations

The sequence of operations in **Figure 16-14** (channel 2) is as follows:

1. Initialize and enable channel 2 as explained in “**Channel Initialization in Buffer DMA or Slave Mode**” on **Page 16-14**.
2. The USB host writes an IN request to the Request queue as soon as channel 2 receives the grant from the arbiter. (Arbitration is performed in a round-robin fashion, with fairness.).
3. The USB host starts writing the received data to the system memory as soon as the last byte is received with no errors.
4. When the last packet is received, the USB host sets an internal flag to remove any extra IN requests from the Request queue.
5. The USB host flushes the extra requests.
6. The final request to disable channel 2 is written to the Request queue. At this point, channel 2 is internally masked for further arbitration.
7. The USB host generates the ChHltd interrupt as soon as the disable request comes to the top of the queue.
8. In response to the ChHltd interrupt, de-allocate the channel for other transfers.

16.6.2.2 Handling Interrupts

The channel-specific interrupt service routine for bulk and control IN transactions in DMA mode is shown in the following flow:

Interrupt Service Routines for Bulk/Control Bulk/Control IN Transactions in DMA Mode

Bulk/Control IN

```

Unmask (ChHltd)
    if (ChHltd) {
        if (XferComp1 or STALL or BblErr) {
            Reset Error Count Mask ACK De-allocate Channel }
        else if (XactErr) {
            if (Error_count == 2) {
                De-allocate Channel
            }
        }
        else {

```

```

        Unmask ACK
        Unmask NAK
        Unmask DataTglErr
        Increment Error
        Count Re-initialize Channel
    }
}
}
else if (ACK or NAK or DataTglErr) {
    Reset Error Count
    Mask ACK
    Mask NAK Mask DataTglErr
}

```

The application must clear and never modify the DoPing bit after enabling the channel and until the ChHltd interrupt is received. The core uses the DoPing excessive IN requests after receiving the last or short packet.

16.6.3 Bulk and Control OUT/SETUP Transactions in Buffer DMA Mode

16.6.3.1 Overview

- The application is attempting to send two maximum-packet-size packets (transfer size = 1,024 bytes).
- The Non-periodic Transmit FIFO can hold two packets (128 bytes for FS).
- The Non-periodic Request Queue depth = 4.

16.6.3.2 Normal Bulk and Control OUT/SETUP Operations

The sequence of operations in [Figure 16-11](#) (channel 1) is as follows:

1. Initialize and enable channel 1 as explained in [“Channel Initialization in Buffer DMA or Slave Mode” on Page 16-14](#).
2. The USB host starts fetching the first packet as soon as the channel is enabled. For internal DMA mode, the USB host uses the programmed DMA address to fetch the packet.
3. After fetching the last DWORD of the second (last) packet, the USB host masks channel 1 internally for further arbitration.
4. The USB host generates a ChHltd interrupt as soon as the last packet is sent.
5. In response to the ChHltd interrupt, de-allocate the channel for other transfers.

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions in DMA mode is shown in [“Handling Interrupts” on Page 16-43](#).

16.6.3.3 NAK and NYET Handling With Internal DMA

1. The USB Host sends a Bulk OUT Transaction.
2. The Device responds with NAK or NYET.
3. If the application has unmasked NAK or NYET, the core generates the corresponding interrupt(s) to the application.
The application is not required to service these interrupts, since the core takes care of rewinding of buffer pointers and re-initializing the Channel without application intervention.
4. The core automatically issues a ping token.
5. When the Device returns an ACK, the core continues with the transfer.

Note: The application must use the Do Ping bit to set the ping bit (HCTSIZ0[31]) for the next transfer and not rely on the NYET status. This ensures that the last response sent from the device (NYET/ACK) does not matter for a new transfer.

Optionally, the application can utilize these interrupts. If utilized by the application:

- The NAK or NYET interrupt is masked by the application.
- The core does not generate a separate interrupt when NAK or NYET is received by the Host functionality.

Application Programming Flow

1. The application programs a channel to do a bulk transfer for a particular data size in each transaction.
 - a) Packet Data size can be up to 512KBytes
 - b) Zero-length data must be programmed as a separate transaction.
2. Program the transfer size register with:
 - a) Transfer size
 - b) Packet Count
3. Program the DMA address.
4. Program the HCCHAR to enable the channel.
5. The application is not required to set the HCCHARx.DoPng bit for NAK/NYET responses. The core sends a Ping token automatically when the device responds with a NAK/NYET for OUT transfers. The core keeps sending the Ping token until an ACK response is received.
6. The Interrupt handling by the application is as depicted in the flow diagram.

Note: The NAK/NYET interrupts are still generated internally. The application can mask off these interrupts from reaching it. The application can use these interrupts optionally

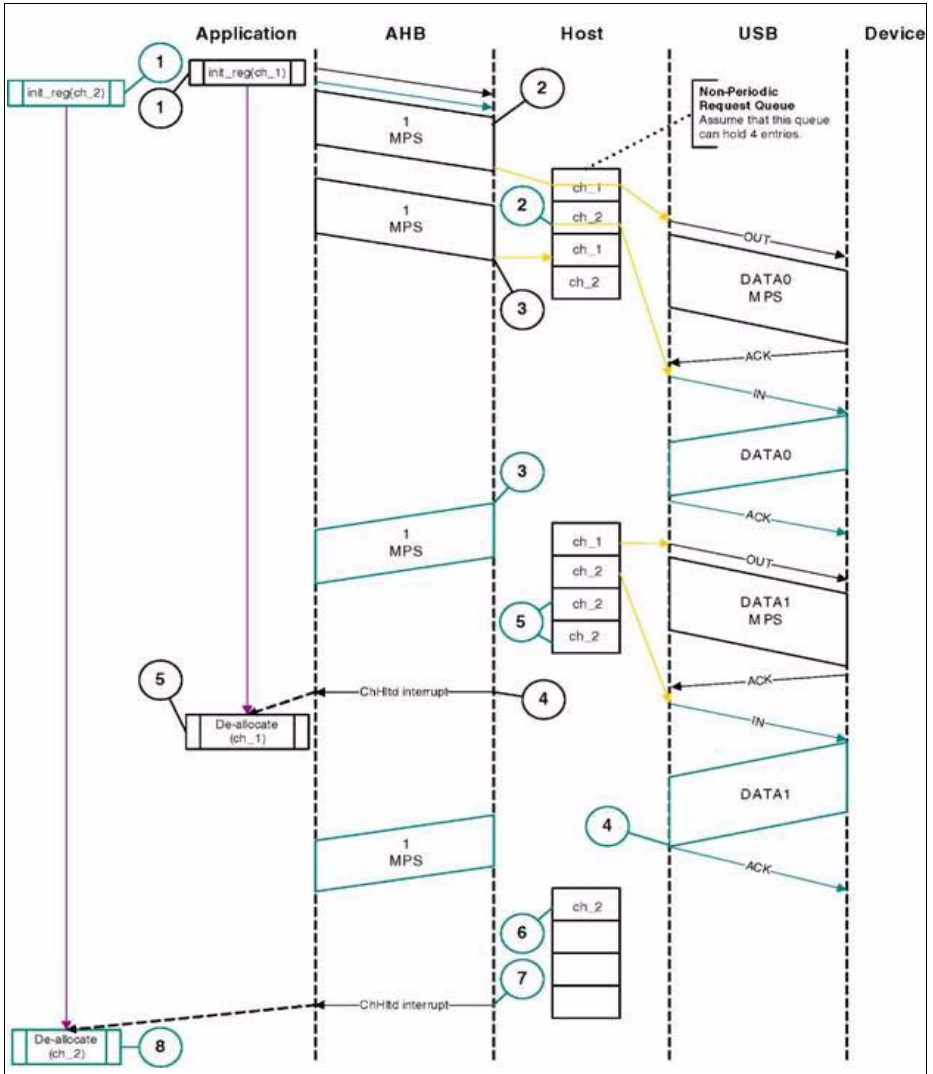


Figure 16-14 Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in DMA Mode

16.6.3.4 Handling Interrupts

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions in DMA mode is shown in the following code samples.

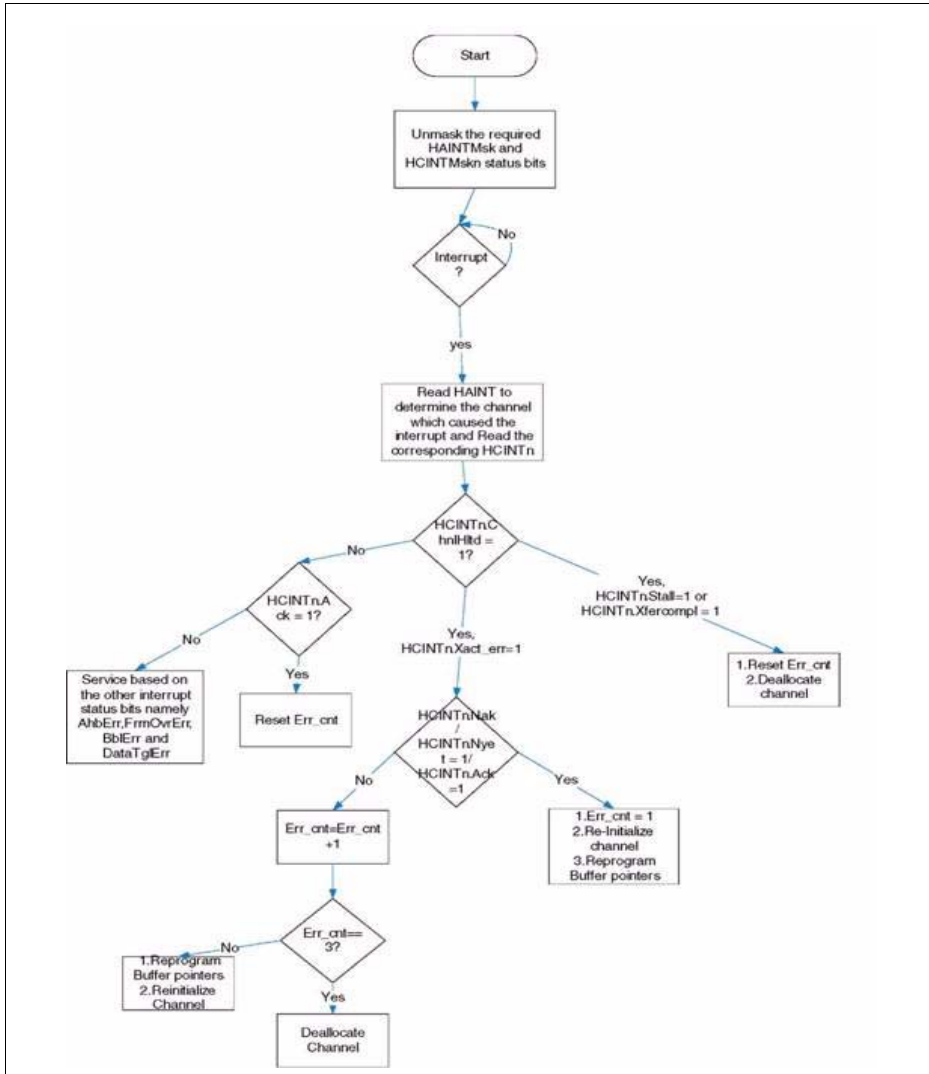


Figure 16-15 Interrupt Service Routine for Bulk/Control OUT Transaction in DMA Mode

In **Figure 16-15** that the Interrupt Service Routine is not required to handle NAK or NYET responses. The core internally sets the HCCHARx.DoPng bit once a NAK/NYET is received. The HCCHARx.DoPng is cleared only when the Ping token receives an ACK response. The application is not required to set the HCCHARx.DoPng bit for NAK/NYET scenarios. This is the difference of proposed flow with respect to current flow. Similar flow is applicable for Control flow also.

The NAK/NYET status bits in HCINTx registers are updated. The application can unmask these interrupts when it requires the core to generate an interrupt for NAK/NYET. The NAK/NYET status is updated because during Xact_err scenarios, this status provides a means for the application to determine whether the Xact_err occurred three times consecutively or there were NAK/NYET responses in between two Xact_err. This provides a mechanism for the application to reset the error counter accordingly. The application must read the NAK / NYET /ACK along with the xact_err. If NAK / NYET /ACK is not set, the Xact_err count must be incremented otherwise application must initialize the Xact_err count to 1.

Bulk/Control OUT/SETUP

Unmask (ChHltd)

if (ChHltd)

```

    {
        if (XferCompl or STALL)
        {
            Reset Error Count (Error_count=1)
            Mask ACK
            De-allocate Channel
        }
        else if (XactErr)
        {
            if (Nak/Nyet/Ack)
            {
                Error_count = 1
                Re-initialize Channel
                Rewind Buffer Pointers }
            }
        else
        {
            Error_count = Error_count + 1
            if (Error_count == 3)
            {
                De allocate channel
            }
        }
        else
    }

```

```

        {
            Re-initialize Channel
            Rewind Buffer Pointers
        }
    }
}
else if (ACK)
{
    Reset Error Count (Error_count=1)
    Mask ACK
}

```

As soon as the channel is enabled, the core attempts to fetch and write data packets, in multiples of the maximum packet size, to the transmit FIFO when space is available in the transmit FIFO and the Request queue. The core stops fetching as soon as the last packet is fetched.

While continuing the transfer to a high-speed device, the application must set the DoPing bit before enabling the channel if the previous transaction ended with XacrErr response. In this case, the core starts with the ping protocol, then automatically switches to Data Transfer mode.

16.6.4 Interrupt IN Transactions in Buffer DMA Mode

A typical interrupt IN operation in DMA mode is shown in [Figure 16-16](#). See channel 2 (ch_2). The assumptions are:

- The application is attempting to receive one packet in every frame (up to 1 maximum packet size of 1,024 bytes).
- The receive FIFO can hold at least one maximum-packet-size packet and two status DWORDs per packet (1,032 bytes for FS).
- Periodic Request Queue depth = 4.

16.6.4.1 Normal Interrupt IN Operation

The sequence of operations in [Figure 16-16](#) on [Page 16-48](#) (channel 2) is as follows:

1. Initialize and enable channel 2 as explained in [“Channel Initialization in Buffer DMA or Slave Mode” on Page 16-14](#).
2. The USB host writes an IN request to the Request queue as soon as the channel 2 gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the USB host writes consecutive writes up to MC times.
3. The USB host attempts to send an IN token at the beginning of the next (odd) frame.
4. As soon the packet is received and written to the receive FIFO, the USB host generates a ChHltd interrupt.

5. In response to the ChHltd interrupt, reinitialize the channel for the next transfer.

16.6.4.2 Handling Interrupts

The channel-specific interrupt service routine for Interrupt IN transactions in DMA mode is as follows.

Interrupt Service Routine for Interrupt IN Transactions in DMA Mode

```

Unmask (ChHltd)
if (ChHltd)
    {
        if (XferCompl)
            {
                Reset Error Count
                Mask ACK
                if (Transfer Done)
                    {
                        De-allocate Channel
                    }
                else
                    {
                        Re-initialize Channel (in next b_interval -
1 uF/F)
                    }
            }
        else if (STALL or BblErr)
            {
                Reset Error Count
                Mask ACK
                De-allocate Channel
            }
        else if (NAK or DataTglErr or FrmOvrnun)
            {
                Mask ACK
                Re-initialize Channel (in next b_interval - 1 uF/F)
                if (DataTglErr or NAK)
                    {
                        Reset Error Count
                    }
            }
        else if (XactErr)
            {
                if (Error_count == 2)

```

```

        {
        De-allocate Channel
        }
    else
    {
        Increment Error Count
        Unmask ACK
        Re-initialize Channel (in next b_interval - 1
uF/F)
    }
}
}
else if (ACK)
{
    Reset Error Count
    Mask ACK

```

As soon as the channel is enabled, the core attempts to write the requests into the Request queue when the space is available up to the count specified in the MC field.

16.6.5 Interrupt OUT Transactions in Buffer DMA Mode

A typical interrupt OUT operation in DMA mode is shown in [Figure 16-16](#). See channel 1 (ch_1). The assumptions are:

- The application is attempting to transmit one packet in every frame (up to 1 maximum packet size of 1,024 bytes).
- The Periodic Transmit FIFO can hold one packet (1 KB for FS).
- Periodic Request Queue depth = 4.

16.6.5.1 Normal Interrupt OUT Operation

1. Initialize and enable channel 1 as explained in [“Channel Initialization in Buffer DMA or Slave Mode” on Page 16-14](#).
2. The USB host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last DWORD fetch. In high-bandwidth transfers, the USB host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.
3. The USB host attempts to send the OUT token in the beginning of the next odd frame.
4. After successfully transmitting the packet, the USB host generates a ChHltd interrupt.
5. In response to the ChHltd interrupt, reinitialize the channel for the next transfer.

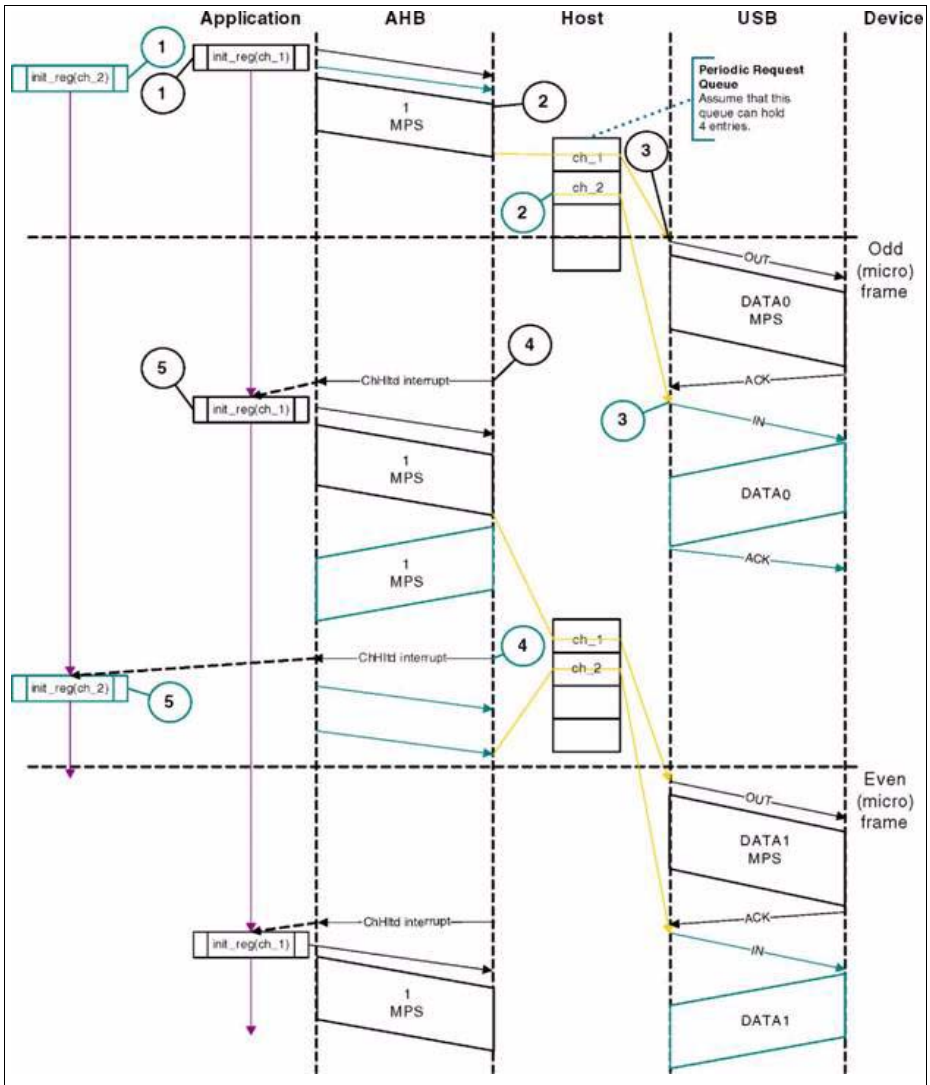


Figure 16-16 Normal Interrupt OUT/IN Transactions in DMA Mode

16.6.5.2 Handling Interrupts

The following code sample shows the channel-specific ISR for an interrupt OUT transaction in DMA mode.

Interrupt Service Routine for Interrupt OUT Transactions in DMA Mode

Interrupt OUT

```

Unmask (ChHltd)
    if (ChHltd)
        {
            if (XferCompl)
                {
                    Reset Error Count
                    Mask ACK
                    if (Transfer Done)
                        {
                            De-allocate Channel
                        }
                    else
                        {
                            Re-initialize Channel (in next b_interval -
1 uF/F)
                        }
                }
            else if (STALL)
                {
                    Transfer Done = 1
                    Reset Error Count
                    Mask ACK
                    De-allocate Channel
                }
            else if (NAK or FrmOvrn)
                {
                    Mask ACK
                    Rewind Buffer Pointers
                    Re-initialize Channel (in next b_interval - 1
uF/F)

                    if (NAK)
                        {
                            Reset Error Count

```



```

        }
    }
else if (XactErr)
{
    if (Error_count == 2)
    {
        De-allocate Channel
    }
else
    {
        Increment Error Count
        Rewind Buffer Pointers
        Unmask ACK
        Re-initialize Channel (in next b_interval - 1
uF/F)
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}

```

As soon as the channel is enabled, the core attempts to fetch and write data packets, in maximum packet size multiples, to the transmit FIFO when the space is available in the transmit FIFO and the Request queue. The core stops fetching as soon as the last packet is fetched (the number of packets is determined by the MC field of the HCCHARx register).

16.6.6 Isochronous IN Transactions in Buffer DMA Mode

A typical isochronous IN operation in DMA mode is shown in [Figure 16-17](#). See channel 2 (ch_2). The assumptions are:

- The application is attempting to receive one packet in every frame (up to 1 maximum packet size of 1,024 bytes).
- The receive FIFO can hold at least one maximum-packet-size packet and two status DWORDS per packet (1,032 bytes for FS).
- Periodic Request Queue depth = 4.

16.6.6.1 Normal Isochronous IN Operation

The sequence of operations in [Figure 16-17](#) (channel 2) is as follows:

1. Initialize and enable channel 2 as explained in “**Channel Initialization in Buffer DMA or Slave Mode**” on Page 16-14.
2. The USB host writes an IN request to the Request queue as soon as the channel 2 gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the USB host performs consecutive writes up to MC times.
3. The USB host attempts to send an IN token at the beginning of the next (odd) frame.
4. As soon the packet is received and written to the receive FIFO, the USB host generates a ChHltd interrupt.
5. In response to the ChHltd interrupt, reinitialize the channel for the next transfer.

16.6.6.2 Handling Interrupts

The channel-specific interrupt service routine for an isochronous IN transaction in DMA mode is as follows.

Isochronous IN

```

Unmask (ChHltd)
if (ChHltd)
    {
        if ( XferCompl or FrmOvrn)
            {
                if (XferCompl and (HCTSIZx.PktCnt == 0))
                    {
                        Reset Error Count
                        De-allocate Channel
                    }
                else
                    {
                        De-allocate Channel
                    }
            }
        else if (XactErr or BblErr)
            {
                if (Error_count == 2)
                    {
                        De-allocate Channel
                    }
                else
                    {
                        Increment Error Count
                        Re-enable Channel (in next b_interval - 1 uF/F)
                    }
            }
    }

```

}

16.6.7 Isochronous OUT Transactions in Buffer DMA Mode

A typical isochronous OUT operation in DMA mode is shown in [Figure 16-17](#). See channel 1 (ch_1). The assumptions are:

- The application is attempting to transmit one packet every frame (up to 1 maximum packet size of 1,024 bytes).
- The Periodic Transmit FIFO can hold one packet (1 KB for FS).
- Periodic Request Queue depth = 4.

16.6.7.1 Normal Isochronous OUT Operation

1. Initialize and enable channel 1 as explained in [“Channel Initialization in Buffer DMA or Slave Mode” on Page 16-14](#).
2. The USB host starts fetching the first packet as soon as the channel is enabled, and writes the OUT request along with the last DWORD fetch. In high-bandwidth transfers, the USB host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.
3. The USB host attempts to send an OUT token in the beginning of the next (odd) frame.
4. After successfully transmitting the packet, the USB host generates a ChHltd interrupt.
5. In response to the ChHltd interrupt, reinitialize the channel for the next transfer.

16.6.7.2 Handling Interrupts

The channel-specific interrupt service routine for Isochronous OUT transactions in DMA mode is shown in the following flow:

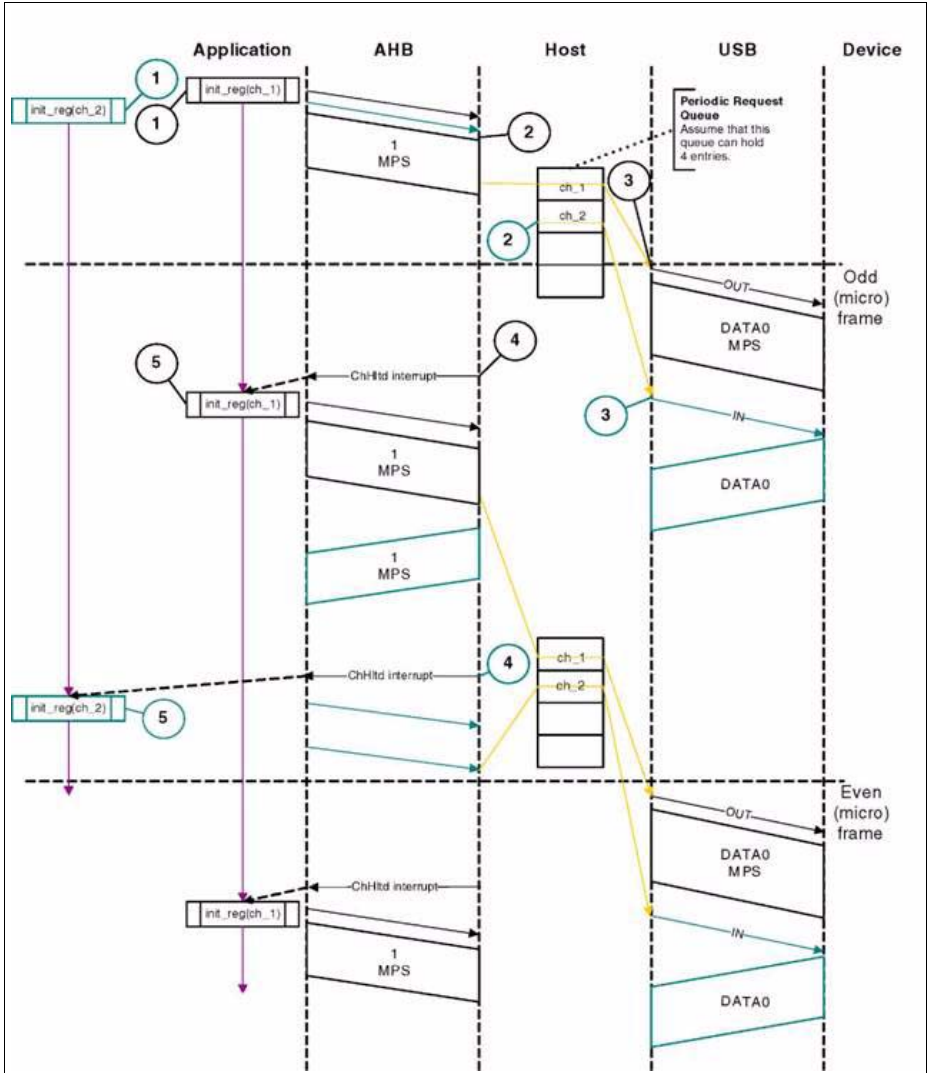


Figure 16-17 Normal Isochronous OUT/IN Transactions in DMA Mode

Interrupt Service Routine for Isochronous OUT Transactions in DMA Mode

Isochronous OUT

Unmask (ChHltd)

```
if (ChHltd)
{
    if (XferCompl or FrmOvrn)
    {
        De-allocate Channel
    }
}
```

16.7 Host Programming in Scatter-Gather DMA Mode

This section describes the programming requirements for the USB core operating in Scatter-Gather (descriptor) DMA host mode. It provides information on programming the core to perform asynchronous transfers (bulk and control), and periodic transfers (isochronous and interrupt).

16.7.1 Programming Requirements

Consider the following points when using the host core in scatter-gather DMA mode:

- USB core supports non-DWORD aligned address access in Scatter/Gather DMA in Host mode only
- NAK/NYET scenario is handled by USB core in Scatter/Gather DMA mode without the application's intervention.
- CONCAT mode is not supported for any of the flows, that is, a single packet cannot span more than one descriptor.

16.7.2 SPRAM Requirements

For each channel, the current descriptor pointer and descriptor status are cached to avoid additional requests to system memory. These are stored in the SPRAM. In addition, the HCDMAx registers are also implemented in the SPRAM.

16.7.2.1 Descriptor Memory Structures

In Scatter/Gather DMA mode, the core implements a true scatter-gather memory distribution in which data buffers are scattered over the system memory. However, the descriptors themselves are continuous. Each channel memory structure is implemented as a contiguous list of descriptors; each descriptor points to a data buffer of predefined size. In addition to the buffer pointer (1 DWORD), the descriptor also has a status quadlet (1 DWORD). When the list is implemented as a ring buffer, the list processor switches to the first element of the list when it encounters last bit. All channels (control, bulk, interrupt, and isochronous) implement these structures in memory.

Note: The descriptors are stored in continuous locations. For example, descriptor 1 is stored in $0000'0000_H$, descriptor 2 is stored in $0000'0008_H$, descriptor 3 in $0000'0010_H$ and so on. The descriptors are always DWORD aligned.

The descriptor memory structures are displayed in **Figure 16-18**.

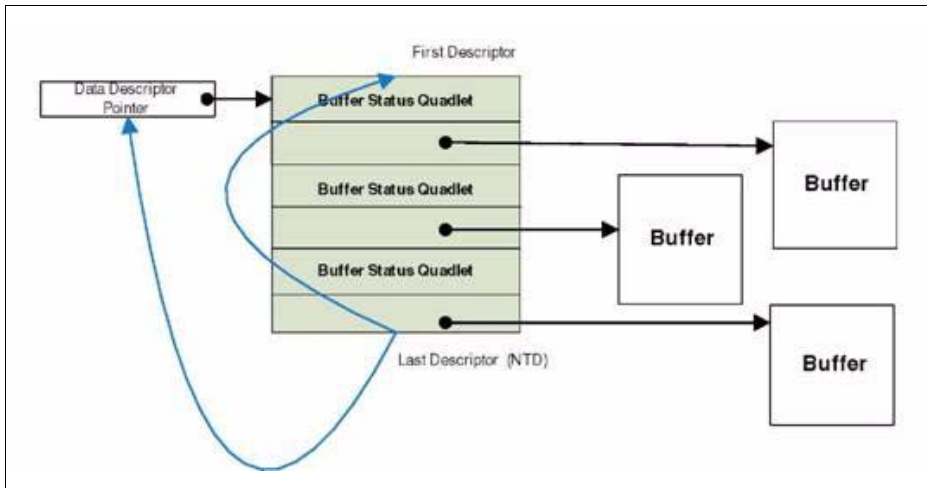


Figure 16-18 Descriptor Memory Structures

All channels must implement the following memory structure:

- Each channel has one memory structure
- Each data buffer must have a descriptor associated with it to provide the status of the buffer. The buffer itself contains only raw data.
- Each buffer descriptor is two quadlets in length. When the descriptor is ready, the DMA fetches and processes its data buffer. The buffers to which the descriptor points hold packet data for non- isochronous channels and packet data corresponding to the frame data for isochronous channels.
- The handshake between the application and core is accomplished by the Active Bit field in the status quadlet of the descriptor as described below:
- A=1 indicates that the descriptor is ready.
- A=0 indicates that the descriptor is not ready.

Universal Serial Bus (USB)

The IN and OUT data memory structures are shown in **Figure 16-19**. The figure shows the definition of status quadlet bits for non-ISO and ISO channels.

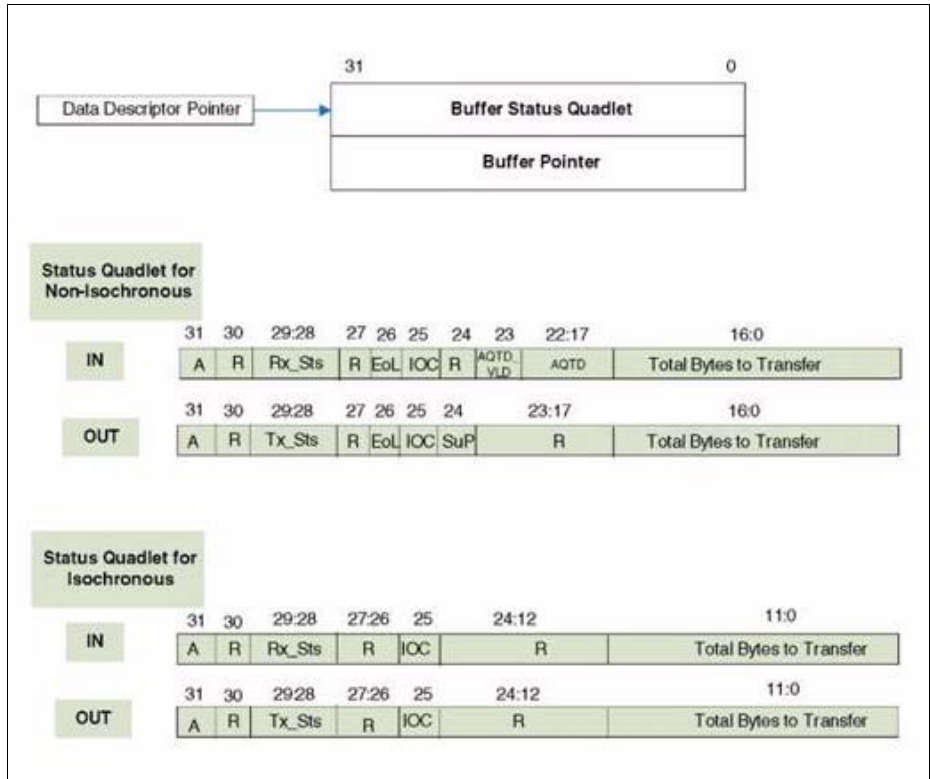


Figure 16-19 Memory Structure

Universal Serial Bus (USB)

In addition, a Frame list in memory for Isochronous and Interrupt channels contains information on the channels that need to be scheduled in a frame. For periodic channels, USB core reads the list corresponding to the frame number and schedules the channel that has Ch_sch=1 in the appropriate frame. **Figure 16-20** shows the frame list for periodic channels.

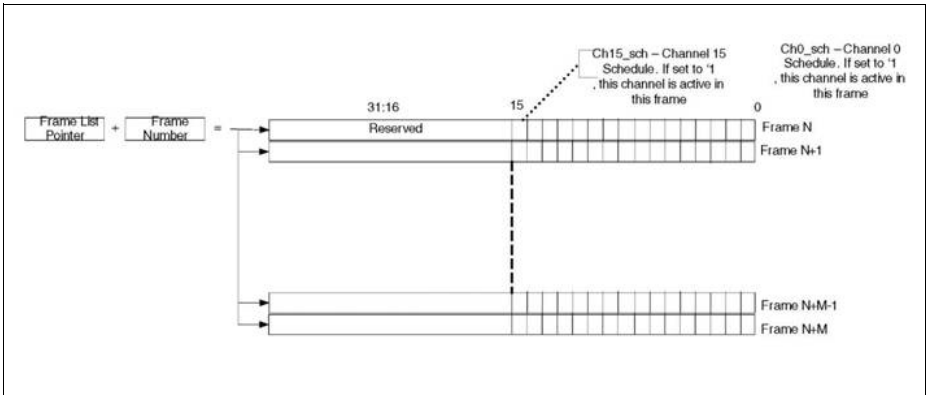


Figure 16-20 Frame List for Periodic Channels

16.7.2.2 IN Memory Structure

All channels that support IN direction transactions (channels that receive data from the USB device) must implement a memory structure with the following characteristics:

- Each data buffer must have a descriptor associated with it to provide the status of the buffer. The buffer itself contains only raw data.
- Each buffer descriptor is two quadlets in length. [Table 16-3](#) displays the IN Data Memory Structure fields.

Table 16-3 IN Data Memory Structure Values

Bit	Bit ID	Description
A[31]	Active Bit	<p>This 1-bit value indicates whether the descriptor is ready.</p> <p>For non-isochronous channels, this bit indicates the following:</p> <ul style="list-style-type: none"> 0_B Descriptor is not ready 1_B Descriptor is ready. USB core can start processing the descriptor. <p>For Isochronous channels, this bit indicates the following:</p> <ul style="list-style-type: none"> 0_B Isochronous channel is not scheduled for the corresponding frame/frame. 1_B Isochronous channel is not scheduled for the corresponding frame/frame. <p>The application sets this bit when the descriptor is ready. USB core resets this bit while closing the descriptor. The application needs to set this bit as a last step after the entire descriptor is ready. The core resets this bit as a final step of processing the descriptor. This bit is accessed by both the core and the application.</p>
Rx Sts [29:28]	Receive Status	<p>This 2-bit field describes the status of the received data. The core updates this field when the descriptor is closed. PKTERR is set by the core when there was an error while receiving the packet. When updated with PKTERR, it is an indication that IN data has been received with errors. The error includes Xact_err scenarios. BUFERR is set by the core when AHB error is encountered during buffer access. The possible combinations are:</p> <ul style="list-style-type: none"> • 00_B Success, No AHB or packet errors • 01_B PKTERR. • 10_B Reserved • 11_B Reserved <p>This field has to be initialized to 00_B by the application and updated by the core subsequently.</p>

Universal Serial Bus (USB)

Table 16-3 IN Data Memory Structure Values (cont'd)

Bit	Bit ID	Description	
EoL [26]	End of List	For Non Isochronous, it indicates that this is the last descriptor in the list, if set. The core does not generate a BNA interrupt for the next descriptor, if it is unavailable. For Isochronous, this field is reserved. This field is controlled by the application.	
IOC [25]	Interrupt On complete	Set by the application, this bit indicates that the core must generate a transfer complete interrupt (XferCompl) after this descriptor is finished.	
[24]	Varies	Non Isochronous Reserved	Isochronous Bit: [24:23] Bit ID: R: Reserved
[23]	Varies	Non Isochronous IN Bit: [23] Bit ID: AQTD_VALID Alternate Queue Transfer Descriptor Valid. When set by the application, if a Short packet is received, the core jumps to a new descriptor in the same list. The new descriptor address is obtained by replacing the CTD value of the corresponding channel with the AQTD value. When the application resets this bit, the core ignores AQTD.	
[22:17]	Varies	Non Isochronous IN Bit: [23] Bit ID: AQTD_VALID Alternate Queue Transfer Descriptor Valid. This is valid only if AQTD_VALID is set. This field gives the offset value in DWORDS. The core will use this offset to jump to a new descriptor address in the same list.	Isochronous IN Bit Bit: [22:12] Bit ID: R Reserved

Universal Serial Bus (USB)

Table 16-3 IN Data Memory Structure Values (cont'd)

Bit	Bit ID	Description
[16:12]	Varies	Non Isochronous IN
[11]	Varies	<p>Bit: [16:0] Bit ID: Total bytes to transfer This 17-bit value can take values from 0 to 128K-1 bytes, depending on the transfer size of data received from the USB device.</p> <p>The application programs the expected transfer size. When the descriptor is closed, this indicates remainder of the transfer size. This field must be in multiple of MPS for the corresponding end point.</p> <p>The MPS for the various packet types are as follows:</p> <ul style="list-style-type: none"> • Control - <ul style="list-style-type: none"> – LS - 8 bytes – FS - 8,16,32,64 bytes • Bulk – <ul style="list-style-type: none"> – FS - 8,16,32,64 bytes • Interrupt <ul style="list-style-type: none"> – LS - up to 8 bytes – FS – up to 64 bytes
[10:0]	Varies	<p>Isochronous IN Bit: 11:0 Bit ID: Received</p> <p>Isochronous IN Bit: [11:0] Bit ID: Total bytes to transfer This 11-bit value can take values from 0 to 4K bytes, depending on the packet size of data received from the USB host. The application programs the expected transfer size. When the descriptor is closed, it indicates remainder of the transfer size. The maximum payload size of each ISO packet as per USB specification 2.0 is as follows.</p> <ul style="list-style-type: none"> • FS - up to 1023 bytes <p><i>Note: Note: A value of 0 indicates zero bytes of data, 1 indicates 1 byte of data, and so on.</i></p>

Table 16-4 displays the out buffer pointer field description.

Table 16-4 IN Buffer Pointer

Buf Addr[31:0]	Buffer Address	The Buffer pointer field in the descriptor is 32 bits wide and contains the address where the received data is to be stored in the system memory. The buffer address does not need to be aligned with DWORD.
----------------	----------------	--

16.7.2.3 OUT Memory Structure

All channels that support OUT direction transactions (channels that transmit data to the USB device) must implement the following memory structure:

- Each buffer must have a descriptor associated with it.
- The application fills the data buffer, updates its status in the descriptor, and enables the channel.
- The DMA fetches this descriptor and processes it, moving on in this manner until it reaches the end of the descriptor chain.
- The buffer to which the descriptor points to holds packet data for non-isochronous channels and frame data for isochronous channels.

Table 16-5 displays the OUT Data Memory Structure fields. Bits that are not present are reserved to be set to zero by the application for writes and ignored during reads.

Table 16-5 OUT Data Memory Structure Values

Bit	Bit ID	Description
A[31]	Active Bit	<p>This 1-bit value indicates whether the descriptor is ready. For non-isochronous channels, this bit indicates the following:</p> <ul style="list-style-type: none"> 0_B Descriptor is not ready 1_B Descriptor is ready. USB core can start processing the descriptor. <p>For Isochronous channels, this bit indicates the following:</p> <ul style="list-style-type: none"> 0_B Isochronous channel is not scheduled for the corresponding frame. 1_B Isochronous channel is not scheduled for the corresponding frame. <p>The application sets this bit when the descriptor is ready. USB core resets this bit while closing the descriptor. The application needs to set this bit as a last step after the entire descriptor is ready. The core resets this bit as a final step of processing the descriptor.</p>
Tx Sts [29:28]	Transmit Status	<p>The status of the transmitted data. This reflects if the OUT data has been transmitted correctly or with errors. BUFERR is set by core when there is a AHB error during buffer access along with asserting AHBERR interrupt (HCINTx register) for the corresponding channel.</p> <p>PKTERR is set by the core when there was an error while transmitting the packet. The error includes Xact_err scenarios.</p> <p>The possible combinations are as follows:</p> <ul style="list-style-type: none"> 00_B Success, No AHB errors 01_B PKTERR 10_B Reserved 11_B Reserved <p>This field has to be initialized to 00_B by the application and updated by the core subsequently.</p>
EoL [26]	End of List	<p>For Non Isochronous, it indicates that this is the last descriptor in the list, if set. The core does not generate a BNA interrupt for the next descriptor, if it is unavailable.</p> <p>For Isochronous, this field is reserved. This field is controlled by the application.</p>
IOC[25]	Interrupt On complete	<p>Set by the application, this bit indicates that the core must generate a transfer complete interrupt after this descriptor is finished.</p>

Table 16-5 OUT Data Memory Structure Values (cont'd)

Bit	Bit ID	Description	
[24] ¹⁾	SuP	Non Isochronous OUT Setup Packet When set, it indicates that the buffer data pointed by this descriptor is a setup packet of 8 bytes	Isochronous Reserved: [24:12]
[23] ¹⁾	R	Non Isochronous OUT Setup Packet Bit: Reserved [23] Bit ID: Reserved	
[22:17] ¹⁾	Varies	Non Isochronous OUT Bit Reserved: [22:17] Bit ID: Reserved	
[16:12] ¹⁾	Varies	Non Isochronous OUT OUT Bit: [16:0]	Isochronous Bit [11:0]
[11:0] ¹⁾	Varies	Bit ID: Total bytes to transfer. This 17-bit value can take values from 0 to 128K-1 bytes, indicating the number of bytes of data to be transmitted to the USB device. Note: A Value of 0 indicates zero bytes of data, 1 indicates 1 byte of data and so on	Bit ID: Total bytes to transfer. This 12-bit value can take values from 0 to 4K bytes, indicating the number of bytes of data to be transmitted to the USB device. Note: A Value of 0 indicates zero bytes of data, 1 indicates 1 byte of data, and so on.

1) The meaning of this field varies. See description.

Table 16-6 displays the out buffer pointer field description.

Table 16-6 IN Buffer Pointer

Buf Addr[31:0]	Buffer Address	The Buffer pointer field in the descriptor is 32 bits wide and contains the address where the transmit data is to be stored in the system memory. The buffer address does not need to be aligned with DWORD.
----------------	----------------	--

16.7.3 Channel Initialization in Scatter-Gather DMA Mode

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps.

- Program the periodic frame list array (for periodic channels).

Universal Serial Bus (USB)

- Program the HFLBAddr register with the base address of the periodic frame list array (for periodic channels).
- Program the HCFG register with PerSchedEn bit set.
- Program at least one transfer descriptor in the system memory.
- Program the HCDMAx with the pointer to the corresponding descriptor.
- Program the GINTMSK register to unmask the Channel Interrupts.
- Program the HAINTMSK register to unmask the selected channels' interrupts.
- Program the HCINTMSK register to unmask the ChHalt, XferCompl, and BNA.
- Program the HCTSIZx register with initial data PID and SCHED_INFO (for periodic channels).
- Program the HCCHARx register with the device's endpoint characteristics, such as type, speed, direction, and so on (The channel can be enabled by setting the channel enable bit to 1_B only when the application is ready to transmit or receive any packet).

16.7.4 Asynchronous Transfers

When the application enables an asynchronous (Bulk and Control) channel by writing into the HCCHARx register, the host controller begins servicing the asynchronous channel. It reads the referenced (CTD) transfer descriptor qTDn (pointed to by the HCDMAx register). If the read qTDn is active, the host controller caches the qTDn and then schedules a transaction. If the read qTDn is inactive, the host controller disables the channel and generates a Buffer Not Available (BNA) interrupt.

If multiple asynchronous channels are enabled simultaneously, the host controller caches the referenced transfer descriptor of the entire enabled channels. The host controller schedules transactions for each enabled channel in round-robin fashion.

When the host controller completes the transfer for a channel, it updates the status quadlet of the processed qTDn in the system memory.

For a normal completion, the host controller updates the status of the qTDn with no errors. The host controller completes a transfer normally if one of the following events occurs:

- Short or zero length packet is received for an IN channel.
- The allocated buffer is fulfilled with the received data packets for an IN channel.
- The allocated buffer is fully transferred to the device for an OUT channel.

When a transfer is completed normally, the host controller attempts to process the next qTDn from the descriptor list, if the End of List (EOL) bit is not set in the completed qTDn. where $m = \text{AQTD}$ (if IN channel with $\text{AQTD_VLD}=1$ received a short packet) or $m = (n + 1) \bmod (\text{NTD} + 1)$

If EOL is set, the host controller disables the channel and generates a Channel Halt interrupt. The transfer complete interrupt is generated for the following conditions.

- IOC is set.
- Short or zero length packet is received for an IN channel.

- EOL is set.

For an abnormal completion, the host controller updates the status of the qTDn with PKT_ERR. The host controller completes a transfer abnormally if one of the following events occurs:

- STALL response is received from the device.
- Excessive transaction errors occurred.
- Babble detected.

When a transfer is completed abnormally, the host controller disables the channel and then generates a Channel Halt interrupt with the appropriate status in HCINT register.

16.7.4.1 Asynchronous Transfer Descriptor

The application must use separate qTD for different stages of control transfers. A three stage control transfer uses three qTDs. The same qTD can be reused for performing different stages of control transfer. The combination of EPType, EPDir fields of the HCCHARx register, and the SuP flag of the qTD decides the stage of the control transfer. See [Table 16-7](#).

Table 16-7 Asynchronous Transfer Descriptor

HCCHARx.EP Type	HCCHARx.EP Dir	qTD.SuP	Control Stage
00 _B	0	1	SETUP
00 _B	0	0	Data stage OUT / Status stage OUT
00 _B	1	0	Data stage IN / Status stage IN
00 _B	1	1	Invalid

The host controller executes a zero-length OUT transaction if the "Num bytes to transmit" field of the qTD is initialized to zero for an OUT channel. For an IN channel, the "Num of bytes received" field of the qTD must be always initialized to an integer multiple of the maximum packet size.

The application can use one or multiple qTDs for bulk IN and OUT transfers. The number of qTDs depends on the available consecutive data buffer space and the size of the transfer. Each qTD can support up to 64KB of consecutive data buffer space.

16.7.5 Periodic Transfers

The periodic schedule is used to manage all isochronous and interrupt transfer streams. The base of the periodic schedule is the periodic frame list. The periodic schedule is referenced from the register space using the HFLBAddrBase and the HFNUM registers. The periodic schedule is based on an array of scheduled channels called the periodic

Universal Serial Bus (USB)

frame list. The periodic frame list implements a sliding window of transactions over time. When the application enables the periodic schedule (PerSchedEna) in the HCFG register, the host controller attempts to read an entry from the frame list that corresponds to the next running frame number at the beginning of each frame.

The periodic frame list can be programmed to 8, 16, 32, or 64 elements. The size of the periodic frame list should be large enough to support the required b-interval of the least frequent channel. The least significant bits [15:0] in the periodic frame list elements are used to identify the scheduled periodic channels (0 through 15) for that corresponding frame. For example if channel 2 and 6 are periodic channels scheduled for a frame then the corresponding entry in the periodic frame list will be 0000_0044_H.

The host controller should program the SCHED_INFO to 1111_1111_B when operating in Full Speed for all the enabled periodic channels.

16.7.5.1 Isochronous Transactions

When the application enables an isochronous channel by writing into the HCCHARx register, the host controller begins servicing the isochronous channel based on the programmed scheduling (periodic frame list and SCHED_INFO). The application must use separate qTD for each frame. Each qTD handles a frame of transactions. The application is expected to allocate a qTD with Active bit zero even if no transaction is scheduled for a frame. The position of the active qTD determines the b-interval of the isochronous channel.

The host controller supports high-bandwidth isochronous transfer via the multi-count (MC) field of the HCCHARx register. The Multi Count represents a transaction count per frame for the endpoint. If the multi-count is zero, the operation of the host controller is undefined. Multi-count greater than one is not applicable for the FS host.

For OUT transfers, the value of the "Num bytes to transmit" field represents the total bytes to be sent during the frame. The application is expected to program the Mult count field to be the maximum number of packets to be sent in any frame. The host controller automatically selects the number of packets and its data PID based on the programmed Xfer Size.

For IN transfers, the host controller issues Mult count transactions. The application is expected to initialize the "Num bytes received" field to (MC * MaxPktSize).

The host controller does not execute all Multi-count transactions if:

- The channel is an OUT and the "Num bytes to transmit" goes to zero before all the Multi-count transactions have executed (ran out of transmit data) or
- The channel is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before all the Multi-count transaction have been executed.
- The channel is an IN and the endpoint delivers a packet with DATA0 PID before all the Multi-count transaction have been executed.

Universal Serial Bus (USB)

Each transfer descriptor (qTD) describes one frame of transactions. The host controller will cache one transfer descriptor in a frame prior to the scheduled frame.

When the application is adding new isochronous transactions to the schedule, it always performs a read of the HFNUM register to determine the current frame and frame the host is currently executing. Because there is no information about where in the frame the host controller is, a constant uncertainty factor of one frame for FS is assumed.

The end of frame (FS) may occur before all of the transaction opportunities are executed. When this happens, the host controller closes the corresponding descriptor and proceeds to processing the next scheduled descriptor. If the scheduled descriptor is not fetched by the host controller due to high system latency, the host controller does not execute any transaction for that scheduled frame and will skip the descriptor without any update (that is, without clearing the Active bit).

When a transfer is completed normally, the host controller generates the transfer complete interrupt only if IOC is set in the completed qTD.

When a transfer is completed abnormally (STALL response or Babble), the host controller disables the channel and then generates a Channel Halt interrupt with the appropriate status in HCINT register. The host controller updates the status of the qTD with PKT_ERR if one of the following conditions occurs:

- STALL response is received from the device
- Error packet received
- Babble detected
- Unable to complete all the transactions in the scheduled frame

An example for the FS isochronous scheduling is shown in [Figure 16-21](#). In this figure, channels 2 and 15 are isochronous channels with b-interval 1ms and 4ms respectively. The host controller fetches only the qTDs that corresponds to the scheduled frame (Periodic Frame List entry). The host controller initiates the qTD fetch in the frame prior to the scheduled frame. If the qTD is active and belongs to an OUT channel, the host controller also fetches the corresponding data in the previous frame. If this qTD is not active, the host controller ignores the qTD and does not generate any BNA interrupt.

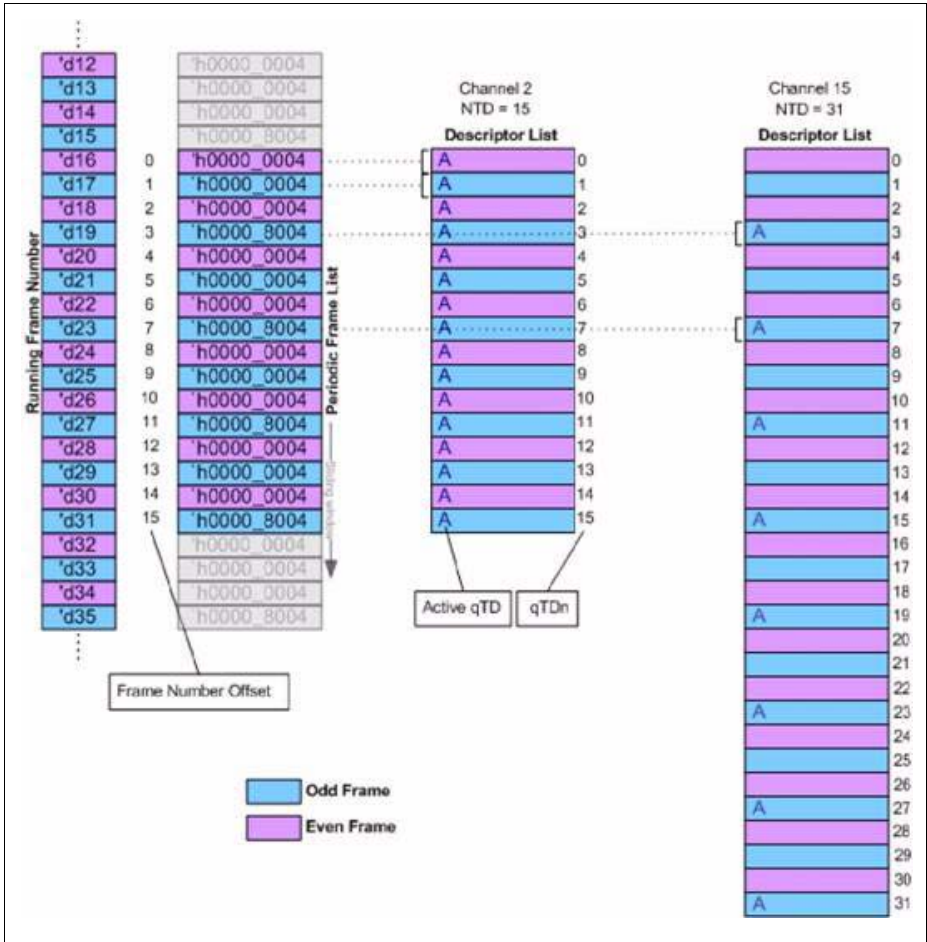


Figure 16-21 Full Speed Isochronous Transfer Scheduling

16.7.5.2 Interrupt Transactions

When the application enables an interrupt channel by writing into the HCCHARx register, the host controller begins servicing the interrupt channel based on the programmed scheduling (periodic frame list and SCHLD_INFO). It reads the referenced (CTD) transfer descriptor qTDn (pointed by the HCDMAX register) in the frame prior to the scheduled frame.

If the read qTDn is active, the host controller caches the qTDn and then schedules a transaction. If the read qTDn is inactive, the host controller disables the channel and generates a Buffer Not Available (BNA) interrupt.

When the host controller completes the transfer, it updates the status quadlet of the processed qTDn in the system memory.

For a normal completion, the host controller updates the status of the qTDn with no errors. The host controller completes a transfer normally if one of the following events occurs:

- Short or zero length packet is received for an IN channel.
- The allocated buffer is fulfilled with the received data packets for an IN channel.
- The allocated buffer is fully transferred to the device for an OUT channel.

When a transfer is completed normally, the host controller attempts to process the next qTDm from the descriptor list if the End of List (EOL) bit is not set in the completed qTDn.

Where $m = (n + 1) \bmod (NTD + 1)$

If EOL is set, the host controller disables the channel and generates Channel Halt interrupt. The transfer complete interrupt will be generated for the following conditions.

- IOC is set.
- Short or zero length packet is received for an IN channel.
- EOL is set.

For an abnormal completion, the host controller updates the status of the qTDn with PKT_ERR. The host controller completes a transfer abnormally if one of the following events occurs:

- STALL response is received from the device.
- Excessive transaction errors occurred.
- Babble detected.

When a transfer is completed abnormally, the host controller disables the channel and then generates a Channel Halt interrupt with the appropriate status in the HCINT register.

The host controller supports high-bandwidth interrupt transfer through the Multi-count (MC) field of HCCHARx register. The Multi-count represents a transaction count per frame for the endpoint. If the Multi-count is zero, the operation of the host controller is undefined. Multi-count greater than one is not applicable for FS host.

The host controller does not execute all Multi-count transactions in a frame if:

Universal Serial Bus (USB)

- The channel is an OUT and the "Num bytes to transmit" goes to zero before all the Multi-count transactions have executed (ran out of transmit data) or
- The channel is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before all the Multi-count transaction have been executed.
- The channel is an IN and the "Num bytes received" goes to zero before all the Multi-count transaction are executed (ran out of receive buffer space).

16.8 Device Programming Overview

This section discusses how to program the DWC_otg core when it is in Device mode.

16.8.1 Device Initialization

As prerequisites, the application must meet the following conditions to set up the device core to handle traffic:

- In Slave mode, GINTMSK.NPTxFEmpMsk, and GINTMSK.RxFLvlMsk must be unset.
- In DMA mode, the GINTMSK.NPTxFEmpMsk, and GINTMSK.RxFLvlMsk interrupts must be masked.

The application must perform the following steps to initialize the core at device on, power on, or after a mode change from Host to Device.

1. Program the following fields in DCFG register.
 - a) DescDMA bit
 - b) Device Speed
 - c) NonZero Length Status OUT Handshake
 - d) Periodic Frame Interval (If Periodic Endpoints are supported)
2. Clear the DCTL.SftDiscon bit. The core issues a connect after this bit is cleared.
3. Program the GINTMSK register to unmask the following interrupts.
 - a) USB Reset
 - b) Enumeration Done
 - c) Early Suspend
 - d) USB Suspend
 - e) SOF
4. Wait for the GINTSTS.USBReset interrupt, which indicates a reset has been detected on the USB and lasts for about 10 ms. On receiving this interrupt, the application must perform the steps listed in [“Initialization on USB Reset” on Page 16-74](#).
5. Wait for the GINTSTS.EnumerationDone interrupt. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the DSTS register to determine the enumeration speed and perform the steps listed in [“Initialization on Enumeration Completion” on Page 16-75](#).

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

16.8.2 Device Connection

The device connect process varies depending if the VBUS is on or off when the device is connected to the USB cable.

VBUS is on when the device is connected

If VBUS is on when the device is connected to the USB cable, there is no SRP from the device. The device connection flow is as follows:

1. The device triggers the GINTSTS.SessReqInt [bit 30] interrupt bit.
2. When the device application detects the GINTSTS.SessReqInt interrupt, it programs the required bits in the DCFG register.
3. When the host drives reset, the device triggers GINTSTS.USBSt [bit 12] on detecting the reset. The host then follows the USB 2.0 Enumeration sequence.

VBUS is off when the device is connected

If VBUS is off when the device is connected to the USB cable, the device initiates SRP in OTG Revision 1.3 mode. The device connection flow is as follows:

1. The application initiates SRP by writing the Session Request bit in the OTG Control and Status register. The USB core performs data-line pulsing followed by VBUS pulsing.
2. The host starts a new session by turning on VBUS, indicating SRP success. The USB core interrupts the application by setting the Session Request Success Status Change bit in the OTG Interrupt Status register.
3. The application reads the Session Request Success bit in the OTG Control and Status register and programs the required bits in DCFG register.
4. When host drives reset, the device triggers GINTSTS.USBSt on detecting the reset. The host then follows the USB 2.0 Enumeration sequence.

16.8.3 Device Disconnection

The device session ends when the USB cable is disconnected or if the VBUS is switched off by the host.

The device disconnect flow is as follows:

1. When the USB cable is unplugged or when the VBUS is switched off by the host, the device core triggers GINTSTS.OTGInt [bit 2] interrupt bit.
2. When the device application detects GINTSTS.OTGInt interrupt, it checks that the GOTGINT.SesEndDet (Session End Detected) bit is set to 1.

16.8.3.1 Device Soft Disconnection

The application can also perform a soft disconnect by setting the DCTL.SftDiscon bit.

Send/Receive USB Transfers -> Soft disconnect->Soft reset->USB Device Enumeration

Sequence of operations:

1. The application configures the device to send or receive transfers.

Universal Serial Bus (USB)

2. The application sets the Soft disconnect bit (SftDiscon) in the Device Control Register (DCTL).
3. The application sets the Soft Reset bit (CSftRst) in the Reset Register (GRSTCTL).
4. Poll the GRSTCTL register until the core clears the soft reset bit, which ensures the soft reset is completed properly.
5. Initialize the core according to the instructions in **“Device Initialization” on Page 16-72.**

Suspend-> Soft disconnect->Soft reset->USB Device Enumeration

Sequence of operations:

1. The core detects a USB suspend and generates a Suspend Detected interrupt.
2. The application sets the Stop PHY Clock bit (StopPclk) in the Power and Clock Gating Control register (PCGCCTL), the core asserts suspend_n to the PHY, and the PHY clock stops.
3. The application clears the StopPclk bit and waits for the PHY clock to come back. The core de-asserts suspend_n to the PHY, and the PHY clock comes back.
4. The application sets the Soft disconnect bit (SftDiscon) in Device Control Register (DCTL).
5. The application sets the Soft Reset bit (CSftRst) in the Reset Register (GRSTCTL).
6. Poll the GRSTCTL register until the core clears the soft reset bit, which ensures the soft reset is completed properly.
7. Initialize the core according to the instructions in **“Device Initialization” on Page 16-72.**

16.8.4 Endpoint Initialization

16.8.4.1 Initialization on USB Reset

1. Set the NAK bit for all OUT endpoints
 - a) DOEPCTLx.SNAK = 1 (for all OUT endpoints)
2. Unmask the following interrupt bits:
 - a) DAINTMSK.INEP0 = 1 (control 0 IN endpoint)
 - b) DAINTMSK.OUTEPO = 1 (control 0 OUT endpoint)
 - c) DOEPMSK.SETUP = 1
 - d) DOEPMSK.XferCompl = 1
 - e) DIEPMSK.XferCompl = 1
 - f) DIEPMSK.TimeOut = 1
3. To transmit or receive data, the device must initialize more registers as specified in **“Device Initialization” on Page 16-72**

Universal Serial Bus (USB)

4. Set up the Data FIFO RAM for each of the FIFOs
 - a) Program the GRXFSIZ Register, to be able to receive control OUT data and setup data. At a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 DWORDs (for the status of the control OUT data packet) + 10 DWORDs (for setup packets).
 - b) Program the dedicated FIFO size register (depending on the FIFO number chosen) in Dedicated FIFO operation, to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
5. Reset the Device Address field in Device Configuration Register (DCFG).
6. (This step is not required if the Scatter/Gather DMA mode is used.) Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
 - a) DOEPTSIZ0.Setup Count = 3 (to receive up to 3 back-to-back SETUP packets)
 - b) In DMA mode, DOEPDMAO register with a memory address to store any SETUP packets received

At this point, all initialization required to receive SETUP packets is done, except for enabling control OUT endpoint 0 in DMA mode.

16.8.4.2 Initialization on Enumeration Completion

This section describes what the application must do when it detects an Enumeration Done interrupt.

1. On the Enumeration Done interrupt (GINTSTS.EnumDone, read the DSTS register to determine the enumeration speed.
2. Program the DIEPCTL0.MPS field to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.
3. In DMA mode, program the DOEPCTL0 register to enable control OUT endpoint 0, to receive a SETUP packet. In Scatter/Gather DMA mode, the descriptors must be set up in memory before enabling the endpoint.
 - a) DOEPCTL0.EPEna = 1
4. Unmask the SOF interrupt.

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

16.8.4.3 Initialization on SetAddress Command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet.

16.8.4.4 Initialization on SetConfiguration/SetInterface Command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. For details on a particular endpoint's activation or deactivation, see **“Endpoint Activation” on Page 16-76** and **“Endpoint Deactivation” on Page 16-76**.
5. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the DAIN_TMSK register.
6. Set up the Data FIFO RAM for each FIFO. See **“Data FIFO RAM Allocation” on Page 16-223** for more detail.
7. After all required endpoints are configured, the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

16.8.4.5 Endpoint Activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the DIEPCTLx register (for IN or bidirectional endpoints) or the DOEPCTLx register (for OUT or bidirectional endpoints).
 - a) Maximum Packet Size
 - b) USB Active Endpoint = 1
 - c) Set Endpoint Data Toggle bit to 0 (for interrupt and bulk endpoints)
 - d) Endpoint Type
 - e) TxFIFO Number
2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

16.8.4.6 Endpoint Deactivation

This section describes the steps required to deactivate an existing endpoint.

Universal Serial Bus (USB)

Before an endpoint can be de-activated, any pending transfers must first be stopped. For more information on stopping transfers, see **“Transfer Stop Programming for OUT Endpoints” on Page 16-79** or **“Transfer Stop Programming for IN Endpoints” on Page 16-82**.

1. In the endpoint to be deactivated, clear the USB Active Endpoint bit in the DIEPCTLx register (for IN or bidirectional endpoints) or the DOEPCTLx register (for OUT or bidirectional endpoints).
2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, resulting in a timeout on the USB.

16.8.5 Programming OUT Endpoint Features

16.8.5.1 Disabling an OUT Endpoint

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application Programming Sequence

1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core, as described in **“Setting the Global OUT NAK” on Page 16-78**.
 - a) $DCTL.DCTL.SGOUTNak = 1_B$
2. Wait for the GINTSTS.GOUTNakEff interrupt
3. Disable the required OUT endpoint by programming the following fields.
 - a) $DOEPCTLx.EPDisable = 1_B$
 - b) $DOEPCTLx.SNAK = 1_B$
4. Wait for the DOEPINTx.EPDisabled interrupt, which indicates that the OUT endpoint is completely disabled. When the EPDisabled interrupt is asserted, the core also clears the following bits.
 - a) $DOEPCTLx.EPDisable = 0_B$
 - b) $DOEPCTLx.EPEnable = 0_B$
5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
 - a) $DCTL.SGOUTNak = 0_B$

16.8.5.2 Stalling a Non-Isochronous OUT Endpoint

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode, as described in **“Setting the Global OUT NAK” on Page 16-78**.
2. Disable the required endpoint, as described in **“Disabling an OUT Endpoint” on Page 16-77**.

Universal Serial Bus (USB)

- a) When disabling the endpoint, instead of setting the DOEPCTL.SNAK bit, set DOEPCTL.STALL = 1.
 - b) The Stall bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the DOEPCTLx.STALL bit must be cleared.

If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the Stall bit must be set or cleared before the application sets up the Status stage transfer on the control endpoint.

16.8.5.3 Setting the Global OUT NAK

Internal Data Flow

1. When the application sets the Global OUT NAK (DCTL.SGOUTNak), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets
2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern. See [“Data FIFO RAM Allocation” on Page 16-223](#).
3. When either the core (in DMA mode) or the application (in Slave mode) pops the Global OUT NAK pattern DWORD from the receive FIFO, the core sets the GINTSTS.GOUTNakEff interrupt.
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the DCTL.SGOUTNak bit.

Application Programming Sequence

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field.
 - a) DCTL.SGOUTNak = 1_B
2. Wait for the assertion of the interrupt GINTSTS.GOUTNakEff. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set DCTL.SGOUTNak and before the core asserts the GINTSTS.GOUTNakEff interrupt.
4. The application can temporarily mask this interrupt by writing to the GINTMSK.GINNAkEffMsk bit.
 - a) GINTMSK.GINNAkEffMsk = 0_B

Universal Serial Bus (USB)

5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the DCTL.SGOUTNak bit. This also clears the GINTSTS.GOUTNakEff interrupt.
 - a) DCTL.CGOUTNak = 1_B
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
 - a) GINTMSK.GINNakEffMsk = 1_B

16.8.5.4 Transfer Stop Programming for OUT Endpoints

When the core is operating as a device, the following programming sequence can be used to stop any transfers (because of an interrupt from the host, typically a reset).

Note: The Rx FIFO is common for OUT endpoints, therefore there is only one transfer stop programming flow for OUT endpoints.

Sequence of operations:

1. Enable all OUT endpoints by setting DOEPCTL.EPEna = 1_B.
2. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core, according to the instructions in **“Setting the Global OUT NAK” on Page 16-78**. This ensures that data in the RX FIFO is sent to the application successfully. Set DCTL.DCTL.SGOUTNak = 1_B.
3. Wait for the GINTSTS.GOUTNakEff interrupt.
4. Disable all active OUT endpoints by programming the following register bits:
 - a) DOEPCTL.EPEna = 1_B
 - b) DOEPCTLn.EPDisable = 1_B
 - c) DOEPCTLn.SNAK = 1_B
5. Wait for the DOEPINTn.EPDisabled interrupt for each OUT endpoint programmed in the previous step. The DOEPINTn.EPDisabled interrupt indicates that the corresponding OUT endpoint is completely disabled. When the EPDisabled interrupt is asserted, the DWC_otg core clears the following bits:
 - a) DOEPCTL.EPEna = 0_B
 - b) DOEPCTLn.EPDisable = 0_B
 - c) DOEPCTLn.EPEnable = 0_B

Note: The application must not flush the Rx FIFO, as the Global out NAK effective interrupt earlier ensures that there is no data left in the Rx FIFO.

16.8.6 Programming IN Endpoint Features

16.8.6.1 Setting IN Endpoint NAK

Internal Data Flow

1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint's transmit FIFO.
2. Non-isochronous IN tokens receive a NAK handshake reply
 - a) Isochronous IN tokens receive a zero-data-length packet reply
3. The core asserts the DIEPINTx.IN NAK Effective interrupt in response to the DIEPCTL.Set NAK bit.
4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the DIEPCTLx.Clear NAK bit.

Application Programming Sequence

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
 - a) DIEPCTLx.SetNAK = 1_B
2. Wait for assertion of the DIEPINTx.NAK Effective interrupt. This interrupt indicates the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the DIEPMSK.NAK Effective bit.
 - a) DIEPMSK.NAK Effective = 0_B
5. To exit Endpoint NAK mode, the application must clear the DIEPCTLx.NAK status. This also clears the DIEPINTx.NAK Effective interrupt.
 - a) DIEPCTLx.ClearNAK = 1_B
6. If the application masked this interrupt earlier, it must be unmasked as follows:
 - a) DIEPMSK.NAK Effective = 1_B

16.8.6.2 IN Endpoint Disable

Use the following sequence to disable a specific IN endpoint (periodic/non-periodic) that has been previously enabled in dedicated FIFO operation.

Application Programming Sequence:

1. In Slave mode, the application must stop writing data on the AHB, for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode. See **“Setting IN Endpoint NAK” on Page 16-79**.
 - a) DIEPCTLx.SetNAK = 1_B
3. Wait for DIEPINTx.NAK Effective interrupt.
4. Set the following bits in the DIEPCTLx register for the endpoint that must be disabled.

- a) DIEPCTLx.Endpoint Disable = 1
- b) DIEPCTLx.SetNAK = 1
5. Assertion of DIEPINTx.Endpoint Disabled interrupt indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits.
 - a) DIEPCTLx.EPEnable = 0_B
 - b) DIEPCTLx.EPDisable = 0_B
6. The application must read the DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint was transmitted on the USB.
7. The application must flush the data in the Endpoint transmit FIFO, by setting the following fields in the GRSTCTL register.
 - a) GRSTCTL.TxFIFONum = Endpoint Transmit FIFO Number
 - b) GRSTCTL.TxFFlush = 1

The application must poll the GRSTCTL register, until the TxFFlush bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

16.8.6.3 Timeout for Control IN Endpoints

The application must treat the TIMEOUT interrupt received for the last IN transaction of a Control Transfers Data Stage separately. This is done to take into account the TIMEOUT due to lost ACK case (core did not receive the ACK send by the host). Application must unmask timeout interrupt for control IN transfers Data phase only. On getting the timeout interrupt for control endpoint data phase, the application must also enable the OUT control endpoint for the status phase.

If the timeout is due to a lost ACK, the host switches to the Data stage, and the application receives Transfer Complete interrupt for the OUT endpoint. The application can then flush the IN packet and disable both the IN and OUT endpoints. If the timeout was due to lost data, the host sends the IN token again, and the application receives a Transfer Complete interrupt for the IN endpoint. The application can thus keep the OUT endpoint enabled for the status phase.

16.8.6.4 Stalling Non-Isochronous IN Endpoints

This section describes how the application can stall a non-isochronous endpoint.

Application Programming Sequence

1. Disable the IN endpoint to be stalled. See **“IN Endpoint Disable” on Page 16-80** for more details. Set the Stall bit as well.
2. DIEPCTLx.Endpoint Disable = 1, when the endpoint is already enabled
 - a) DIEPCTLx.STALL = 1
 - b) The Stall bit always takes precedence over the NAK bit

Universal Serial Bus (USB)

3. Assertion of the DIEPINTx.Endpoint Disabled interrupt indicates to the application that the core has disabled the specified endpoint.
4. The application must flush the Non-periodic or Periodic Transmit FIFO, depending on the endpoint type. In case of a non-periodic endpoint, the application must re-enable the other non-periodic endpoints, which do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the DIEPCTLx.STALL bit must be cleared.
6. If the application sets or clears a STALL for an endpoint due to a SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the Stall bit must be set or cleared before the application sets up the Status stage transfer on the control endpoint.

Special Case: Stalling the Control IN/OUT Endpoint

The core must stall IN/OUT tokens if, during the Data stage of a control transfer, the host sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the application must enable DIEPINTx.INTknTXFemp and DOEPINTx.OUTTknEPdis interrupts during the Data stage of the control transfer, after the core has transferred the amount of data specified in the SETUP packet. Then, when the application receives this interrupt, it must set the STALL bit in the corresponding endpoint control register, and clear this interrupt.

16.8.6.5 Transfer Stop Programming for IN Endpoints

When the core is operating as a device, the following programming sequence can be used to stop any transfers (because of an interrupt from the host, typically a reset).

Sequence of operations:

1. Disable the IN endpoint by programming DIEPCTLn.EPDis = 1_B.
2. Wait for the DIEPINTn.EPDisabled interrupt, which indicates that the IN endpoint is completely disabled. When the EPDisabled interrupt is asserted, the core clears the following bits:
 - a) DIEPCTL.EPDisable = 0_B
 - b) DIEPCTL.EPEnable = 0_B
3. Flush the TX FIFO by programming the following bits:
 - a) GRSTCTL.TxFFIsh = 1_B
 - b) GRSTCTL.TxFNum = <FIFO number specific to endpoint>
4. The application can start polling till GRSTCTL.TXFFIsh is cleared. When this bit is cleared, it ensures that there is no data left in the TX FIFO.

16.8.6.6 Non-Periodic IN Endpoint Sequencing

In DMA mode, the DIEPCTLx.NextEp value programmed controls the order in which the core fetches non- periodic data for IN endpoints.

Universal Serial Bus (USB)

If application requires the core to fetch data for the non-periodic IN endpoints in a certain endpoint order, it must program the DIEPCTLx.NextEP field accordingly before enabling the endpoints. To enable a single endpoint enabled at a time the application must set the DIEPCTLx.NextEP field to the endpoint number itself. The core uses the NextEP field irrespective of the DIEPCTLx.EPEna bit.

16.8.7 Worst-Case Response Time

When the USB core acts as a device, there is a worst case response time for any tokens that follow an isochronous OUT. This worst case response time depends on the AHB clock frequency.

The core registers are in the AHB domain, and the core does not accept another token before updating these register values. The worst case is for any token following an isochronous OUT, because for an isochronous transaction, there is no handshake and the next token could come sooner. This worst case value is 7 PHY clocks when the AHB clock is the same as the PHY clock. When AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/ interrupt tokens with a NAK and drops isochronous and SETUP tokens. The host interprets this as a timeout condition for SETUP and retries the SETUP packet. For isochronous transfers, the incomplISOCIN and incomplISOCOUT interrupts inform the application that isochronous IN/OUT packets were dropped.

16.8.8 Choosing the Value of GUSBCFG.USBTrdTim

The value in GUSBCFG.USBTrdTim is the time it takes for the Media Access Controller (MAC), in terms of PHY clocks after it has received an IN token, to get the FIFO status, and thus the first data. The MAC is the part of the USB core that handles USB transactions and protocols. This time involves the synchronization delay between the PHY and AHB clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock. In this case, the delay is 5 clocks. If the PHY clock is running at 60 MHz and the AHB is running at 30 MHz, this value is 9 clocks.

If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for GUSBCFG.USBTrdTim.

The application can use the following formula to calculate the value of GUSBCFG.USBTrdTim:

$$4 * \text{AHB Clock} + 1 \text{ PHY Clock} \\ = (2 \text{ clock sync} + 1 \text{ clock memory address} + 1 \text{ clock memory data from sync RAM}) + (1 \text{ PHY Clock (next PHY clock MAC can sample the 2-clock FIFO output)})$$

16.8.9 Handling Babble Conditions

If USB core receives a packet that is larger than the maximum packet size for that endpoint, the core stops writing data to the Rx buffer and waits for the end of packet (EOP). When the core detects the EOP, it flushes the packet in the Rx buffer and does not send any response to the host.

If the core continues to receive data at the EOF2 (the end of frame 2, which is very close to SOF), the core generates an `early_suspend` interrupt (`GINTSTS.ErlySusp`). On receiving this interrupt, the application must check the `erratic_error` status bit (`DSTS.ErrticErr`). If this bit is set, the application must take it as a long babble and perform a soft reset.

16.8.10 Device Programming Operations in Buffer DMA or Slave Mode

Table 16-8 provides links to the programming sequence for different USB transaction types when the core is in Slave or Buffer DMA mode of operation.

For information on device programming operations when in Scatter/Gather DMA mode, see **Section 16.11**.

Table 16-8 Device Programming Operations

Device Mode	IN	SETUP	OUT
Control			
Slave	“Non-Periodic (Bulk and Control) IN Data Transfers” on Page 16-98	“OUT Data Transfers” on Page 16-94	“Non-Isochronous OUT Data Transfers” on Page 16-104
DMA	“Non-Periodic (Bulk and Control) IN Data Transfers” on Page 16-132	“OUT Data Transfers” on Page 16-129	“Non-Isochronous OUT Data Transfers” on Page 16-134
Bulk			
Slave	“Non-Periodic (Bulk and Control) IN Data Transfers” on Page 16-98	-	“Non-Isochronous OUT Data Transfers” on Page 16-104
DMA	“Non-Periodic (Bulk and Control) IN Data Transfers” on Page 16-132	-	“Non-Isochronous OUT Data Transfers” on Page 16-134

Table 16-8 Device Programming Operations (cont'd)

Device Mode	IN	SETUP	OUT
Interrupt			
Slave	<p>“Periodic IN (Interrupt and Isochronous) Data Transfers” on Page 16-117</p> <p>“Periodic IN Data Transfers Using the Periodic Transfer Interrupt” on Page 16-119</p>	-	<p>“Non-Isochronous OUT Data Transfers” on Page 16-104</p> <p>“Interrupt OUT Data Transfers Using Periodic Transfer Interrupt” on Page 16-125</p>
DMA	<p>“Periodic IN (Interrupt and Isochronous) Data Transfers” on Page 16-138</p> <p>“Periodic IN Data Transfers Using the Periodic Transfer Interrupt” on Page 16-140</p>	-	<p>“Non-Isochronous OUT Data Transfers” on Page 16-134</p> <p>“Interrupt OUT Data Transfers Using Periodic Transfer Interrupt” on Page 16-146</p>

Table 16-8 Device Programming Operations (cont'd)

Device Mode	IN	SETUP	OUT
Isochronous			
Slave	<p>“Periodic IN (Interrupt and Isochronous) Data Transfers” on Page 16-117</p> <p>“Periodic IN Data Transfers Using the Periodic Transfer Interrupt” on Page 16-119</p>	-	<p>“Control Read Transfers (SETUP, Data IN, Status OUT)” on Page 16-88</p> <p>“Incomplete Isochronous OUT Data Transfers” on Page 16-114</p>
DMA	<p>“Periodic IN (Interrupt and Isochronous) Data Transfers” on Page 16-138</p> <p>“Periodic IN Data Transfers Using the Periodic Transfer Interrupt” on Page 16-140</p>	-	<p>“Control Read Transfers (SETUP, Data IN, Status OUT)” on Page 16-128</p> <p>“Incomplete Isochronous OUT Data Transfers” on Page 16-136</p>

16.9 Device Programming in Slave Mode

This section discusses how to program the core when it is acting as a Device in the Slave mode of operation.

16.9.1 Control Transfers

This section describes the various types of control transfers.

16.9.1.1 Control Write Transfers (SETUP, Data OUT, Status IN)

This section describes control write transfers.

Application Programming Sequence

1. Assertion of the DOEPINTx.SETUP Packet interrupt indicates that a valid SETUP packet has been transferred to the application. See **“OUT Data Transfers” on**

- Page 16-94** for more details. At the end of the Setup stage, the application must reprogram the DOEPTSIZE.SUPCnt field to 3 to receive the next SETUP packet.
2. If the last SETUP packet received before the assertion of the SETUP interrupt indicates a data OUT phase, program the core to perform a control OUT transfer as explained in **“Non-Isochronous OUT Data Transfers” on Page 16-104**.
 3. In a single OUT data transfer on control endpoint 0, the application can receive up to 64 bytes. If the application is expecting more than 64 bytes in the Data OUT stage, the application must re-enable the endpoint to receive another 64 bytes, and must continue to do so until it has received all the data in the Data stage.
 4. Assertion of the DOEPINTx.Transfer Compl interrupt on the last data OUT transfer indicates the completion of the data OUT phase of the control transfer.
 5. On completion of the data OUT phase, the application must do the following.
 - a) To transfer a new SETUP packet in DMA mode, the application must re-enable the control OUT endpoint as explained in section **“OUT Data Transfers” on Page 16-94**.
 - $DOEPCTLx.EPEna = 1_B$
 - b) To execute the received Setup command, the application must program the required registers in the core. This step is optional, based on the type of Setup command received.
 6. For the status IN phase, the application must program the core as described in **“Non-Periodic (Bulk and Control) IN Data Transfers” on Page 16-98** to perform a data IN transfer.
 7. Assertion of the DIEPINTx.Transfer Compl interrupt indicates completion of the status IN phase of the control transfer.

16.9.1.2 Control Read Transfers (SETUP, Data IN, Status OUT)

This section describes control write transfers.

Application Programming Sequence

1. Assertion of the DOEPINTx.SETUP Packet interrupt indicates that a valid SETUP packet has been transferred to the application. See **“OUT Data Transfers” on Page 16-94** for more details. At the end of the Setup stage, the application must reprogram the DOEPTSIZE.SUPCnt field to 3 to receive the next SETUP packet.
2. If the last SETUP packet received before the assertion of the SETUP interrupt indicates a data IN phase, program the core to perform a control IN transfer as explained in **“Non-Periodic (Bulk and Control) IN Data Transfers” on Page 16-98**.
3. On a single IN data transfer on control endpoint 0, the application can transmit up to 64 bytes. To transmit more than 64 bytes in the Data IN stage, the application must re-enable the endpoint to transmit another 64 bytes, and must continue to do so, until it has transmitted all the data in the Data stage.

Universal Serial Bus (USB)

4. The DIEPINTx.Transfer Compl interrupt on the last IN data transfer marks the completion of the control transfer's Data stage.
5. To perform a data OUT transfer in the status OUT phase, the application must program the core as described in **“OUT Data Transfers” on Page 16-94**. The application must program the DCFG.NZStsOUTHShk handshake field to a proper setting before transmitting an data OUT transfer for the Status stage.
6. Assertion of the DOEPINTx.Transfer Compl interrupt indicates completion of the status OUT phase of the control transfer. This marks the successful completion of the control read transfer.

16.9.1.3 Two-Stage Control Transfers (SETUP/Status IN)

This section describes two-stage control transfers.

Application Programming Sequence

1. Assertion of the DOEPINTx.SetUp interrupt indicates that a valid SETUP packet has been transferred to the application. See **“OUT Data Transfers” on Page 16-94** for more detail. To receive the next SETUP packet, the application must reprogram the DOEPTSIZx.SUPCnt field to 3 at the end of the Setup stage.
2. Decode the last SETUP packet received before the assertion of the SETUP interrupt. If the packet indicates a two-stage control command, the application must do the following.
 - a) Set DOEPCTLx.EPEna = 1_B
 - b) Depending on the type of Setup command received, the application can be required to program registers in the core to execute the received Setup command.
3. For the status IN phase, the application must program the core described in **“Non-Periodic (Bulk and Control) IN Data Transfers” on Page 16-98** to perform a data IN transfer.
4. Assertion of the DIEPINTx.Transfer Compl interrupt indicates the completion of the status IN phase of the control transfer.

Example: Two-Stage Control Transfer

These notes refer to **Figure 16-22**.

1. SETUP packet #1 is received on the USB and is written to the receive FIFO, and the core responds with an ACK handshake. This handshake is lost and the host detects a timeout.
2. The SETUP packet in the receive FIFO results in a GINTSTS.RxFLVl interrupt to the application, causing the application to empty the receive FIFO.
3. SETUP packet #2 on the USB is written to the receive FIFO, and the core responds with an ACK handshake.
4. The SETUP packet in the receive FIFO sends the application the GINTSTS.RxFLVl interrupt and the application empties the receive FIFO.

Universal Serial Bus (USB)

5. After the second SETUP packet, the host sends a control IN token for the status phase. The core issues a NAK response to this token, and writes a Setup Stage Done entry to the receive FIFO. This entry results in a GINTSTS.RxFLvl interrupt to the application, which empties the receive FIFO. After reading out the Setup Stage Done DWORD, the core asserts the DOEPINTx.Setup packet interrupt to the application.
6. On this interrupt, the application processes SETUP Packet #2, decodes it to be a two-stage control command, and clears the control IN NAK bit.
 - a) DIEPCTLx.CNAK = 1
7. When the application clears the IN NAK bit, the core interrupts the application with a DIEPINTx.INTknTXFemp. On this interrupt, the application enables the control IN endpoint with a DIEPTSIZx.XferSize of 0 and a DIEPTSIZx.PktCnt of 1. This results in a zero-length data packet for the status IN token on the USB.
8. At the end of the status IN phase, the core interrupts the application with a DIEPINTx.XferCompl interrupt.

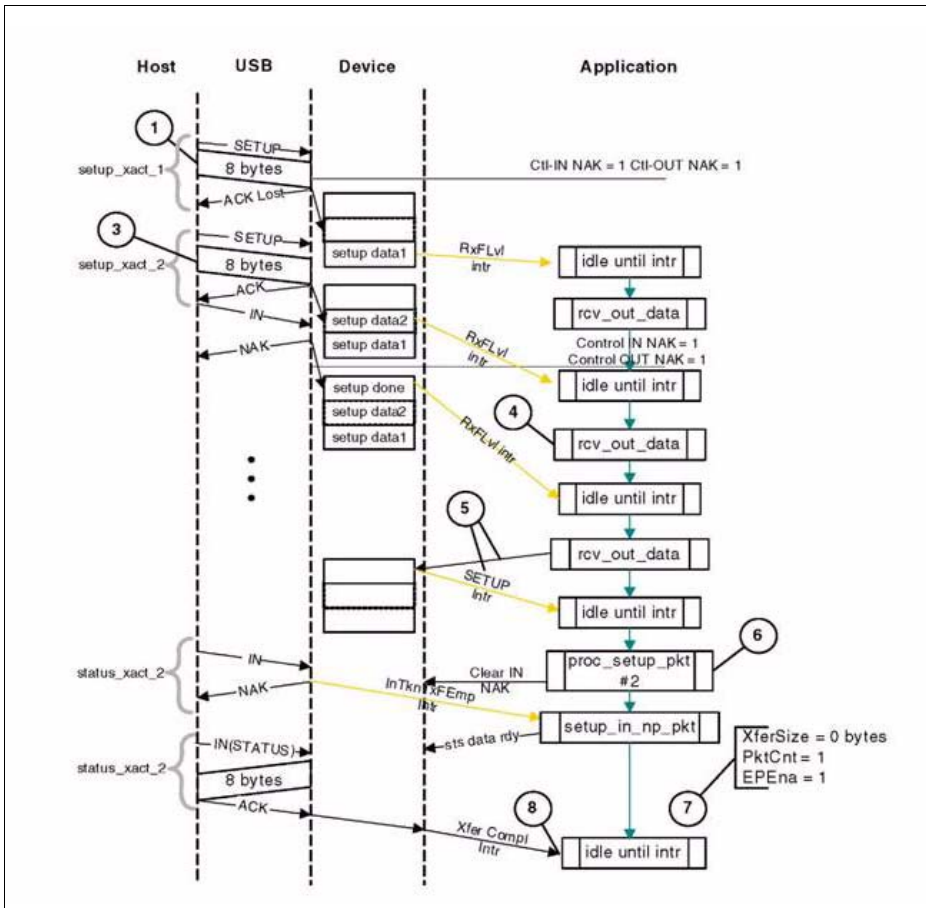


Figure 16-22 Two-Stage Control Transfer

16.9.1.4 Packet Read from FIFO

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO in Slave mode.

1. On catching a GINTSTS.RxFLvl interrupt, the application must read the Receive Status Pop register (GRXSTSP).
2. The application can mask the GINTSTS.RxFLvl interrupt by writing to GINTMSK.RxFLvl = 0_B, until it has read the packet from the receive FIFO.

Universal Serial Bus (USB)

3. If the received packet's byte count is not 0, the byte count amount of data is popped from the receive Data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the Receive Data FIFO.
4. The receive FIFO's packet status readout indicates one of the following.
5. Global OUT NAK Pattern: PktSts = Global OUT NAK, BCnt = 11'h000, EPNum = Dont Care (4'h0), DPID = Dont Care (00_B). This data indicates that the global OUT NAK bit has taken effect.
 - a) SETUP Packet Pattern: PktSts = SETUP, BCnt = 11'h008, EPNum = Control EP Num, DPID = D0. This data indicates that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.
 - b) Setup Stage Done Pattern: PktSts = Setup Stage Done, BCnt = 11'h0, EPNum = Control EP Num, DPID = Don't Care (00_B). This data indicates that the Setup stage for the specified endpoint has completed and the Data stage has started. After this entry is popped from the receive FIFO, the core asserts a Setup interrupt on the specified control OUT endpoint.
 - c) Data OUT Packet Pattern: PktSts = DataOUT, BCnt = size of the Received data OUT packet (0 < BCnt <1,024), EPNum = EPNum on which the packet was received, DPID = Actual Data PID.
 - d) Data Transfer Completed Pattern: PktSts = Data OUT Transfer Done, BCnt = 11'h0, EPNum = OUT EP Num on which the data transfer is complete, DPID = Dont Care (00_B). This data indicates that a OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a Transfer Completed interrupt on the specified OUT endpoint.
The encoding for the PktSts is listed in **“Receive Status Debug Read/Status Read and Pop Registers (GRXSTSR/GRXSTSP)” on Page 16-266.**
6. After the data payload is popped from the receive FIFO, the GINTSTS.RxFLVL interrupt must be unmasked.
7. Steps 1-5 are repeated every time the application detects assertion of the interrupt line due to GINTSTS.RxFLVL. Reading an empty receive FIFO can result in undefined core behavior.

Figure 16-23 provides a flow chart of this procedure.

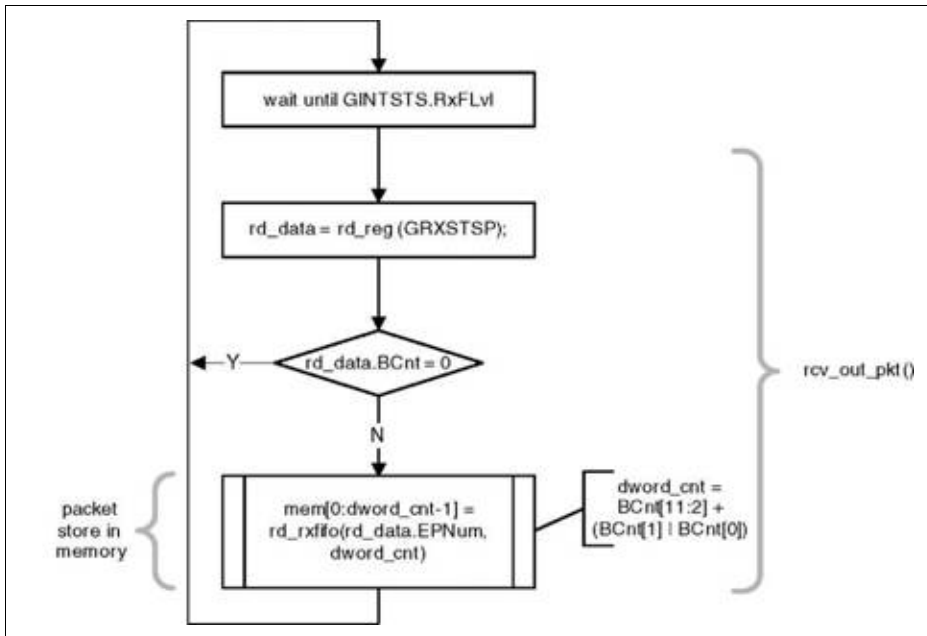


Figure 16-23 Receive FIFO Packet Read in Slave Mode

16.9.2 IN Data Transfers

This section describes the internal data flow and application-level operations during IN data transfers.

16.9.2.1 Packet Write

This section describes how the application writes data packets to the endpoint FIFO in Slave mode with dedicated transmit FIFOs.

1. The application can either choose polling or interrupt mode.
 - a) In polling mode, application monitors the status of the endpoint transmit data FIFO, by reading the DTXFSTSx register, to determine, if there is enough space in the data FIFO.
 - b) In interrupt mode, application waits for the DIEPINTx.TxFEmp interrupt and then reads the DTXFSTSx register, to determine, if there is enough space in the data FIFO.
 - c) To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.

- d) For writing zero length packet, application must not look for FIFO space.
- Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. The application, typically must do a read modify write on the DIEPCTLx, to avoid modifying the contents of the register, except for setting the Endpoint Enable bit.

The application can write multiple packets for the same endpoint, into the transmit FIFO, if space is available. For periodic IN endpoints, application must write packets only for one frame. It can write packets for the next periodic transaction, only after getting transfer complete for the previous transaction.

16.9.3 OUT Data Transfers

This section describes the internal data flow and application-level operations during data OUT transfers and setup transactions.

16.9.3.1 Control Setup Transactions

This section describes how the core handles SETUP packets and the application's sequence for handling setup transactions. To initialize the core after power-on reset, the application must follow the sequence in [“Core Initialization” on Page 16-11](#). Before it can communicate with the host, it must initialize an endpoint as described in [“Endpoint Initialization” on Page 16-74](#). See [“Packet Read from FIFO” on Page 16-91](#).

Application Requirements

- To receive a SETUP packet, the DOEPTSIz.SUPCnt field in a control OUT endpoint must be programmed to a non-zero value. When the application programs the SUPCnt field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the DOEPCTLx.NAK status and DOEPCTLx.EPEna bit setting. The SUPCnt field is decremented every time the control endpoint receives a SETUP packet. If the SUPCnt field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the SUPCnt field, but the application possibly is not be able to determine the correct number of SETUP packets received in the Setup stage of a control transfer.
 - DOEPTSIz.SUPCnt = 3
- The application must always allocate some extra space in the Receive Data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
 - The space to be Reserved is 10 DWORDs. Three DWORDs are required for the first SETUP packet, 1 DWORD is required for the Setup Stage Done DWORD, and 6 DWORDs are required to store two extra SETUP packets among all control endpoints.

Universal Serial Bus (USB)

- b) 3 DWORDs per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (Setup Packet Pattern). The core reserves this space in the receive data
- c) FIFO to write SETUP data only, and never uses this space for data packets.
3. In Slave mode, the application must read the 2 DWORDs of the SETUP packet from the receive FIFO.
4. The application must read and discard the Setup Stage Done DWORD from the receive FIFO.

Internal Data Flow

1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and Stall bit settings.
 - a) The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB, 3 DWORDs of data is written to the receive FIFO, and the SUPCnt field is decremented by 1.
 - a) The first DWORD contains control information used internally by the core
 - b) The second DWORD contains the first 4 bytes of the SETUP command
 - c) The third DWORD contains the last 4 bytes of the SETUP command
3. When the Setup stage changes to a Data IN/OUT stage, the core writes an entry (Setup Stage Done DWORD) to the receive FIFO, indicating the completion of the Setup stage.
4. On the AHB side, the application empties the SETUP packets.
5. When the application pops the Setup Stage Done DWORD from the receive FIFO, the core interrupts the application with a DOEPINTx.SETUP interrupt, indicating it can process the received SETUP packet.
6. The core clears the endpoint enable bit for control OUT endpoints.

Application Programming Sequence

1. Program the DOEPTSIz register.
 - a) DOEPTSIz.SUPCnt = 3
2. Wait for the GINTSTS.RxFLvl interrupt and empty the data packets from the receive FIFO, as explained in **“Packet Read from FIFO” on Page 16-91**. This step can be repeated many times.
3. Assertion of the DOEPINTx.SETUP interrupt marks a successful completion of the SETUP Data Transfer. On this interrupt, the application must read the DOEPTSIz register to determine the number of SETUP packets received and process the last received SETUP packet.

Note: If the application has not enabled EPO before the host sends the SETUP packet, the core ACKs the SETUP packet and stores it in the FIFO, but does not write to the memory until EPO is enabled. When the application enables the EPO (first

Universal Serial Bus (USB)

enable) and clears the NAK bit at the same time the Host sends DATA OUT, the DATA OUT is stored in the RxFIFO. The USB core then writes the setup data to the memory and disables the endpoint. Though the application expects a Transfer Complete interrupt for the Data OUT phase, this does not occur, because the SETUP packet, rather than the DATA OUT packet, enables EP0 the first time. Thus, the DATA OUT packet is still in the RxFIFO until the application re-enables EP0. The application must enable EP0 one more time for the core to process the DATA OUT packet.

Figure 16-24 charts this flow.

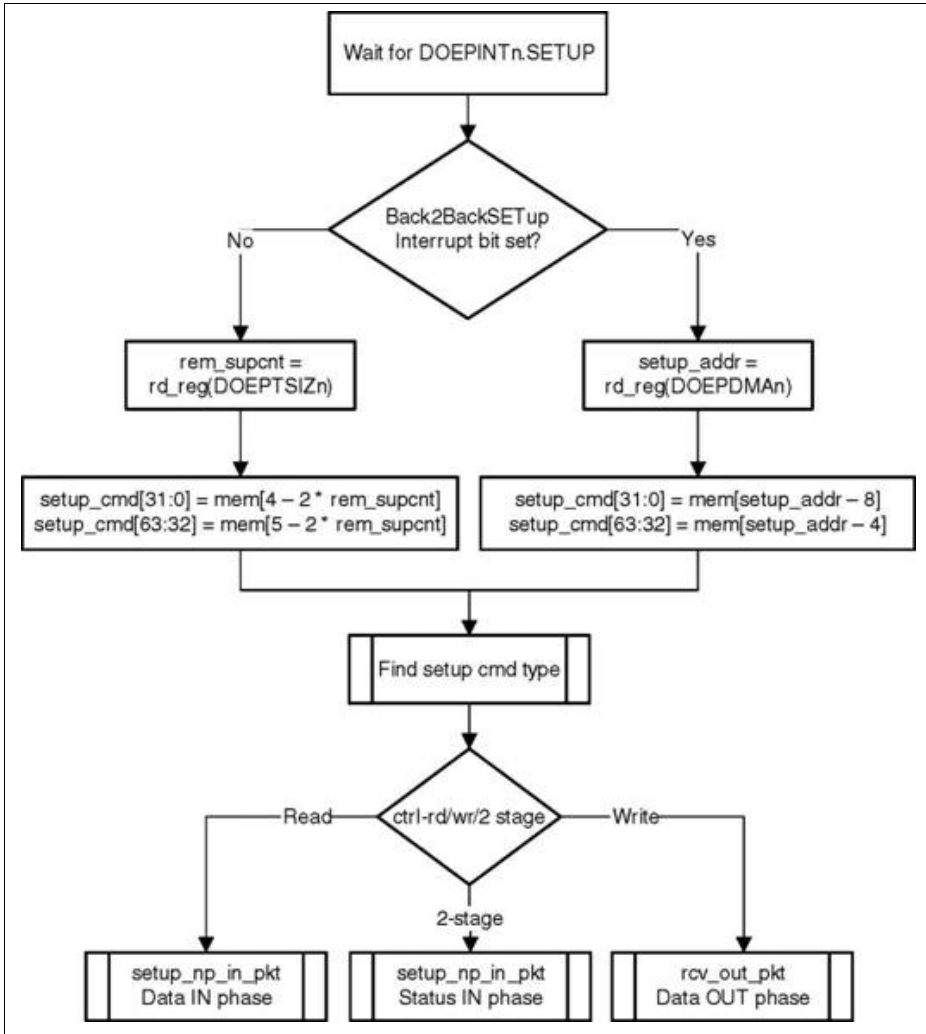


Figure 16-24 Processing a SETUP Packet

16.9.3.2 Handling More Than Three Back-to-Back SETUP Packets

According to the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP

packets a host can send to the same endpoint. When this condition occurs, the USB core generates an interrupt (DOEPINTx.Back2BackSETup).

16.9.4 Non-Periodic (Bulk and Control) IN Data Transfers

This section describes a regular non-periodic IN data transfer.

Application Requirements

- For IN transfers, the Transfer Size field in the Endpoint Transfer Size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
 - To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
 - Transfer size[epnum] = $n * mps[epnum] + sp$
(where n is an integer > 0 , and $0 < sp < mps[epnum]$)
 - If ($sp > 0$), then packet count[epnum] = $n + 1$.
 - Otherwise, packet count[epnum] = n
 - To transmit a single zero-length data packet:
 - Transfer size[epnum] = 0
 - Packet count[epnum] = 1
 - To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer in two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.
 - First transfer: transfer size[epnum] = $n * mps[epnum]$; packet count = n ;
 - Second transfer: transfer size[epnum] = 0; packet count = 1;
- Once an endpoint is enabled for data transfers, the core updates the Transfer Size register. At the end of IN transfer, which ended with an Endpoint Disabled interrupt, the application must read the Transfer Size register to determine how much data posted in the transmit FIFO was already sent on the USB.
- Data fetched into transmit FIFO = Application-programmed initial transfer size - core-updated final transfer size
 - Data transmitted on USB = (application-programmed initial packet count - Core updated final packet count) * $mps[epnum]$
 - Data yet to be transmitted on USB = (Application-programmed initial transfer size - data transmitted on USB)

Internal Data Flow

- The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
- In Slave mode, the application must also write the required data to the transmit FIFO for the endpoint.

Universal Serial Bus (USB)

3. Every time the application writes a packet into the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the "number of packets in FIFO" count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data is written to the transmit FIFO, the core reads it out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a TIMEOUT.
5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the Packet Count field.
6. If there is no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates a IN Tkn Rcvd When FIFO Empty Interrupt for the endpoint, provided the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. In Dedicated FIFO operation, the core internally rewinds the FIFO pointers and no timeout interrupt is generated except for Control IN endpoint.
8. When the transfer size is 0 and the packet count is 0, the transfer complete interrupt for the endpoint is generated and the endpoint enable is cleared.

Application Programming Sequence

1. Program the DIEPTSIZx register with the transfer size and corresponding packet count.
2. Program the DIEPCTLx register with the endpoint characteristics and set the CNAK and Endpoint Enable bits.
3. When transmitting non-zero length data packet, the application must poll the DTXFSTSx register (where n is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use DIEPINTx.TxFEmp before writing the data.

16.9.4.1 Examples**Slave Mode Bulk IN Transaction**

These notes refer to **Figure 16-25**.

1. The host attempts to read data (IN token) from an endpoint.
2. On receiving the IN token on the USB, the core returns a NAK handshake, because no data is available in the transmit FIFO.

Universal Serial Bus (USB)

3. To indicate to the application that there was no data to send, the core generates a DIEPINTx.IN Token Rcvd When Tx FIFO Empty interrupt.
4. When data is ready, the application sets up the DIEPTSIZx register with the Transfer Size and Packet Count fields.
5. The application writes one maximum packet size or less of data to the Non-periodic Tx FIFO.
6. The host reattempts the IN token.
7. Because data is now ready in the FIFO, the core now responds with the data and the host ACKs it.
8. Because the XferSize is now zero, the intended transfer is complete. The device core generates a DIEPINTx.XferCompl interrupt.
9. The application processes the interrupt and uses the setting of the DIEPINTx.XferCompl interrupt bit to determine that the intended transfer is complete.

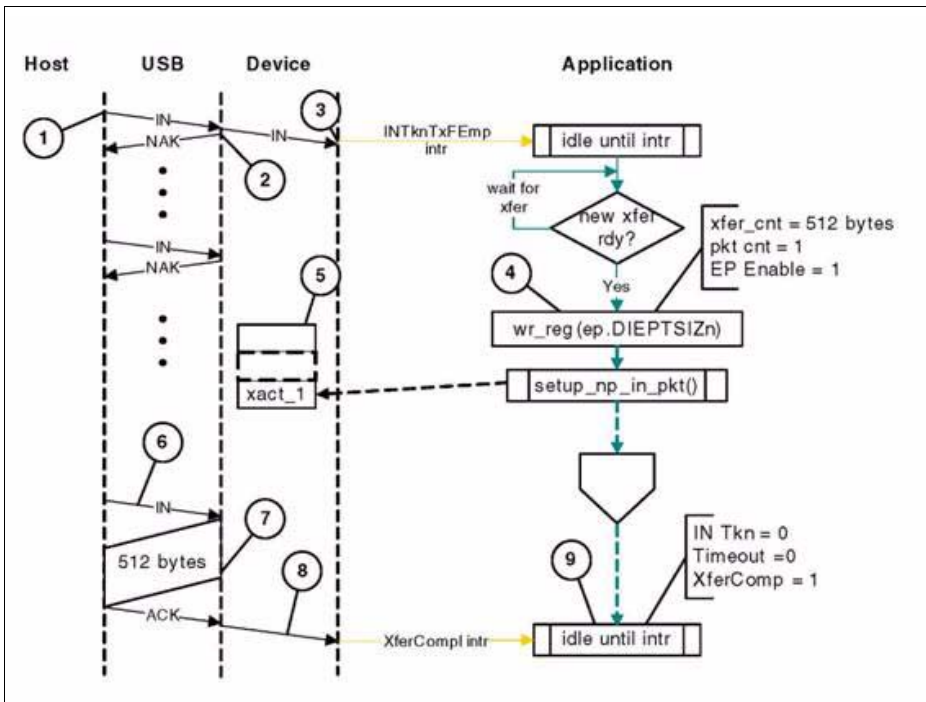


Figure 16-25 Slave Mode Bulk IN Transaction

Slave Mode Bulk IN Transfer (Pipelined Transaction)

These notes refer to [Figure 16-26](#).

Universal Serial Bus (USB)

1. The host attempts to read data (IN token) from an endpoint.
2. On receiving the IN token on the USB, the core returns a NAK handshake, because no data is available in the transmit FIFO.
3. To indicate that there was no data to send, the core generates an DIEPINTx.InTkn Rcvd When TxFIFO Empty interrupt.
4. When data is ready, the application sets up the DIEPTSIZx register with the transfer size and packet count.
5. The application writes one maximum packet size or less of data to the Non-periodic TxFIFO.
6. The host reattempts the IN token.
7. Because data is now ready in the FIFO, the core responds with the data, and the host ACKs it.
8. When the TxFIFO level falls below the halfway mark, the core generates a GINTSTS.NonPeriodic TxFIFO Empty interrupt. This triggers the application to start writing additional data packets to the FIFO.
9. A data packet for the second transaction is ready in the TxFIFO.
10. A data packet for third transaction is ready in the TxFIFO while the data for the second packet is being sent on the bus.
11. The second data packet is sent to the host.
12. The last short packet is sent to the host.
13. Because the last packet is sent and XferSize is now zero, the intended transfer is complete. The core generates a DIEPINTx.XferCompl interrupt.
14. The application processes the interrupt and uses the setting of the DIEPINTx.XferCompl interrupt bit to determine that the intended transfer is complete

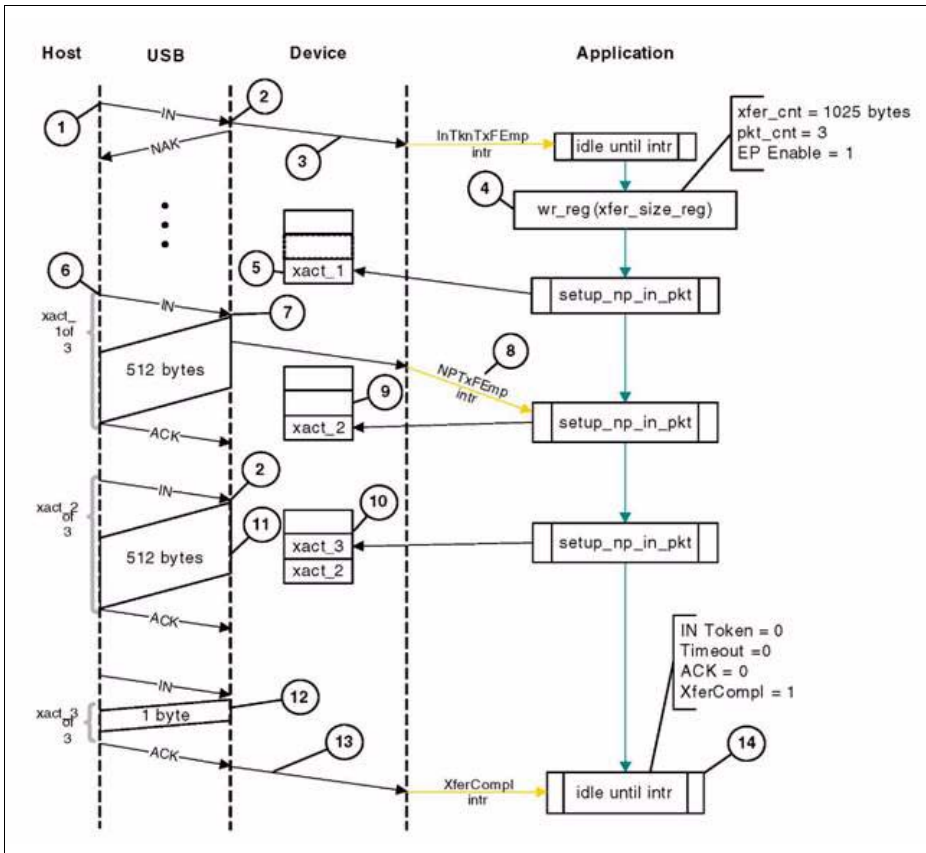


Figure 16-26 Slave Mode Bulk IN Transfer (Pipelined Transaction)

Slave Mode Bulk IN Two-Endpoint Transfer

These notes refer to [Figure 16-27](#).

1. The host attempts to read data (IN token) from endpoint 1.
2. On receiving the IN token on the USB, the core returns a NAK handshake, because no data is available in the transmit FIFO for endpoint 1, and generates a DIEPINT1.InTkn Rcvd When TxFIFO Empty interrupt.
3. The application processes the interrupt and initializes DIEPTSIZ1 register with the Transfer Size and Packet Count fields. The application starts writing the transaction data to the transmit FIFO.

Universal Serial Bus (USB)

4. The application writes one maximum packet size or less of data for endpoint 1 to the Non-periodic TxFIFO.
5. Meanwhile, the host attempts to read data (IN token) from endpoint 2.
6. On receiving the IN token on the USB, the core returns a NAK handshake, because no data is available in the transmit FIFO for endpoint 2, and the core generates a DIEPINT2.InTkn Rcvd When TxFIFO Empty interrupt.
7. Because the application has completed writing the packet for endpoint 1, it initializes the DIEPTSIZ2 register with the Transfer Size and Packet Count fields. The application starts writing the transaction data into the transmit FIFO for endpoint 2.
8. The host repeats its attempt to read data (IN token) from endpoint 1.
9. Because data is now ready in the TxFIFO, the core returns the data, which the host ACKs.
10. Meanwhile, the application has initialized the data for the next two packets in the TxFIFO (ep2.xact1 and ep1.xact2, in order).
11. The host repeats its attempt to read data (IN token) from endpoint 2.
12. Because endpoint 2's data is ready, the core responds with the data (ep2.xact_1), which the host ACKs.
13. Meanwhile, the application has initialized the data for the next two packets in the TxFIFO (ep2.xact2 and ep1.xact3, in order). The application has finished initializing data for the two endpoints involved in this scenario.
14. The host repeats its attempt to read data (IN token) from endpoint 1.
15. Because data is now ready in the FIFO, the core responds with the data, which the host ACKs.
16. The host repeats its attempt to read data (IN token) from endpoint 2.
17. With data now ready in the FIFO, the core responds with the data, which the host ACKs.
18. With the last packet for endpoint 2 sent and its XferSize now zero, the intended transfer is complete. The core generates a DIEPINT2.XferCompl interrupt for this endpoint.
19. The application processes the interrupt and uses the setting of the DIEPINT2.XferCompl interrupt bit to determine that the intended transfer on endpoint 2 is complete.
20. The host repeats its attempt to read data (IN token) from endpoint 1 (last transaction).
21. With data now ready in the FIFO, the core responds with the data, which the host ACKs.
22. Because the last endpoint one packet has been sent and XferSize is now zero, the intended transfer is complete. The core generates a DIEPINT1.XferCompl interrupt for this endpoint.
23. The application processes the interrupt and uses the setting of the DIEPINT1.XferCompl interrupt bit to determine that the intended transfer on endpoint 1 is complete.

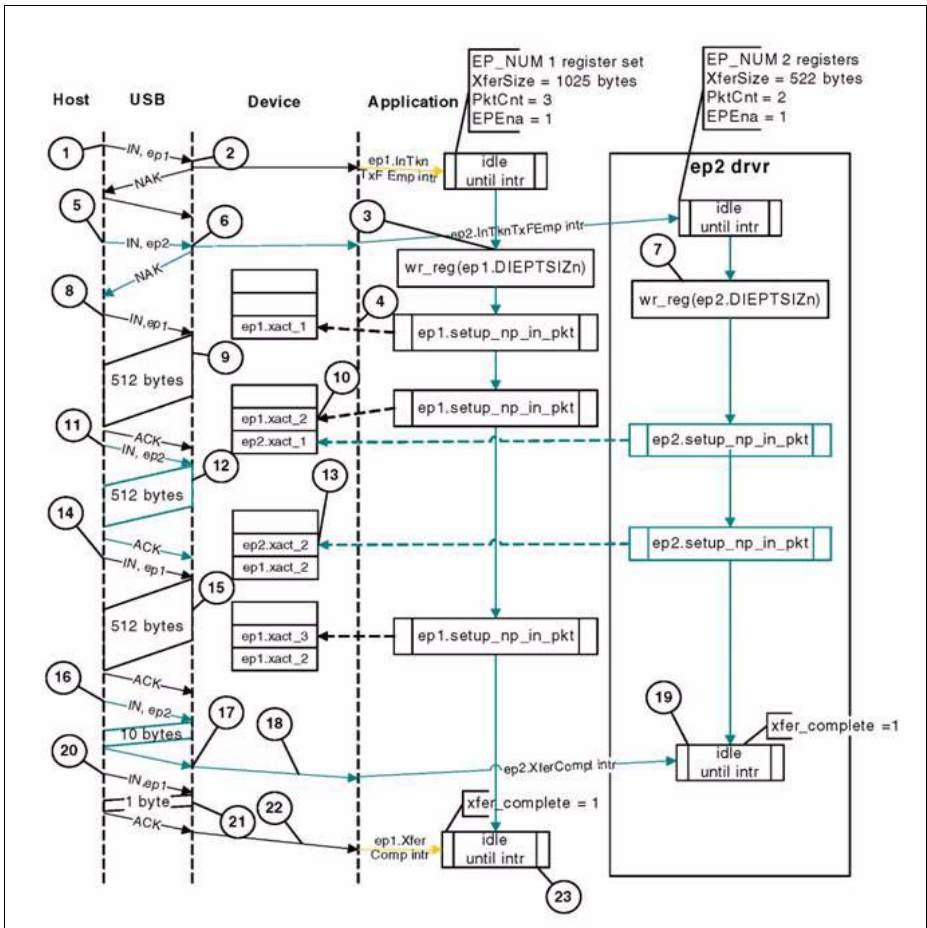


Figure 16-27 Slave Mode Bulk IN Two-Endpoint Transfer

16.9.5 Non-Isochronous OUT Data Transfers

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application Requirements

1. For OUT transfers, the Transfer Size field in the endpoint's Transfer Size register must be a multiple of the maximum packet size of the endpoint, adjusted to the

DWORD boundary.

```
if (mps[epnum] mod 4) == 0
transfer size[epnum] = n * (mps[epnum] //DWORD aligned
else
transfer size[epnum] = n * (mps[epnum] + 4 - (mps[epnum] mod 4))
//Non-DWORD aligned
packet count[epnum] = n
n > 0
```

2. On any OUT endpoint interrupt, the application must read the endpoint's Transfer Size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
 - a) Payload size in memory = application-programmed initial transfer size - core updated final transfer size
 - b) Number of USB packets in which this payload was received = application-programmed initial packet count - core updated final packet count

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the Packet Count field for that endpoint by 1.
 - a) OUT data packets received with Bad Data CRC are flushed from the receive FIFO automatically.
 - b) After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
 - c) If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
 - d) In all the above three cases, the packet count is not decremented because no data is written to the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
4. After the data is written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.

Universal Serial Bus (USB)

5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
6. The OUT Data Transfer Completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions.
 - a) The transfer size is 0 and the packet count is 0
 - b) The last OUT data packet written to the receive FIFO is a short packet (0 < packet size < maximum packet size)
7. When the application pops this entry (OUT Data Transfer Completed), a Transfer Completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application Programming Sequence

1. Program the DOEPTSIZE register for the transfer size and the corresponding packet count.
2. Program the DOEPCTL register with the endpoint characteristics, and set the Endpoint Enable and ClearNAK bits.
 - a) DOEPCTL.EPENA = 1
 - b) DOEPCTL.CNAK = 1
3. In Slave mode, wait for the GINTSTS.Rx StsQ level interrupt and empty the data packets from the receive FIFO as explained in **“Packet Read from FIFO” on Page 16-91**.
 - a) This step can be repeated many times, depending on the transfer size.
4. Asserting the DOEPINT.XferCompl interrupt marks a successful completion of the non- isochronous OUT data transfer.
5. Read the DOEPTSIZE register to determine the size of the received data payload.

Note: The XferSize is not decremented for the last packet.

Bulk OUT Transactions in Slave Mode

Figure 16-28 depicts the reception of a single bulk OUT data packet from the USB to the AHB and describes the events involved in the process.

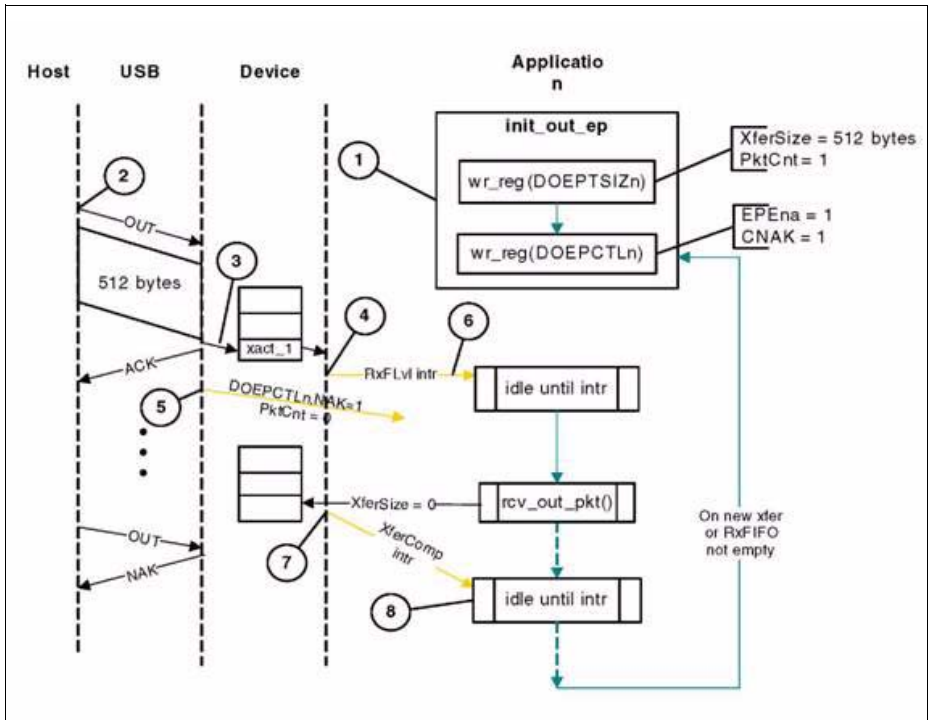


Figure 16-28 Slave Mode Bulk OUT Transaction

After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting DOEPCTLx.CNAK = 1 and DOEPCTLx.EPEna = 1, and setting a suitable XferSize and PktCnt in the DOEPSIZx register.

1. Host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the RxFIFO because space is available there.
3. After writing the complete packet in the RxFIFO, the core then asserts the GINTSTS.RxFLvl interrupt.
4. On receiving the PktCnt number of USB packets, the core sets the NAK bit for this endpoint internally to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the RxFIFO.

Universal Serial Bus (USB)

6. When the application has read all the data (equivalent to XferSize), the core generates a DOEPINTx.XferCompl interrupt.
7. The application processes the interrupt and uses the setting of the DOEPINTx.XferCompl interrupt bit to determine that the intended transfer is complete.

16.9.6 Isochronous OUT Data Transfers

This section describes a regular isochronous OUT data transfer.

Application Requirements

1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers
2. For isochronous OUT data transfers, the Transfer Size and Packet Count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
 - a) $1 \leq \text{packet count}[\text{epnum}] \leq 3$
3. When isochronous OUT endpoints are supported in the device, the application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (GINTSTS.EOPF interrupt).
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the GINTSTS.EOPF and before the GINTSTS.SOF.

Internal Data Flow

1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on a isochronous OUT endpoint in a particular frame only if the following condition is met.
 - a) $\text{DOEPCTLx.Even/Odd frame} = \text{DSTS.SOFFN}[0]$
3. When the application completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the DOEPTSIZx.Received DPID field with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application Programming Sequence

1. Program the DOEPTSIZx register for the transfer size and the corresponding packet count.
2. Program the DOEPCTLx register with the endpoint characteristics and set the Endpoint Enable, ClearNAK, and Even/Odd frame bits.

- a) Endpoint Enable = 1
- b) CNAK=1
- c) Even/Odd frame = (0: Even/1: Odd)
3. In Slave mode, wait for the GINTSTS.Rx StsQ level interrupt and empty the data packets from the receive FIFO as explained in **“Packet Read from FIFO” on Page 16-91**.
 - a) This step can be repeated many times, depending on the transfer size.
4. The assertion of the DOEPINTx.XferCompl interrupt marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory is good.
5. This interrupt can not always be detected for isochronous OUT transfers. Instead, the application can detect the GINTSTS.incomplete Isochronous OUT data interrupt. See **“Incomplete Isochronous OUT Data Transfers” on Page 16-114**, for more details
6. Read the DOEPTSIZx register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met.
 - a) DOEPTSIZx.RxDPID = D0 and the number of USB packets in which this payload was received = 1
 - b) DOEPTSIZx.RxDPID = D1 and the number of USB packets in which this payload was received = 2
 - c) DOEPTSIZx.RxDPID = D2 and the number of USB packets in which this payload was received = 3
7. The number of USB packets in which this payload was received = App Programmed Initial Packet Count - Core Updated Final Packet Count
The application can discard invalid data packets.

16.9.7 Isochronous OUT Data Transfers Using Periodic Transfer Interrupt

This section describes a regular isochronous OUT data transfer with the Periodic Transfer Interrupt feature.

Application Requirements

1. Before setting up ISOC OUT transfers spanned across multiple frames, the application must allocate buffer in the memory to accommodate all data to be received as part of the OUT transfers, then program that buffer's size and start address in the endpoint-specific registers.
 - a) The application must mask the GINTSTS.incomp ISO OUT.
 - b) The application must enable the DCTL.IgnrFrmNum
2. For ISOC transfers, the Transfer Size field in the DOEPTSIZx.XferSize register must be a multiple of the maximum packet size of the endpoint, adjusted to the DWORD boundary. The Transfer Size programmed can span across multiple frames based on

Universal Serial Bus (USB)

the periodicity after which the application wants to receive the DOEPINTx.XferCompl interrupt

- a) $\text{transfer size}[\text{epnum}] = n * (\text{mps}[\text{epnum}] + 4 - (\text{mps}[\text{epnum}] \bmod 4))$
 - b) $\text{packet count}[\text{epnum}] = n$
 - c) $n > 0$ (Higher value of n reduces the periodicity of the DOEPINTx.XferCompl interrupt)
 - d) $1 \leq \text{packet count}[\text{epnum}] \leq n$ (Higher value of n reduces the periodicity of the DOEPINTx.XferCompl interrupt).
3. In DMA mode, the core stores a received data packet in the memory, always starting on a DWORD boundary. If the maximum packet size of the endpoint is not a multiple of 4, the core inserts byte pads at end of a maximum-packet-size packet up to the end of the DWORD. The application will not be informed about the frame number and the PID value on which a specific OUT packet has been received.
 4. The assertion of the DOEPINTx.XferCompl interrupt marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory is good.
 - a) On DOEPINTx.XferCompl, the application must read the endpoint's Transfer Size register to calculate the size of the payload in the memory.
 - b) Payload size in memory = application-programmed initial transfer size - core updated final transfer size
 - c) Number of USB packets in which this payload was received = application-programmed initial packet count - core updated final packet count.
 - d) If for some reason, the host stop sending tokens, there will be no interrupt to the application, and the application must timeout on its own.
 5. The assertion of the DOEPINTx.XferCompl can also mark a packet drop on USB due to unavailability of space in the RxFifo or due to any packet errors.
 - a) The application must read the DOEPINTx.PktDrpSts (DOEPINTx.Bit[11] is now used as the DOEPINTx.PktDrpSts) register to differentiate whether the DOEPINTx.XferCompl was generated due to the normal end of transfer or due to dropped packets. In case of packets being dropped on the USB due to unavailability of space in the RxFifo or due to any packet errors the endpoint enable bit is cleared.
 - b) In case of packet drop on the USB application must re-enable the endpoint after recalculating the values DOEPTSIZx.XferSize and DOEPTSIZx.PktCnt.
 - c) Payload size in memory = application-programmed initial transfer size - core updated final transfer size
 - d) Number of USB packets in which this payload was received = application-programmed initial packet count - core updated final packet count.

Note: Due to application latencies it is possible that DOEPINT.XferComplete interrupt is generated without DOEPINT.PktDrpSts being set, This scenario is possible only if back-to-back packets are dropped for consecutive frames and the PktDrpSts is merged, but the XferSize and PktCnt values for the endpoint are nonzero. In this

case, the application must proceed further by programming the `PktCnt` and `XferSize` register for the next frame, as it would if `PktDrpSts` were being set.

Figure 16-29 gives the application flow for Isochronous OUT Periodic Transfer Interrupt feature.

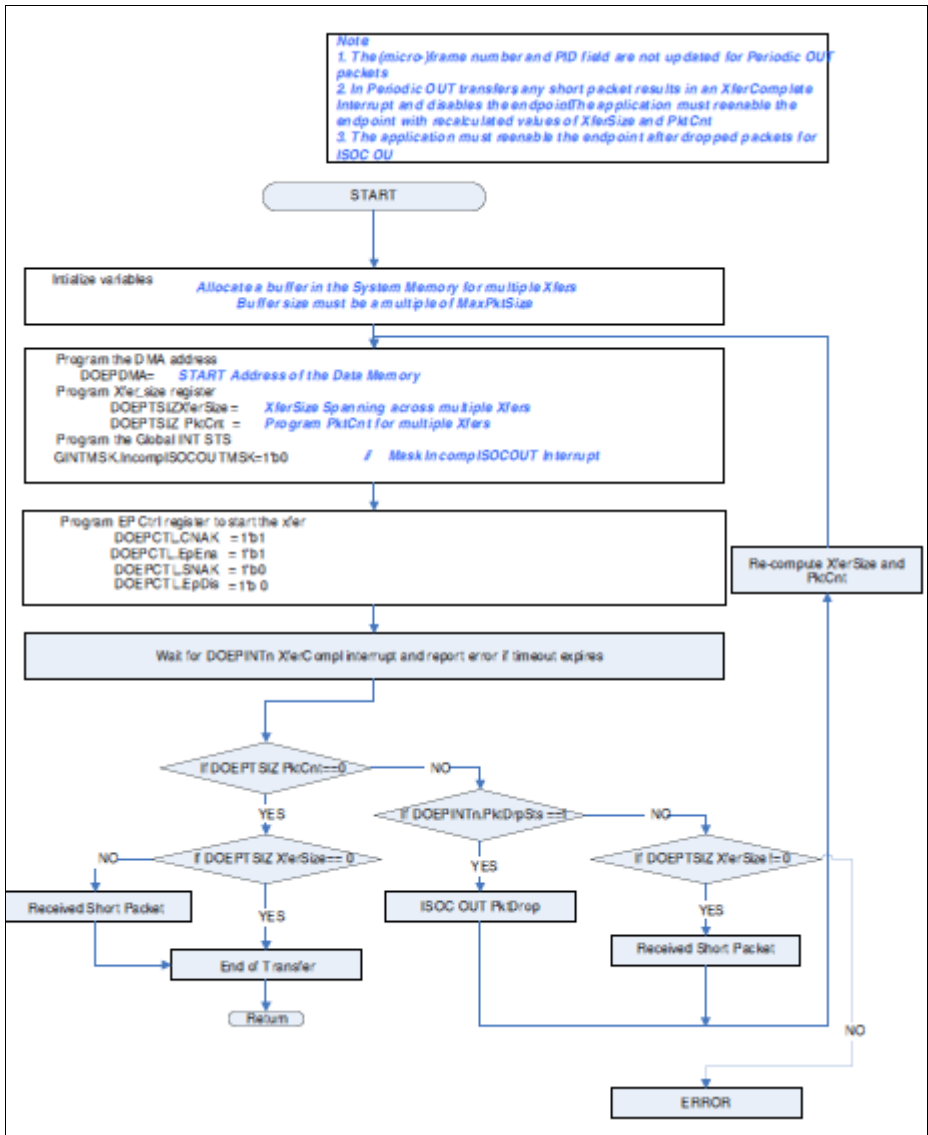


Figure 16-29 ISOC OUT Application Flow for Periodic Transfer Interrupt Feature

Internal Data Flow

1. The application must set the Transfer Size, Packets to be received in a frame and Packet Count Fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. When an isochronous OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame will be ignored by the core.
3. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the Packet Count field for that endpoint by 1.
4. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the ISOC packets are ignored and not written to the receive FIFO.
5. After the data is written to the receive FIFO, the core's DMA engine, reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
6. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
7. The OUT Data Transfer Completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions.
 - a) The transfer size is 0 and the packet count is 0
 - b) The last OUT data packet written to the receive FIFO is a short packet ($0 < \text{packet size} < \text{maximum packet size}$).
8. When the DMA pops this entry (OUT Data Transfer Completed), a Transfer Completed interrupt is generated for the endpoint or the endpoint enable is cleared.
9. OUT data packets received with Bad Data CRC or any packet error are flushed from the receive FIFO automatically.

In these two cases, the packet count and transfer size registers are not decremented because no data is written to the receive FIFO. [Figure 16-30](#) illustrates the internal data flow.

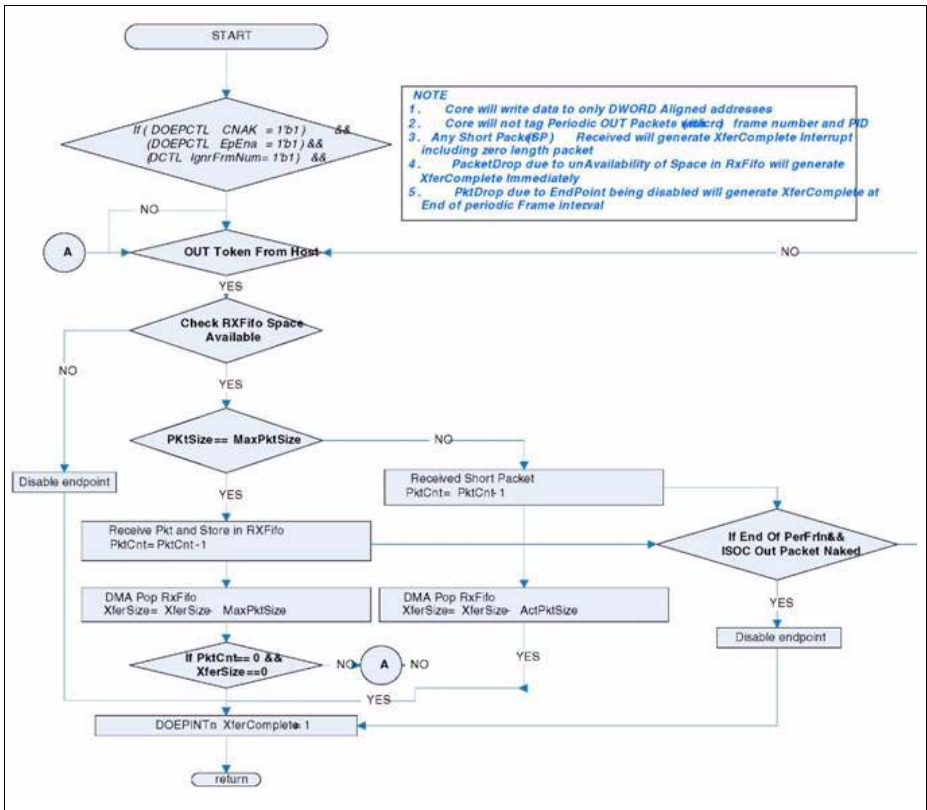


Figure 16-30 Isochronous OUT Core Internal Flow for Periodic Transfer Interrupt Feature

16.9.8 Incomplete Isochronous OUT Data Transfers

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal Data Flow

1. For isochronous OUT endpoints, the DOEPINTx.XferCompl interrupt possibly is not always asserted. If the core drops isochronous OUT data packets, the application could fail to detect the DOEPINTx.XferCompl interrupt under the following circumstances.

Universal Serial Bus (USB)

- a) When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data.
 - b) When the isochronous OUT data packet is received with CRC errors
 - c) When the isochronous OUT token received by the core is corrupted
 - d) When the application is very slow in reading the data from the receive FIFO
2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the GINTSTS.incomplete Isochronous OUT data interrupt, indicating that a DOEPINTx.XferCompl interrupt is not asserted on at least one of the isochronous OUT endpoints. At this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remains in progress on this endpoint on the USB.
 3. This step is applicable only if the USB core is operating in slave mode.

Application Programming Sequence

1. Asserting the GINTSTS.incomplete Isochronous OUT data interrupt indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.
2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must empty all isochronous OUT data (data and status) from the receive FIFO before proceeding.
 - a) When all data is emptied from the receive FIFO, the application can detect the DOEPINTx.XferCompl interrupt. In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame, as described in **“Isochronous OUT Data Transfers” on Page 16-108**.
3. When it receives a GINTSTS.incomplete Isochronous OUT data interrupt, the application must read the control registers of all isochronous OUT endpoints (DOEPCTLx) to determine which endpoints had an incomplete transfer in the current frame. An endpoint transfer is incomplete if both the following conditions are met.
 - a) DOEPCTLx.Even/Odd frame bit = DSTS.SOFFN[0]
 - b) DOEPCTLx.Endpoint Enable = 1
4. The previous step must be performed before the GINTSTS.SOF interrupt is detected, to ensure that the current frame number is not changed.
5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the DOEPCTLx.Endpoint Disable bit.
6. Wait for the DOEPINTx.Endpoint Disabled interrupt and enable the endpoint to receive new data in the next frame as explained in **“Isochronous OUT Data Transfers” on Page 16-108**.

Because the core can take some time to disable the endpoint, the application possibly is not able to receive the data in the next frame after receiving bad isochronous data.

16.9.9 Incomplete Isochronous IN Data Transfers

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal Data Flow

1. An isochronous IN transfer is treated as incomplete in one of the following conditions.
 - a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects a GINTSTS.incomplete Isochronous IN Transfer interrupt.
 - b) The application or DMA is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects a DIEPINTx.IN Tkn Rcvd When TxFIFO Empty interrupt. The application can ignore this interrupt, as it eventually results in a GINTSTS.incomplete Isochronous IN Transfer interrupt at the end of periodic frame.
 - The core transmits a zero-length data packet on the USB in response to the received IN token.
2. In either of the aforementioned cases, in Slave mode, the application must stop writing the data payload to the transmit FIFO as soon as possible.
3. The application must set the NAK bit and the disable bit for the endpoint.
4. The core disables the endpoint, clears the disable bit, and asserts the Endpoint Disable interrupt for the endpoint.

Application Programming Sequence

1. The application can ignore the DIEPINTx.IN Tkn Rcvd When TxFIFO empty interrupt on any isochronous IN endpoint, as it eventually results in a GINTSTS.incomplete Isochronous IN Transfer interrupt.
2. Assertion of the GINTSTS.incomplete Isochronous IN Transfer interrupt indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the Endpoint Control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. In Slave mode, the application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.
5. In both modes of operation, program the following fields in the DIEPCTLx register to disable the endpoint. See **“IN Endpoint Disable” on Page 16-80** for more details.
 - a) DIEPCTLx.SetNAK = 1
 - b) DIEPCTLx.Endpoint Disable = 1
6. The DIEPINTx.Endpoint Disabled interrupt's assertion indicates that the core has disabled the endpoint.

7. At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next frame. To flush the data, the application must use the GRSTCTL register.

16.9.10 Periodic IN (Interrupt and Isochronous) Data Transfers

This section describes a typical periodic IN data transfer.

Application Requirements

1. Application requirements 1, 2, 3, and 4 of **“Non-Periodic (Bulk and Control) IN Data Transfers” on Page 16-98** also apply to periodic IN data transfers, except for a slight modification of Requirement 2.
 - a) The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To transmit a few maximum- packet-size packets and a short packet at the end of the transfer, the following conditions must be met.
 - $\text{transfer size}[\text{epnum}] = n * \text{mps}[\text{epnum}] + \text{sp}$
(where n is an integer > 0, and $0 < \text{sp} < \text{mps}[\text{epnum}]$)
 - If $(\text{sp} > 0)$, $\text{packet count}[\text{epnum}] = n + 1$
Otherwise, $\text{packet count}[\text{epnum}] = n$;
 - $\text{mc}[\text{epnum}] = \text{packet count}[\text{epnum}]$
 - b) The application cannot transmit a zero-length data packet at the end of transfer. It can transmit a single zero-length data packet by it self. To transmit a single zero-length data packet,
 - c) $\text{transfer size}[\text{epnum}] = 0$
 - $\text{packet count}[\text{epnum}] = 1$
 - $\text{mc}[\text{epnum}] = \text{packet count}[\text{epnum}]$
2. The application can only schedule data transfers 1 frame at a time.
 - a) $(\text{DIEPTSIZx.MC} - 1) * \text{DIEPCTLx.MPS} < \text{DIEPTSIZx.XferSiz} < \text{DIEPTSIZx.MC} * \text{DIEPCTLx.MPS}$
 - b) $\text{DIEPTSIZx.PktCnt} = \text{DIEPTSIZx.MC}$
 - c) If $\text{DIEPTSIZx.XferSiz} < \text{DIEPTSIZx.MC} * \text{DIEPCTLx.MPS}$, the last data packet of the transfer is a short packet.
3. This step is not applicable for isochronous data transfers, only for interrupt transfers. The application can schedule data transfers for multiple frames, only if multiples of max packet sizes (up to 3 packets), must be transmitted every frame. This is can be done, only when the core is operating in DMA mode. This is not a recommended mode though.
 - a) $((n * \text{DIEPTSIZx.MC}) - 1) * \text{DIEPCTLx.MPS} \leq \text{DIEPTSIZx.Transfer Size} \leq n * \text{DIEPTSIZx.MC} * \text{DIEPCTLx.MPS}$
 - b) $\text{DIEPTSIZx.Packet Count} = n * \text{DIEPTSIZx.MC}$
 - c) n is the number of frames for which the data transfers are scheduled

Universal Serial Bus (USB)

Data Transmitted per frame in this case would be $\text{DIEPTSIZE} \times \text{MC} \times \text{DIEPCTL} \times \text{MPS}$, in all the frames except the last one. In the frame "n", the data transmitted would be $(\text{DIEPTSIZE} \times \text{TransferSize} - (n-1) \times \text{DIEPTSIZE} \times \text{MC} \times \text{DIEPCTL} \times \text{MPS})$

4. For Periodic IN endpoints, the data must always be prefetched 1 frame ahead for transmission in the next frame. This can be done, by enabling the Periodic IN endpoint 1 frame ahead of the frame in which the data transfer is scheduled.
5. The complete data to be transmitted in the frame must be written into the transmit FIFO (either by the application or the DMA), before the Periodic IN token is received. Even when 1 DWORD of the data to be transmitted per frame is missing in the transmit FIFO when the Periodic IN token is received, the core behaves as when the FIFO was empty. When the transmit FIFO is empty, a zero data length packet would be transmitted on the USB for ISO IN endpoints. A NAK handshake is transmitted on the USB for INTR IN endpoints.
6. For a High Bandwidth IN endpoint with three packets in a frame, the application can program the endpoint FIFO size to be $2 \times \text{max_pkt_size}$ and have the third packet load in after the first packet has been transmitted on the USB.

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the associated transmit FIFO for the endpoint.
3. Every time either the core's internal DMA (in DMA mode) or the application (in Slave mode) writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data is fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for an periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO mode) for the frame is not present in the FIFO, then the core generates an IN Tkn Rcvd When TxF Empty Interrupt for the endpoint.
 - a) A zero-length data packet is transmitted on the USB for isochronous IN endpoints
 - b) A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. The packet count for the endpoint is decremented by 1 under the following conditions:
 - a) For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
 - b) For interrupt endpoints, when an ACK handshake is transmitted
 - c) When the transfer size and packet count are both 0, the Transfer Completed interrupt for the endpoint is generated and the endpoint enable is cleared.

6. At the "Periodic frame Interval" (controlled by DCFG.PerFrint), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates a GINTSTS.incomplISOIN interrupt.

Application Programming Sequence (Transfer Per Frame)

1. Program the DIEPTSIZx register.
2. Program the DIEPCTLx register with the endpoint characteristics and set the CNAK and Endpoint Enable bits.
3. In Slave mode, write the data to be transmitted in the next frame to the transmit FIFO as described in **"Packet Write" on Page 16-93**.
4. Asserting the DIEPINTx.In Token Rcvd When TxF Empty interrupt indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
5. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
 - a) If the isochronous endpoint is already enabled when this interrupt is detected, see **"Incomplete Isochronous IN Data Transfers" on Page 16-116** for more details.
6. The core handles timeouts internally, without application intervention. The application, thus, never detects a DIEPINTn.TimeOUT interrupt for periodic interrupt IN endpoints.
7. Asserting the DIEPINTx.XferCompl interrupt with no DIEPINTx.In Tkn Rcvd When TxF Empty interrupt indicates the successful completion of an isochronous IN transfer. A read to the DIEPTSIZx register must indicate transfer size = 0 and packet count = 0, indicating all data is transmitted on the USB.
8. Asserting the DIEPINTx.XferCompl interrupt, with or without the DIEPINTx.In Tkn Rcvd When TxF Empty interrupt, indicates the successful completion of an interrupt IN transfer. A read to the DIEPTSIZx register must indicate transfer size = 0 and packet count = 0, indicating all data is transmitted on the USB.
9. Asserting the GINTSTS.incomplete Isochronous IN Transfer interrupt with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

For isochronous IN endpoints, see **"Incomplete Isochronous IN Data Transfers" on Page 16-116**, for more details.

16.9.11 Periodic IN Data Transfers Using the Periodic Transfer Interrupt

This section describes a typical Periodic IN (ISOC / INTR) data transfer with the Periodic Transfer Interrupt feature.

1. For IN transfers, the Transfer Size field in the Endpoint Transfer Size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.

Universal Serial Bus (USB)

- a) To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
 - Transfer size[epnum] = $n * mps[epnum] + sp$
(where n is an integer > 0 , and $0 < sp < mps[epnum]$. A higher value of n reduces the periodicity of the DOEPINTx.XferCompl interrupt)
 - If ($sp > 0$), then packet count[epnum] = $n + 1$. Otherwise, packet count[epnum] = n
 - b) To transmit a single zero-length data packet:
 - Transfer size[epnum] = 0
 - Packet count[epnum] = 1
 - c) To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer in two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.
 - First transfer: transfer size[epnum] = $n * mps[epnum]$; packet count = n ;
 - Second transfer: transfer size[epnum] = 0; packet count = 1;
 - d) The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met.
 - transfer size[epnum] = $n * mps[epnum] + sp$ (where n is an integer > 0 , and $0 < sp < mps[epnum]$)
 - If ($sp > 0$), packet count[epnum] = $n + 1$ Otherwise, packet count[epnum] = n ;
 - mc[epnum] = number of packets to be sent out in a frame.
 - e) The application cannot transmit a zero-length data packet at the end of transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet,
 - transfer size[epnum] = 0
 - packet count[epnum] = 1
 - mc[epnum] = packet count[epnum]
2. Once an endpoint is enabled for data transfers, the core updates the Transfer Size register. At the end of IN transfer, which ended with an Endpoint Disabled interrupt, the application must read the Transfer Size register to determine how much data posted in the transmit FIFO was already sent on the USB.
 - a) Data fetched into transmit FIFO = Application-programmed initial transfer size - core-updated final transfer size
 - b) Data transmitted on USB = (application-programmed initial packet count - Core updated final packet count) * mps[epnum]
 - c) Data yet to be transmitted on USB = (Application-programmed initial transfer size - data transmitted on USB)
 3. For Periodic IN endpoints, the data must always be prefetched 1 frame ahead for transmission in the next frame. This can be done, by enabling the Periodic IN endpoint 1 frame ahead of the frame in which the data transfer is scheduled.

Universal Serial Bus (USB)

4. The complete data to be transmitted in the frame must be written into the transmit FIFO, before the Periodic IN token is received. Even when 1 DWORD of the data to be transmitted per frame is missing in the transmit FIFO when the Periodic IN token is received, the core behaves as when the FIFO was empty. When the transmit FIFO is empty,
 - a) A zero data length packet would be transmitted on the USB for ISOC IN endpoints
 - b) A NAK handshake would be transmitted on the USB for INTR IN endpoints
 - c) DIEPTSIZx.PktCnt is not decremented in this case.

For a High Bandwidth IN endpoint with three packets in a frame, the application can program the endpoint FIFO size to be $2 * \text{max_pkt_size}$ and have the third packet load in after the first packet has been transmitted on the USB.

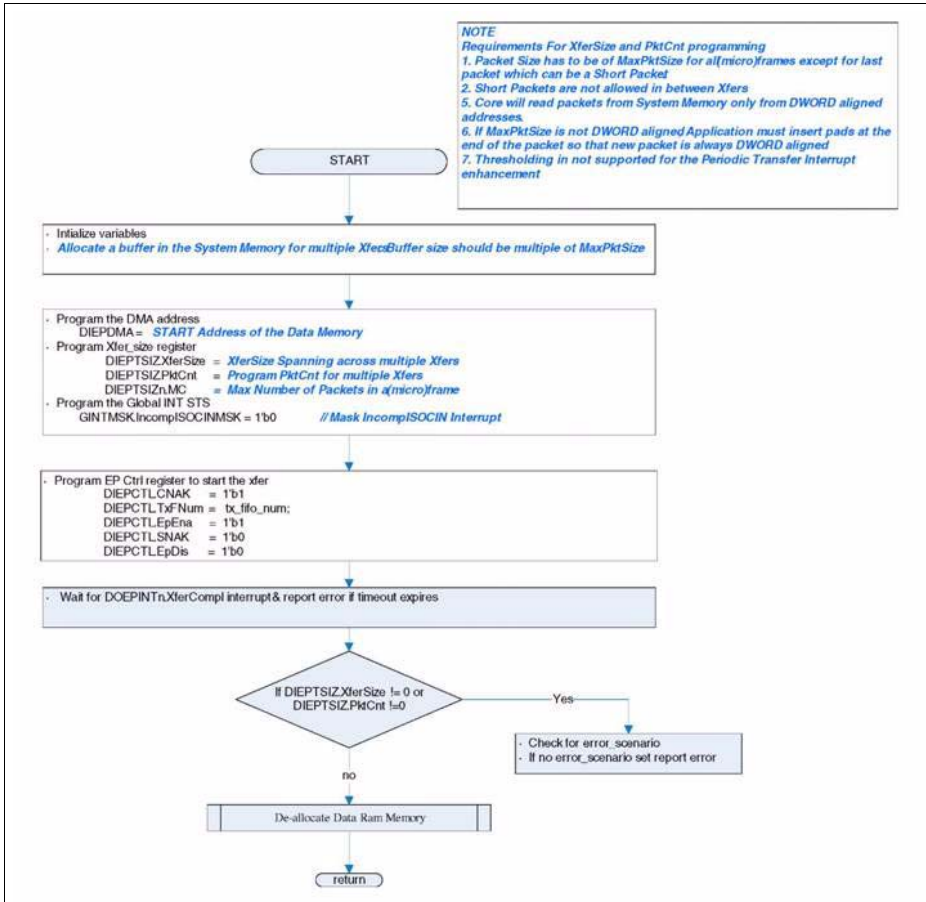


Figure 16-31 Periodic IN Application Flow for Periodic Transfer Interrupt Feature

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
 - a) The application must enable the DCTL.IgnrFrmNum
2. When an isochronous OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame will be ignored by the core. Subsequently the core updates the Even / Odd bit on its own.
3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data is fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete packet for the frame is not present in the FIFO, then the core generates an IN Tkn Rcvd When TxFifo Empty Interrupt for the endpoint.
 - a) A zero-length data packet is transmitted on the USB for isochronous IN endpoints
 - b) A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. If an IN token comes for an endpoint on the bus, and if the corresponding TxFIFO for that endpoint has at least 1 packet available, and if the DIEPCTLx.NAK bit is not set, and if the internally maintained even/odd bit match with the bit 0 of the current frame number, then the core will send this data out on the USB. The core will also decrement the packet count. Core also toggles the MultCount in DIEPCTLx register and based on the value of MultCount the next PID value is sent.
 - a) If the IN token results in a timeout (core did not receive the handshake or handshake error), core rewind the FIFO pointers. Core does not decrement packet count. It does not toggle PID. DIEPINTx.TimeOut interrupt will be set which the application could check.
 - b) At the end of periodic frame interval (Based on the value programmed in the DCFG.PerFrint register, core will internally set the even/ odd internal bit to match the next frame.
6. The packet count for the endpoint is decremented by 1 under the following conditions:
 - a) For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
 - b) For interrupt endpoints, when an ACK handshake is transmitted
7. The data PID of the transmitted data packet is based on the value of DIEPTSIZx.MC programmed by the application. In case the DIEPTSIZx.MC value is set to 3 then, for a particular frame the core expects to receive 3 Isochronous IN token for the respective endpoint. The data PIDs transmitted will be D2 followed by D1 and D0 respectively for the tokens.
 - a) If any of the tokens responded with a zero-length packet due to non-availability of data in the TxFIFO, the packet is sent in the next frame with the pending data PID. For example, in a frame, the first received token is responded to with data and data

Universal Serial Bus (USB)

PID value D2. If the second token is responded to with a zero-length packet, the host is expected not to send any more tokens for the respective endpoint in the current frame. When a token arrives in the next frame it will be responded to with the pending data PID value of D1.

- b) Similarly the second token of the current frame gets responded with D0 PID. The host is expected to send only two tokens for this frame as the first token got responded with D1 PID.
- 8. When the transfer size and packet count are both 0, the Transfer Completed interrupt for the endpoint is generated and the endpoint enable is cleared.
- 9. The GINTSTS.incomplISOIN will be masked by the application hence at the Periodic Frame interval (controlled by DCFG.PerFrint), even though the core finds non-empty any of the isochronous IN endpoint FIFOs, GINTSTS.incomplISOIN interrupt will not be generated.

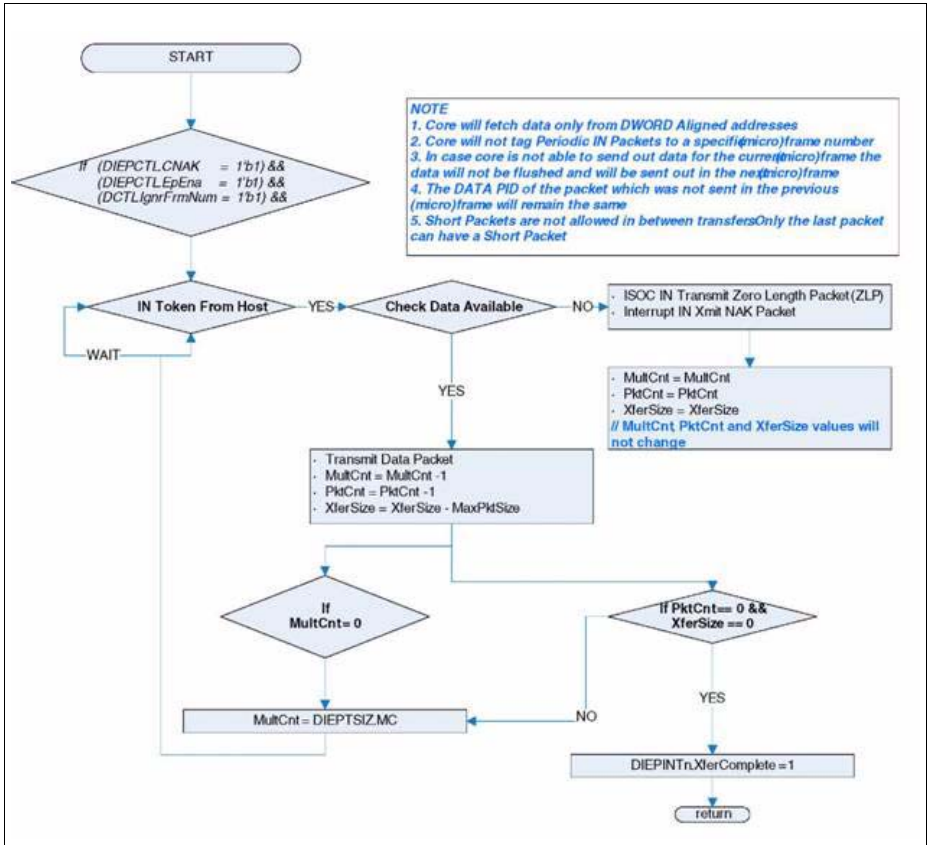


Figure 16-32 Periodic IN Core Internal Flow for Periodic Transfer Interrupt Feature

16.9.12 Interrupt OUT Data Transfers Using Periodic Transfer Interrupt

This section describes a regular INTR OUT data transfer with the Periodic Transfer Interrupt feature.

Application Requirements

1. Before setting up a periodic OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer, then program that buffer's size and start address in the endpoint-specific registers.
2. For Interrupt OUT transfers, the Transfer Size field in the endpoint's Transfer Size register must be a multiple of the maximum packet size of the endpoint, adjusted to

Universal Serial Bus (USB)

the DWORD boundary. The Transfer Size programmed can span across multiple frames based on the periodicity after which the application want to receive the DOEPINTx.XferCompl interrupt

- a) $\text{transfer size}[\text{epnum}] = n * (\text{mps}[\text{epnum}] + 4 - (\text{mps}[\text{epnum}] \bmod 4))$
 - b) $\text{packet count}[\text{epnum}] = n$
 - c) $n > 0$ (Higher value of n reduces the periodicity of the DOEPINTx.XferCompl interrupt)
 - d) $1 < \text{packet count}[\text{epnum}] < n$ (Higher value of n reduces the periodicity of the DOEPINTx.XferCompl interrupt)
3. On DOEPINTx.XferCompl interrupt, the application must read the endpoint's Transfer Size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
 - a) Payload size in memory = application-programmed initial transfer size - core updated final transfer size
 - b) Number of USB packets in which this payload was received = application-programmed initial packet count - core updated final packet count.
 - c) If for some reason, the host stops sending tokens, there are no interrupts to the application, and the application must timeout on its own.
 4. The assertion of the DOEPINTx.XferCompl interrupt marks the completion of the interrupt OUT data transfer. This interrupt does not necessarily mean that the data in memory is good.
 5. Read the DOEPTSiZx register to determine the size of the received transfer and to determine the validity of the data received in the frame.

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
 - a) The application must enable the DCTL.IgnrFrmNum
2. When an interrupt OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame will be ignored by the core.
3. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the Packet Count field for that endpoint by 1.
 - a) OUT data packets received with Bad Data CRC or any packet error are flushed from the receive FIFO automatically.
 - b) Interrupt packets with PID errors are not passed to application. Core discards the packet, sends ACK and does not decrement packet count.
 - c) If there is no space in the receive FIFO, interrupt data packets are ignored and not written to the receive FIFO. Additionally, interrupt OUT tokens receive a NAK handshake reply.

Universal Serial Bus (USB)

4. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or interrupt data packets are ignored and not written to the receive FIFO, and interrupt OUT tokens receive a NAK handshake reply.
5. After the data is written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
6. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
7. The OUT Data Transfer Completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions.
 - a) The transfer size is 0 and the packet count is 0.
 - b) The last OUT data packet written to the receive FIFO is a short packet ($0 < \text{packet size} < \text{maximum packet size}$)
8. When the application pops this entry (OUT Data Transfer Completed), a Transfer Completed interrupt is generated for the endpoint and the endpoint enable is cleared.

16.10 Device Programming in Buffer DMA Mode

This section discusses how to program the core when it is acting as a Device in the Slave mode of operation.

16.10.1 Control Transfers

This section describes the various types of control transfers.

16.10.1.1 Control Write Transfers (SETUP, Data OUT, Status IN)

This section describes control write transfers.

Application Programming Sequence

1. Assertion of the DOEPINTx.SETUP Packet interrupt indicates that a valid SETUP packet has been transferred to the application. At the end of the Setup stage, the application must reprogram the DOEPTSIZx.SUPCnt field to 3 to receive the next SETUP packet.
2. If the last SETUP packet received before the assertion of the SETUP interrupt indicates a data OUT phase, program the core to perform a control OUT transfer as explained in [“Non-Isochronous OUT Data Transfers” on Page 16-134](#). The application must reprogram the DOEPDMAx register to receive a control OUT data packet to a different memory location.
3. In a single OUT data transfer on control endpoint 0, the application can receive up to 64 bytes. If the application is expecting more than 64 bytes in the Data OUT stage, the application must re-enable the endpoint to receive another 64 bytes, and must continue to do so until it has received all the data in the Data stage.

Universal Serial Bus (USB)

4. Assertion of the DOEPINTx.Transfer Compl interrupt on the last data OUT transfer indicates the completion of the data OUT phase of the control transfer.
5. On completion of the data OUT phase, the application must do the following.
 - a) To transfer a new SETUP packet in DMA mode, the application must re-enable the control OUT endpoint as explained in section in section **“OUT Data Transfers” on Page 16-129**.
 - DOEPCTLx.EPEna = 1_B
 - b) To execute the received Setup command, the application must program the required registers in the core. This step is optional, based on the type of Setup command received.
6. For the status IN phase, the application must program the core as described in **“Non-Periodic (Bulk and Control) IN Data Transfers” on Page 16-132** to perform a data IN transfer.
7. Assertion of the DIEPINTx.Transfer Compl interrupt indicates completion of the status IN phase of the control transfer.

16.10.1.2 Control Read Transfers (SETUP, Data IN, Status OUT)

This section describes control write transfers.

Application Programming Sequence

1. Assertion of the DOEPINTx.SETUP Packet interrupt indicates that a valid SETUP packet has been transferred to the application. At the end of the Setup stage, the application must reprogram the DOEPTSIZx.SUPCnT field to 3 to receive the next SETUP packet.
2. If the last SETUP packet received before the assertion of the SETUP interrupt indicates a data IN phase, program the core to perform a control IN transfer as explained in **“Non-Periodic (Bulk and Control) IN Data Transfers” on Page 16-132**.
3. On a single IN data transfer on control endpoint 0, the application can transmit up to 64 bytes. To transmit more than 64 bytes in the Data IN stage, the application must re-enable the endpoint to transmit another 64 bytes, and must continue to do so, until it has transmitted all the data in the Data stage.
4. The DIEPINTx.Transfer Compl interrupt on the last IN data transfer marks the completion of the control transfer's Data stage.
5. To perform a data OUT transfer in the status OUT phase, the application must program the core as described in **“OUT Data Transfers” on Page 16-129**.
 - a) The application must program the DCFG.NZStsOUTHShk handshake field to a proper setting before transmitting an data OUT transfer for the Status stage.
 - b) The application must then reprogram the DOEPDMAn register to receive the control OUT data packet to a different memory location.

Universal Serial Bus (USB)

6. Assertion of the DOEPINTx.Transfer Compl interrupt indicates completion of the status OUT phase of the control transfer. This marks the successful completion of the control read transfer.
 - a) To transfer a new SETUP packet, the application must re-enable the control OUT endpoint as explained in **“OUT Data Transfers” on Page 16-129**.
 - b) $DOEPCTLn.EPEna = 1_B$

16.10.1.3 Two-Stage Control Transfers (SETUP/Status IN)

This section describes two-stage control transfers.

Application Programming Sequence

1. Assertion of the DOEPINTx.Setup interrupt indicates that a valid SETUP packet has been transferred to the application. To receive the next SETUP packet, the application must reprogram the DOEPSIZx.SUPCnt field to 3 at the end of the Setup stage.
2. Decode the last SETUP packet received before the assertion of the SETUP interrupt. If the packet indicates a two-stage control command, the application must do the following.
 - a) To transfer a new SETUP packet in DMA mode, the application must re-enable the control OUT endpoint. For more information, see **“OUT Data Transfers” on Page 16-129**.
 - Set $DOEPCTLx.EPEna = 1_B$
 - b) Depending on the type of Setup command received, the application can be required to program registers in the core to execute the received Setup command.
3. For the status IN phase, the application must program the core described in **“Non-Periodic (Bulk and Control) IN Data Transfers” on Page 16-132** to perform a data IN transfer.
4. Assertion of the DIEPINTx.Transfer Compl interrupt indicates the completion of the status IN phase of the control transfer.

16.10.2 OUT Data Transfers

This section describes the internal data flow and application-level operations during data OUT transfers and setup transactions.

16.10.2.1 Control Setup Transactions

This section describes how the core handles SETUP packets and the application's sequence for handling setup transactions. To initialize the core after power-on reset, the application must follow the sequence in **“Core Initialization” on Page 16-11**. Before it can communicate with the host, it must initialize an endpoint as described in **“Endpoint Initialization” on Page 16-74**. See **“Packet Read from FIFO” on Page 16-91**.

Application Requirements

1. To receive a SETUP packet, the DOEPTSIz.SUPCnt field in a control OUT endpoint must be programmed to a non-zero value. When the application programs the SUPCnt field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the DOEPTLx.NAK status and DOEPTLx.EPEna bit setting. The SUPCnt field is decremented every time the control endpoint receives a SETUP packet. If the SUPCnt field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the SUPCnt field, but the application possibly is not be able to determine the correct number of SETUP packets received in the Setup stage of a control transfer.
 - a) DOEPTSIz.SUPCnt = 3
2. In DMA mode, the OUT endpoint must also be enabled, to transfer the received SETUP packet data from the internal receive FIFO to the external memory.
 - a) DOEPTLn.EPEna = 1_B
3. The application must always allocate some extra space in the Receive Data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
 - a) The space to be Reserved is 10 DWORDs. Three DWORDs are required for the first SETUP packet, 1 DWORD is required for the Setup Stage Done DWORD, and 6 DWORDs are required to store two extra SETUP packets among all control endpoints.
 - b) 3 DWORDs per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (Setup Packet Pattern). The core reserves this space in the receive data
 - c) FIFO to write SETUP data only, and never uses this space for data packets.
4. The core writes the 2 DWORDs of the SETUP data to the memory.
5. The application must read and discard the Setup Stage Done DWORD from the receive FIFO.

Internal Data Flow

1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and Stall bit settings.
 - a) The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB, 3 DWORDs of data is written to the receive FIFO, and the SUPCnt field is decremented by 1.
 - a) The first DWORD contains control information used internally by the core
 - b) The second DWORD contains the first 4 bytes of the SETUP command
 - c) The third DWORD contains the last 4 bytes of the SETUP command

Universal Serial Bus (USB)

3. When the Setup stage changes to a Data IN/OUT stage, the core writes an entry (Setup Stage Done DWORD) to the receive FIFO, indicating the completion of the Setup stage.
4. On the AHB side, SETUP packets are emptied either by the DMA or the application. In DMA mode, the SETUP packets (2 DWORDs) are written to the memory location programmed in the DOEPDMA_n register, only if the endpoint is enabled. If the endpoint is not enabled, the data remains in the receive FIFO until the enable bit is set.
5. When either the DMA or the application pops the Setup Stage Done DWORD from the receive FIFO, the core interrupts the application with a DOEPINT_n.SETUP interrupt, indicating it can process the received SETUP packet.
6. The core clears the endpoint enable bit for control OUT endpoints.

Application Programming Sequence

1. Program the DOEPTSI_{Zx} register.
 - a) DOEPTSI_{Zx}.SUPCnt = 3
2. Program the DOEPDMA_n register and DOEPCTL_n register with the endpoint characteristics and set the Endpoint Enable bit (DOEPCTL_n.EPEna).
 - a) Endpoint Enable = 1
3. Assertion of the DOEPINT_x.SETUP interrupt marks a successful completion of the SETUP Data Transfer.
 - a) On this interrupt, the application must read the DOEPTSI_{Zx} register to determine the number of SETUP packets received and process the last received SETUP packet.
 - b) In DMA mode, the application must also determine if the interrupt bit DOEPINT_n.Back2BackSETup is set. This bit is set if the core has received more than three back-to-back SETUP packets. If this is the case, the application must ignore the DOEPTSI_{Zn}.SUPCnt value and use the DOEPDMA_n directly to read out the last SETUP packet received. DOEPDMA₈ provides the pointer to the last valid SETUP data.

Note: If the application has not enabled EP0 before the host sends the SETUP packet, the core ACKs the SETUP packet and stores it in the FIFO, but does not write to the memory until EP0 is enabled. When the application enables the EP0 (first enable) and clears the NAK bit at the same time the Host sends DATA OUT, the DATA OUT is stored in the Rx FIFO. The USB core then writes the setup data to the memory and disables the endpoint. Though the application expects a Transfer Complete interrupt for the Data OUT phase, this does not occur, because the SETUP packet, rather than the DATA OUT packet, enables EP0 the first time. Thus, the DATA OUT packet is still in the Rx FIFO until the application re-enables EP0. The application must enable EP0 one more time for the core to process the DATA OUT packet.

Figure 16-24 charts this flow.

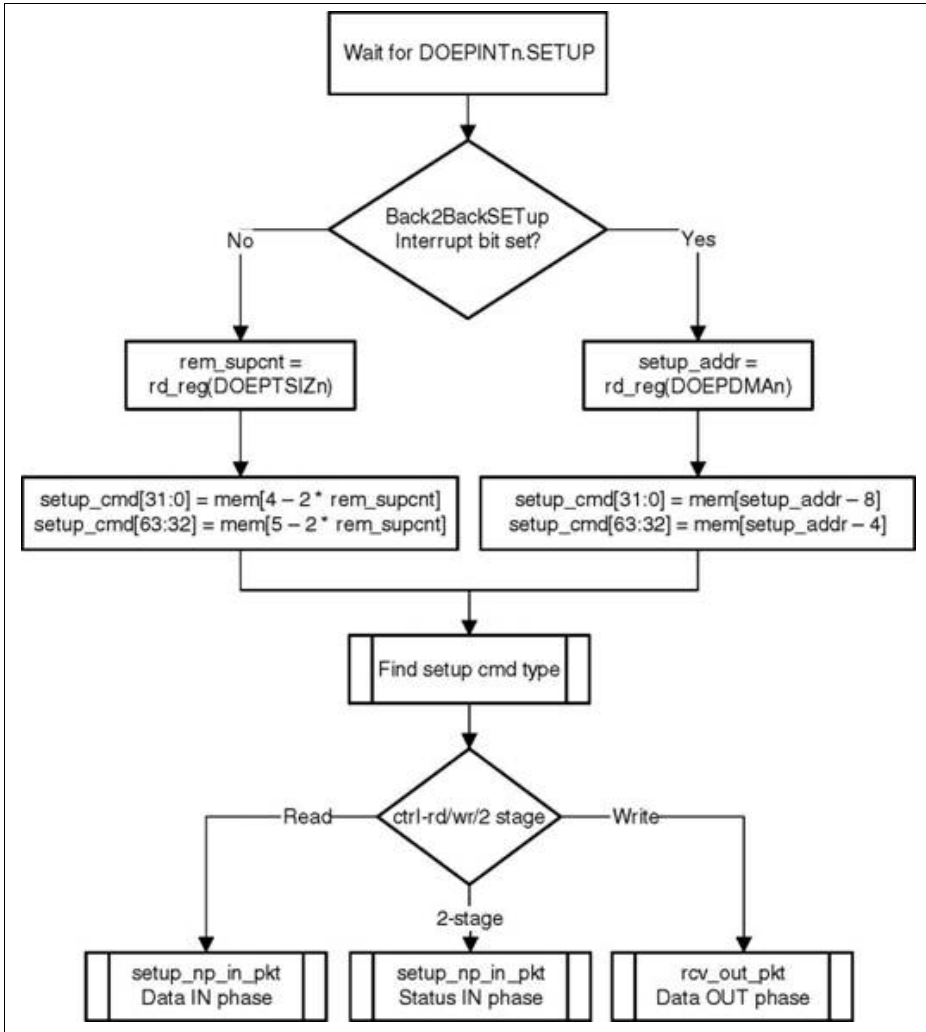


Figure 16-33 Processing a SETUP Packet

16.10.3 Non-Periodic (Bulk and Control) IN Data Transfers

This section describes a regular non-periodic IN data transfer.

Application Requirements

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer is part of a single buffer, and must program the size of that buffer and its start address (in DMA mode) to the endpoint-specific registers.
2. For IN transfers, the Transfer Size field in the Endpoint Transfer Size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
 - a) To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
 - b) $\text{Transfer size}[\text{epnum}] = n * \text{mps}[\text{epnum}] + \text{sp}$
(where n is an integer > 0 , and $0 < \text{sp} < \text{mps}[\text{epnum}]$)
- If $(\text{sp} > 0)$, then $\text{packet count}[\text{epnum}] = n + 1$.
Otherwise, $\text{packet count}[\text{epnum}] = n$
 - c) To transmit a single zero-length data packet:
- $\text{Transfer size}[\text{epnum}] = 0$
- $\text{Packet count}[\text{epnum}] = 1$
 - d) To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer in two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.
- First transfer: $\text{transfer size}[\text{epnum}] = n * \text{mps}[\text{epnum}]$; $\text{packet count} = n$;
- Second transfer: $\text{transfer size}[\text{epnum}] = 0$; $\text{packet count} = 1$;
3. In DMA mode, the core fetches an IN data packet from the memory, always starting at a DWORD boundary. If the maximum packet size of the IN endpoint is not a multiple of 4, the application must arrange the data in the memory with pads inserted at the end of a maximum-packet-size packet so that a new packet always starts on a DWORD boundary.
4. Once an endpoint is enabled for data transfers, the core updates the Transfer Size register. At the end of IN transfer, which ended with an Endpoint Disabled interrupt, the application must read the Transfer Size register to determine how much data posted in the transmit FIFO was already sent on the USB.
5. Data fetched into transmit FIFO = Application-programmed initial transfer size - core-updated final transfer size
 - a) $\text{Data transmitted on USB} = (\text{application-programmed initial packet count} - \text{Core updated final packet count}) * \text{mps}[\text{epnum}]$
 - b) $\text{Data yet to be transmitted on USB} = (\text{Application-programmed initial transfer size} - \text{data transmitted on USB})$

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers and enable the endpoint to transmit the data.

Universal Serial Bus (USB)

2. The core fetches the data from memory according to the application setting for the endpoint.
3. Every time the core's internal DMA writes a packet into the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the "number of packets in FIFO" count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data is written to the transmit FIFO, the core reads it out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a TIMEOUT.
5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the Packet Count field.
6. If there is no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates a IN Tkn Rcvd When FIFO Empty Interrupt for the endpoint, provided the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. In Dedicated FIFO operation, the core internally rewinds the FIFO pointers and no timeout interrupt is generated except for Control IN endpoint.
8. When the transfer size is 0 and the packet count is 0, the transfer complete interrupt for the endpoint is generated and the endpoint enable is cleared.

Application Programming Sequence

1. Program the DIEPTSIZx register with the transfer size and corresponding packet count. Program also the DIEPDMAx register.
2. Program the DIEPCTLx register with the endpoint characteristics and set the CNAK and Endpoint Enable bits.
3. In DMA mode, ensure that the NextEp field is programmed so that the core fetches the data for IN endpoints in the correct order. See **“Non-Periodic IN Endpoint Sequencing” on Page 16-82** for details.
 - a) This step can be repeated multiple times, depending on the transfer size.

16.10.4 Non-Isynchronous OUT Data Transfers

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application Requirements

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer, then program that buffer's size and start address (in DMA mode) in the endpoint-specific registers.
2. For OUT transfers, the Transfer Size field in the endpoint's Transfer Size register must be a multiple of the maximum packet size of the endpoint, adjusted to the DWORD boundary.

```
if (mps[epnum] mod 4) == 0
transfer size[epnum] = n * (mps[epnum] //DWORD aligned)
else
transfer size[epnum] = n * (mps[epnum] + 4 - (mps[epnum] mod 4))
//Non-DWORD aligned
packet count[epnum] = n
n > 0
```

3. In DMA mode, the core stores a received data packet in the memory, always starting on a DWORD boundary. If the maximum packet size of the endpoint is not a multiple of 4, the core inserts byte pads at end of a maximum-packet-size packet up to the end of the DWORD.
4. On any OUT endpoint interrupt, the application must read the endpoint's Transfer Size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
 - a) Payload size in memory = application-programmed initial transfer size - core updated final transfer size
 - b) Number of USB packets in which this payload was received = application-programmed initial packet count - core updated final packet count

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the Packet Count field for that endpoint by 1.
 - a) OUT data packets received with Bad Data CRC are flushed from the receive FIFO automatically.
 - b) After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.

Universal Serial Bus (USB)

- c) If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
- d) In all the above three cases, the packet count is not decremented because no data is written to the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
4. After the data is written to the receive FIFO, the core's DMA engine reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
6. The OUT Data Transfer Completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions.
 - a) The transfer size is 0 and the packet count is 0
 - b) The last OUT data packet written to the receive FIFO is a short packet (0 < packet size < maximum packet size)
7. When either the application or the DMA pops this entry (OUT Data Transfer Completed), a Transfer Completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application Programming Sequence

1. Program the DOEPTSIz register for the transfer size and the corresponding packet count. Additionally, in DMA mode, program the DOEPDMAx register.
2. Program the DOEPCTLx register with the endpoint characteristics, and set the Endpoint Enable and ClearNAK bits.
 - a) DOEPCTLx.EPEna = 1
 - b) DOEPCTLx.CNAK = 1
3. Asserting the DOEPINTx.XferCompl interrupt marks a successful completion of the non- isochronous OUT data transfer.
4. Read the DOEPTSIz register to determine the size of the received data payload.

Note: The XferSize is not decremented for the last packet.

16.10.5 Incomplete Isochronous OUT Data Transfers

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal Data Flow

- For isochronous OUT endpoints, the DOEPINTx.XferCompl interrupt possibly is not always asserted. If the core drops isochronous OUT data packets, the application could fail to detect the DOEPINTx.XferCompl interrupt under the following circumstances.
 - When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data.
 - When the isochronous OUT data packet is received with CRC errors
 - When the isochronous OUT token received by the core is corrupted
 - When the application is very slow in reading the data from the receive FIFO
- When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the GINTSTS.incomplete Isochronous OUT data interrupt, indicating that a DOEPINTx.XferCompl interrupt is not asserted on at least one of the isochronous OUT endpoints. At this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remains in progress on this endpoint on the USB.

Application Programming Sequence

- Asserting the GINTSTS.incomplete Isochronous OUT data interrupt indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.
- If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must empty all isochronous OUT data (data and status) from the receive FIFO before proceeding.
 - When all data is emptied from the receive FIFO, the application can detect the DOEPINTx.XferCompl interrupt. In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame, as described in **“Control Read Transfers (SETUP, Data IN, Status OUT)” on Page 16-128**.
- When it receives a GINTSTS.incomplete Isochronous OUT data interrupt, the application must read the control registers of all isochronous OUT endpoints (DOEPCTLx) to determine which endpoints had an incomplete transfer in the current frame. An endpoint transfer is incomplete if both the following conditions are met.
 - DOEPCTLx.Even/Odd frame bit = DSTS.SOFFN[0]
 - DOEPCTLx.Endpoint Enable = 1
- The previous step must be performed before the GINTSTS.SOF interrupt is detected, to ensure that the current frame number is not changed.
- For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the DOEPCTLx.Endpoint Disable bit.
- Wait for the DOEPINTx.Endpoint Disabled interrupt and enable the endpoint to receive new data in the next frame as explained in **“Control Read Transfers (SETUP, Data IN, Status OUT)” on Page 16-128**.

Because the core can take some time to disable the endpoint, the application possibly is not able to receive the data in the next frame after receiving bad isochronous data.

16.10.6 Periodic IN (Interrupt and Isochronous) Data Transfers

This section describes a typical periodic IN data transfer.

Application Requirements

1. Application requirements 1, 2, 3, and 4 of “**Non-Periodic (Bulk and Control) IN Data Transfers**” on Page 16-132 also apply to periodic IN data transfers, except for a slight modification of Requirement 2.
 - a) The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To transmit a few maximum- packet-size packets and a short packet at the end of the transfer, the following conditions must be met.
 - $\text{transfer size}[\text{epnum}] = n * \text{mps}[\text{epnum}] + \text{sp}$
(where n is an integer > 0 , and $0 < \text{sp} < \text{mps}[\text{epnum}]$)
 - If $(\text{sp} > 0)$, $\text{packet count}[\text{epnum}] = n + 1$
Otherwise, $\text{packet count}[\text{epnum}] = n$;
 - $\text{mc}[\text{epnum}] = \text{packet count}[\text{epnum}]$
 - b) The application cannot transmit a zero-length data packet at the end of transfer. It can transmit a single zero-length data packet by it self. To transmit a single zero-length data packet,
 - c) $\text{transfer size}[\text{epnum}] = 0$
 - $\text{packet count}[\text{epnum}] = 1$
 - $\text{mc}[\text{epnum}] = \text{packet count}[\text{epnum}]$
2. The application can only schedule data transfers 1 frame at a time.
 - a) $(\text{DIEPTSIZx.MC} - 1) * \text{DIEPCTLx.MPS} < \text{DIEPTSIZx.XferSiz} < \text{DIEPTSIZx.MC} * \text{DIEPCTLx.MPS}$
 - b) $\text{DIEPTSIZx.PktCnt} = \text{DIEPTSIZx.MC}$
 - c) If $\text{DIEPTSIZx.XferSiz} < \text{DIEPTSIZx.MC} * \text{DIEPCTLx.MPS}$, the last data packet of the transfer is a short packet.
3. This step is not applicable for isochronous data transfers, only for interrupt transfers. The application can schedule data transfers for multiple frames, only if multiples of max packet sizes (up to 3 packets), must be transmitted every frame. This is can be done, only when the core is operating in DMA mode. This is not a recommended mode though.
 - a) $((n * \text{DIEPTSIZx.MC}) - 1) * \text{DIEPCTLx.MPS} \leq \text{DIEPTSIZx.Transfer Size} \leq n * \text{DIEPTSIZx.MC} * \text{DIEPCTLx.MPS}$
 - b) $\text{DIEPTSIZx.Packet Count} = n * \text{DIEPTSIZx.MC}$
 - c) n is the number of frames for which the data transfers are scheduled

Universal Serial Bus (USB)

Data Transmitted per frame in this case would be $DIEPTSIZE \cdot MC \cdot DIEPCTL \cdot MPS$, in all the frames except the last one. In the frame "n", the data transmitted would be $(DIEPTSIZE \cdot TransferSize - (n-1) \cdot DIEPTSIZE \cdot MC \cdot DIEPCTL \cdot MPS)$

4. For Periodic IN endpoints, the data must always be prefetched 1 frame ahead for transmission in the next frame. This can be done, by enabling the Periodic IN endpoint 1 frame ahead of the frame in which the data transfer is scheduled.
5. The complete data to be transmitted in the frame must be written into the transmit FIFO (either by the application or the DMA), before the Periodic IN token is received. Even when 1 DWORD of the data to be transmitted per frame is missing in the transmit FIFO when the Periodic IN token is received, the core behaves as when the FIFO was empty. When the transmit FIFO is empty, a zero data length packet would be transmitted on the USB for ISO IN endpoints. A NAK handshake is transmitted on the USB for INTR IN endpoints.
6. For a High Bandwidth IN endpoint with three packets in a frame, the application can program the endpoint FIFO size to be $2 \cdot \text{max_pkt_size}$ and have the third packet load in after the first packet has been transmitted on the USB.

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The core fetches the data for the endpoint from memory, according to the application setting.
3. Every time either the core's internal DMA writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data is fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for an periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO mode) for the frame is not present in the FIFO, then the core generates an IN Tkn Rcvd When Tx F Empty Interrupt for the endpoint.
 - a) A zero-length data packet is transmitted on the USB for isochronous IN endpoints
 - b) A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. The packet count for the endpoint is decremented by 1 under the following conditions:
 - a) For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
 - b) For interrupt endpoints, when an ACK handshake is transmitted
 - c) When the transfer size and packet count are both 0, the Transfer Completed interrupt for the endpoint is generated and the endpoint enable is cleared.
6. At the "Periodic frame Interval" (controlled by $DCFG.PerFrint$), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates a $GINTSTS.incomplISOIN$ interrupt.

Application Programming Sequence (Transfer Per Frame)

1. Program the DIEPTSIZE and DIEPDMA registers.
2. Program the DIEPCTL register with the endpoint characteristics and set the CNAK and Endpoint Enable bits.
3. Asserting the DIEPINTx.In Token Rcvd When TxF Empty interrupt indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
 - a) If the isochronous endpoint is already enabled when this interrupt is detected, see **“Incomplete Isochronous IN Data Transfers” on Page 16-116** for more details.
5. The core handles timeouts internally, without application intervention. The application, thus, never detects a DIEPINTn.TimeOUT interrupt for periodic interrupt IN endpoints.
6. Asserting the DIEPINTx.XferCompl interrupt with no DIEPINTx.In Tkn Rcvd When TxF Empty interrupt indicates the successful completion of an isochronous IN transfer. A read to the DIEPTSIZE register must indicate transfer size = 0 and packet count = 0, indicating all data is transmitted on the USB.
7. Asserting the DIEPINTx.XferCompl interrupt, with or without the DIEPINTx.In Tkn Rcvd When TxF Empty interrupt, indicates the successful completion of an interrupt IN transfer. A read to the DIEPTSIZE register must indicate transfer size = 0 and packet count = 0, indicating all data is transmitted on the USB.
8. Asserting the GINTSTS.incomplete Isochronous IN Transfer interrupt with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

For isochronous IN endpoints, see **“Incomplete Isochronous IN Data Transfers” on Page 16-116**, for more details.

16.10.7 Periodic IN Data Transfers Using the Periodic Transfer Interrupt

This section describes a typical Periodic IN (ISOC / INTR) data transfer with the Periodic Transfer Interrupt feature.

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer is part of a single buffer, and must program the size of that buffer and its start address (in DMA mode) to the endpoint-specific registers.
2. For IN transfers, the Transfer Size field in the Endpoint Transfer Size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
 - a) To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
- Transfer size[epnum] = n * mps[epnum] + sp

Universal Serial Bus (USB)

(where n is an integer > 0 , and $0 < sp < mps[epnum]$. A higher value of n reduces the periodicity of the DOEPINTx.XferCompl interrupt)

- If ($sp > 0$), then $packet\ count[epnum] = n + 1$. Otherwise, $packet\ count[epnum] = n$

b) To transmit a single zero-length data packet:

- $Transfer\ size[epnum] = 0$

- $Packet\ count[epnum] = 1$

c) To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer in two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.

- First transfer: $transfer\ size[epnum] = n * mps[epnum]$; $packet\ count = n$;

- Second transfer: $transfer\ size[epnum] = 0$; $packet\ count = 1$;

d) The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met.

- $transfer\ size[epnum] = n * mps[epnum] + sp$ (where n is an integer > 0 , and $0 < sp < mps[epnum]$)

- If ($sp > 0$), $packet\ count[epnum] = n + 1$ Otherwise, $packet\ count[epnum] = n$;

- $mc[epnum] =$ number of packets to be sent out in a frame.

e) The application cannot transmit a zero-length data packet at the end of transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet,

- $transfer\ size[epnum] = 0$

- $packet\ count[epnum] = 1$

- $mc[epnum] = packet\ count[epnum]$

3. In DMA mode, the core fetches an IN data packet from the memory, always starting at a DWORD boundary. If the maximum packet size of the IN endpoint is not a multiple of 4, the application must arrange the data in the memory with pads inserted at the end of a maximum-packet-size packet so that a new packet always starts on a DWORD boundary.

4. Once an endpoint is enabled for data transfers, the core updates the Transfer Size register. At the end of IN transfer, which ended with an Endpoint Disabled interrupt, the application must read the Transfer Size register to determine how much data posted in the transmit FIFO was already sent on the USB.

a) $Data\ fetched\ into\ transmit\ FIFO = Application-programmed\ initial\ transfer\ size - core-updated\ final\ transfer\ size$

b) $Data\ transmitted\ on\ USB = (application-programmed\ initial\ packet\ count - Core\ updated\ final\ packet\ count) * mps[epnum]$

c) $Data\ yet\ to\ be\ transmitted\ on\ USB = (Application-programmed\ initial\ transfer\ size - data\ transmitted\ on\ USB)$

Universal Serial Bus (USB)

5. The application can schedule data transfers for multiple frames, only if multiples of max packet sizes (up to 3 packets), must be transmitted every frame. This is can be done, only when the core is operating in DMA mode.
 - a) $((n * \text{DIEPTSIZE}_n.MC) - 1) * \text{DIEPCTLEN}.MPS \leq \text{DIEPTSIZE}_n.Transfer\ Size \leq n * \text{DIEPTSIZE}_n.MC * \text{DIEPCTLEN}.MPS$
 - b) $\text{DIEPTSIZE}_n.Packet\ Count = n * \text{DIEPTSIZE}_n.MC$
 - c) n is the number of frames for which the data transfers are scheduled. Data Transmitted per frame in this case is $\text{DIEPTSIZE}_n.MC * \text{DIEPCTLEN}.MPS$ in all frames except the last one. In frame n, the data transmitted is $(\text{DIEPTSIZE}_n.TransferSize - (n - 1) * \text{DIEPTSIZE}_n.MC * \text{DIEPCTLEN}.MPS)$
6. For Periodic IN endpoints, the data must always be prefetched 1 frame ahead for transmission in the next frame. This can be done, by enabling the Periodic IN endpoint 1 frame ahead of the frame in which the data transfer is scheduled.
7. The complete data to be transmitted in the frame must be written into the transmit FIFO, before the Periodic IN token is received. Even when 1 DWORD of the data to be transmitted per frame is missing in the transmit FIFO when the Periodic IN token is received, the core behaves as when the FIFO was empty. When the transmit FIFO is empty,
 - a) A zero data length packet would be transmitted on the USB for ISOC IN endpoints
 - b) A NAK handshake would be transmitted on the USB for INTR IN endpoints
 - c) $\text{DIEPTSIZE}_x.PktCnt$ is not decremented in this case.

For a High Bandwidth IN endpoint with three packets in a frame, the application can program the endpoint FIFO size to be $2 * \text{max_pkt_size}$ and have the third packet load in after the first packet has been transmitted on the USB.

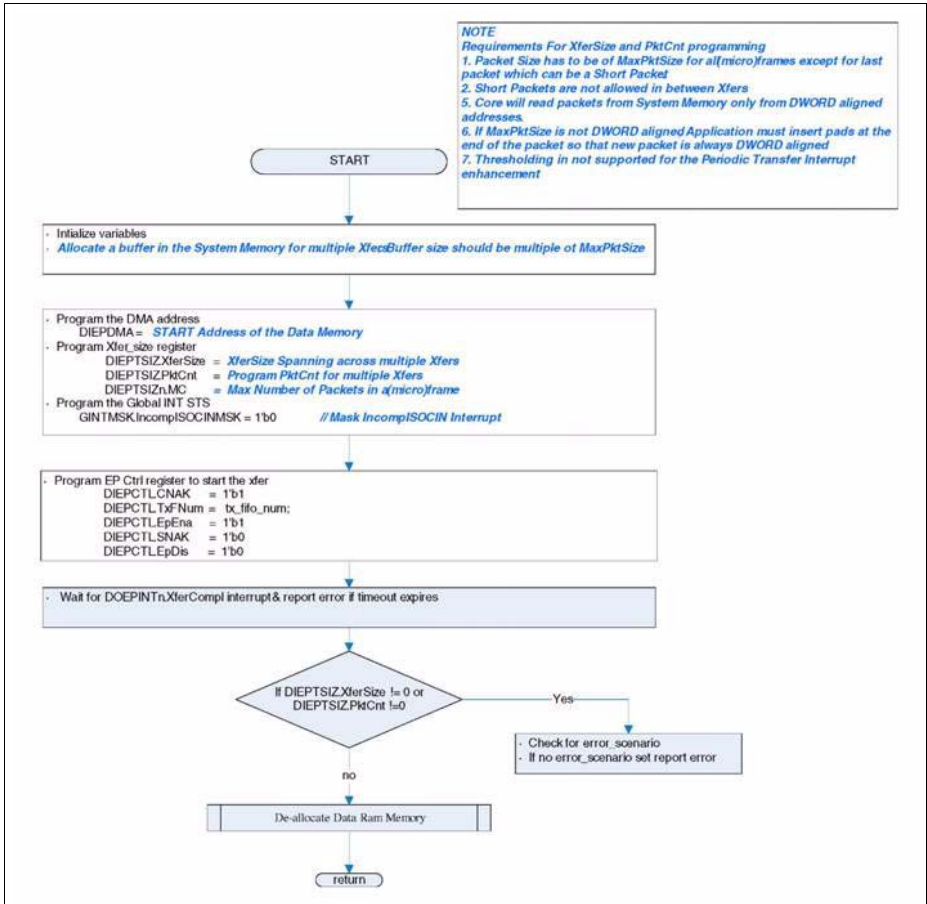


Figure 16-34 Periodic IN Application Flow for Periodic Transfer Interrupt Feature

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
 - a) The application must enable the DCTL.IgnrFrmNum
2. When an isochronous OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame will be ignored by the core. Subsequently the core updates the Even / Odd bit on its own.
3. Every time either the core's internal DMA writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data is fetched from DMA or application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete packet for the frame is not present in the FIFO, then the core generates an IN Tkn Rcvd When Tx Fifo Empty Interrupt for the endpoint.
 - a) A zero-length data packet is transmitted on the USB for isochronous IN endpoints
 - b) A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. If an IN token comes for an endpoint on the bus, and if the corresponding Tx FIFO for that endpoint has at least 1 packet available, and if the DIEPCTLx.NAK bit is not set, and if the internally maintained even/odd bit match with the bit 0 of the current frame number, then the core will send this data out on the USB. The core will also decrement the packet count. Core also toggles the MultCount in DIEPCTLx register and based on the value of MultCount the next PID value is sent.
 - a) If the IN token results in a timeout (core did not receive the handshake or handshake error), core rewind the FIFO pointers. Core does not decrement packet count. It does not toggle PID. DIEPINTx.TimeOut interrupt will be set which the application could check.
 - b) At the end of periodic frame interval (Based on the value programmed in the DCFG.PerFrint register, core will internally set the even/ odd internal bit to match the next frame.
6. The packet count for the endpoint is decremented by 1 under the following conditions:
 - a) For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
 - b) For interrupt endpoints, when an ACK handshake is transmitted
7. The data PID of the transmitted data packet is based on the value of DIEPTSIZx.MC programmed by the application. In case the DIEPTSIZx.MC value is set to 3 then, for a particular frame the core expects to receive 3 Isochronous IN token for the respective endpoint. The data PIDs transmitted will be D2 followed by D1 and D0 respectively for the tokens.
 - a) If any of the tokens responded with a zero-length packet due to non-availability of data in the Tx FIFO, the packet is sent in the next frame with the pending data PID. For example, in a frame, the first received token is responded to with data and data

Universal Serial Bus (USB)

PID value D2. If the second token is responded to with a zero-length packet, the host is expected not to send any more tokens for the respective endpoint in the current frame. When a token arrives in the next frame it will be responded to with the pending data PID value of D1.

- b) Similarly the second token of the current frame gets responded with D0 PID. The host is expected to send only two tokens for this frame as the first token got responded with D1 PID.
- 8. When the transfer size and packet count are both 0, the Transfer Completed interrupt for the endpoint is generated and the endpoint enable is cleared.
- 9. The GINTSTS.incomplISOIN will be masked by the application hence at the Periodic Frame interval (controlled by DCFG.PerFrint), even though the core finds non-empty any of the isochronous IN endpoint FIFOs, GINTSTS.incomplISOIN interrupt will not be generated.

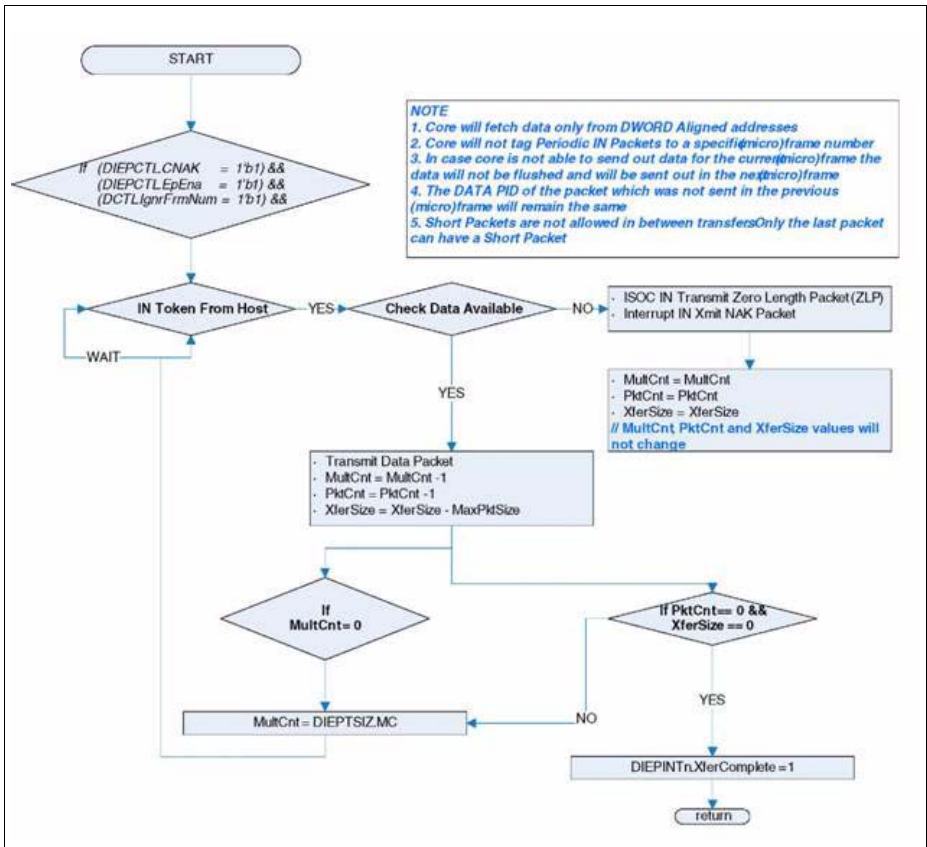


Figure 16-35 Periodic IN Core Internal Flow for Periodic Transfer Interrupt Feature

16.10.8 Interrupt OUT Data Transfers Using Periodic Transfer Interrupt

This section describes a regular INTR OUT data transfer with the Periodic Transfer Interrupt feature.

Application Requirements

1. Before setting up a periodic OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer, then program that buffer's size and start address in the endpoint-specific registers.
2. For Interrupt OUT transfers, the Transfer Size field in the endpoint's Transfer Size register must be a multiple of the maximum packet size of the endpoint, adjusted to

Universal Serial Bus (USB)

the DWORD boundary. The Transfer Size programmed can span across multiple frames based on the periodicity after which the application want to receive the DOEPINTx.XferCompl interrupt

- a) $\text{transfer size}[\text{epnum}] = n * (\text{mps}[\text{epnum}] + 4 - (\text{mps}[\text{epnum}] \bmod 4))$
 - b) $\text{packet count}[\text{epnum}] = n$
 - c) $n > 0$ (Higher value of n reduces the periodicity of the DOEPINTx.XferCompl interrupt)
 - d) $1 < \text{packet count}[\text{epnum}] < n$ (Higher value of n reduces the periodicity of the DOEPINTx.XferCompl interrupt)
3. In DMA mode, the core stores a received data packet in the memory, always starting on a DWORD boundary. If the maximum packet size of the endpoint is not a multiple of 4, the core inserts byte pads at end of a maximum-packet-size packet up to the end of the DWORD. The application will not be informed about the (micro)frame number on which a specific packet has been received.
 4. On DOEPINTx.XferCompl interrupt, the application must read the endpoint's Transfer Size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
 - a) $\text{Payload size in memory} = \text{application-programmed initial transfer size} - \text{core updated final transfer size}$
 - b) $\text{Number of USB packets in which this payload was received} = \text{application-programmed initial packet count} - \text{core updated final packet count.}$
 - c) If for some reason, the host stops sending tokens, there are no interrupts to the application, and the application must timeout on its own.
 5. The assertion of the DOEPINTx.XferCompl interrupt marks the completion of the interrupt OUT data transfer. This interrupt does not necessarily mean that the data in memory is good.
 6. Read the DOEPTSIZE register to determine the size of the received transfer and to determine the validity of the data received in the frame.

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
 - a) The application must enable the DCTL.IgnrFrmNum
2. When an interrupt OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame will be ignored by the core.
3. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the Packet Count field for that endpoint by 1.
 - a) OUT data packets received with Bad Data CRC or any packet error are flushed from the receive FIFO automatically.

Universal Serial Bus (USB)

- b) Interrupt packets with PID errors are not passed to application. Core discards the packet, sends ACK and does not decrement packet count.
 - c) If there is no space in the receive FIFO, interrupt data packets are ignored and not written to the receive FIFO. Additionally, interrupt OUT tokens receive a NAK handshake reply.
4. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or interrupt data packets are ignored and not written to the receive FIFO, and interrupt OUT tokens receive a NAK handshake reply.
 5. After the data is written to the receive FIFO, the core's DMA engine reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
 6. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
 7. The OUT Data Transfer Completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions.
 - a) The transfer size is 0 and the packet count is 0.
 - b) The last OUT data packet written to the receive FIFO is a short packet ($0 < \text{packet size} < \text{maximum packet size}$)
 8. When either the application or the DMA pops this entry (OUT Data Transfer Completed), a Transfer Completed interrupt is generated for the endpoint and the endpoint enable is cleared.

16.11 Device Programming in Scatter-Gather DMA Mode

This chapter describes the programming requirements for the Device core operating in Scatter/Gather DMA mode. It describes how to initialize the channel and provides information on asynchronous transfers (bulk and control) and periodic transfers (isochronous and interrupt).

16.11.1 Programming Overview

When the Scatter/Gather DMA mode is enabled data buffers are presented through descriptor structures

1. The application prepares the descriptors, and sets the bit DIEPCTLx/DOEPCTLx.EPEna.
2. DMA fetches the corresponding descriptor (initially determined by DIEPDMAx/DOEPDMAx).
3. DMA internally sets the transfer size from descriptor back to DIEPTSIZx/DOEPTSIZx.
4. From this point, the current USB flow executes.
5. Once the transfer size data is moved by DMA, the DMA checks for further links in the descriptor chain.

Universal Serial Bus (USB)

6. If this is the last descriptor, the DMA sets the DIOEPINTn.XferCompl interrupt.
7. If there are further active links, the DMA continues to process them.

Note: The registers DIEPTSIZx/DOEPTSIZx must not be written by the application in Scatter/Gather DMA mode.

In Scatter/Gather DMA mode, the core implements a true scatter-gather memory distribution in which data buffers are scattered over the system memory. Each endpoint memory structure is implemented as a contiguous list of descriptors, in which each descriptor points to a data buffer of predefined size. In addition to the buffer pointer (1 DWORD), the descriptor also has a status quadlet (1 DWORD). When the list is implemented as a ring buffer, the list processor switches to the first element of the list when it encounters last bit. All endpoints (control, bulk, interrupt, and isochronous) implement these structures in memory.

Note: The descriptors are stored in continuous locations. For example descriptor 1 is stored in 0000'0000_H, descriptor 2 is stored in 0000'0008_H, descriptor 3 in 0000'0010_H and so on. The descriptors are always DWORD aligned.

16.11.2 SPRAM Requirements

For each endpoint the current descriptor pointer and descriptor status are cached to avoid additional requests to system memory. These are stored in SPRAM. In addition DIEPDMAx/DOEPDMAx registers are also implemented in SPRAM.

16.11.3 Descriptor Memory Structures

The descriptor memory structures are displayed in [Figure 16-36](#).

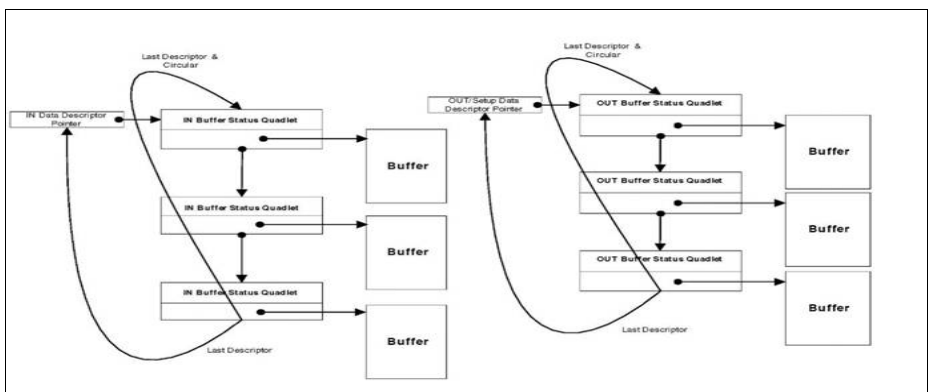


Figure 16-36 Descriptor Memory Structures

16.11.3.1 OUT Data Memory Structure

All endpoints that support OUT direction transactions (endpoints that receive data from the USB host), must implement a memory structure with the following characteristics:

- Each data buffer must have a descriptor associated with it to provide the status of the buffer. The buffer itself contains only raw data.
- Each buffer descriptor is two quadlets in length.

When the buffer status of the first descriptor is host Ready, the DMA fetches and processes its data buffer; otherwise the DMA optionally skips to the next descriptor until it reaches the end of the descriptor chain. The buffers to which the descriptor points hold packet data for non-isochronous endpoints and frame (FS)/μframe (FS) data for isochronous endpoints.

Host Ready — indicates that the descriptor is available for the DMA to process.

DMA Busy — indicates that the DMA is still processing the descriptor.

DMA Done—indicates that the buffer data transfer is complete.

Host Busy—indicates that the application is processing the descriptor.

The OUT data memory structure is shown in **Figure 16-37**, which shows the definition of status quadlet bits for non-ISO and ISO end points

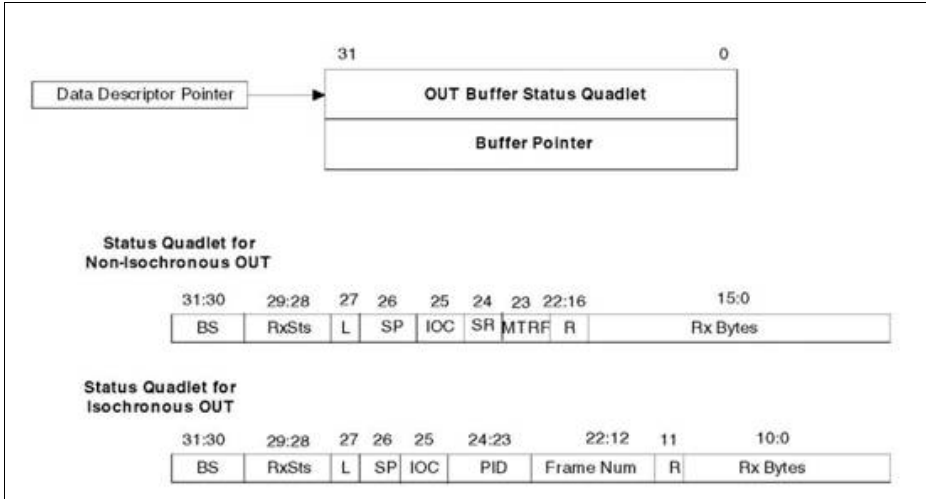


Figure 16-37 Out Data Memory Structure

The status quadlet interpretation depends on the end point type field (DOEPCTLx.EPType) for the corresponding end point. For example, if an end point is OUT and periodic, then the status quadlet is interpreted as Status Quadlet for Isochronous OUT.

Table 16-9 displays the OUT Data Memory Structure fields.

Note: Note that some fields change depending on the mode.

Table 16-9 OUT Data Memory Structure Values

Bit	Bit ID	Description
BS [31:30]	Buffer Status	<p>This 2-bit value describes data buffer status. Possible options are:</p> <ul style="list-style-type: none"> 00_B Host Ready 01_B DMA Busy 10_B DMA Done 11_B Host Busy <p>Application sets to Host Ready if the descriptor is ready or to Host Busy if the descriptor is not ready. Core sets to DMA busy if the descriptor is being serviced or to DMA Done if the transfer finished associated with the descriptor.</p> <p>The application needs to make these bits as 00_B (Host Ready) as a last step after preparing the entire descriptor ready. Once the software makes these bits as Host Ready then it must not alter the descriptor until DMA completes</p>
Rx Sts [29:28]	Receive Statu	<p>This 2-bit value describes the status of the received data. Core updates this when the descriptor is closed. This reflects whether OUT data has been received correctly or with errors. BUFERR is set by the core when AHB error is encountered during buffer access. BUFERR is set by the core after asserting AHBErr for the corresponding end point. The possible combinations are:</p> <ul style="list-style-type: none"> • 00_B Success, No AHB errors • 01_B Reserved • 10_B Reserved • 11_B BUFERR
L [27]	Last	<p>Set by the application, this bit indicates that this descriptor is the last one in the chain. Note - L Bit is interpreted by the core even when BS value is other than Host ready. For example, BNA is set, the core keeps traversing all the descriptors until it encounters a descriptor whose L bit is set after which the core disables the corresponding endpoint.</p>
SP[26]	Short Packet	<p>Set by the Core, this bit indicates that this descriptor closed after short packet. When reset it indicates that the descriptor is closed after requested amount of data is received.</p>

Table 16-9 OUT Data Memory Structure Values (cont'd)

Bit	Bit ID	Description	
IOC[25]	Interrupt On complete	Set by the application, this bit indicates that the core must generate a transfer complete interrupt(XferCompl) after this descriptor is finished.	
[24] ¹⁾	Varies	<p>Non Isochronous Out Bit: SR[24] Bit ID: Setup Packet Received Set by the Core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one setup packet per descriptor. On reception of a setup packet, the descriptor is closed and the corresponding endpoint is disabled after SETUP_COMPLETE status is seen in the Rx fifo. The core puts a SETUP_COMPLETE status into the Rx FIFO when it sees the first IN/OUT token after the SETUP packet for that particular endpoint. However, if the L bit of the descriptor is set, the endpoint is disabled and the descriptor is closed irrespective of the SETUP_COMPLETE status. The application has to re-enable for receiving any OUT data for the control transfer. (It also need to reprogram the descriptor start address) Note - Because of the above behavior, the core can receive any number of back to back setup packets and one descriptor for every setup packet is used.</p>	<p>Isochronous Out Bit: Reserved [24:23] Bit ID: This field is reserved and the core writes 00_B.</p>

Table 16-9 OUT Data Memory Structure Values (cont'd)

Bit	Bit ID	Description	
[23] ¹⁾	Varies	Non Isochronous Out Bit: MTRF[23] Bit ID: Multiple Transfer Set by the application, this bit indicates the Core can continue processing the list after it encountered last descriptor. This is to support multiple transfers without application intervention. Reserved for ISO OUT and Control OUT endpoints.	See description for bit [24]
[22:16] ¹⁾	Varies	Non Isochronous Out Bit: [22:12] Bit ID: R Reserved	Isochronous Out Bit: Frame Number [22:12] Bit ID: Frame number The 11-bit frame number corresponds to full speed frame number.

Universal Serial Bus (USB)

Table 16-9 OUT Data Memory Structure Values (cont'd)

Bit	Bit ID	Description	
[15:12] ¹⁾	Varies	Non Isochronous Out Bit: Rx Bytes [15:0]	See description for bits [22:16]
[11] ¹⁾	Varies	Bit ID: Received number of bytes remaining This 16-bit value can take values from 0 to (64K-1) bytes, depending on the transfer size of data received from the USB host. The application programs the expected transfer size. When the descriptor is done this indicates remainder of the transfer size. Here, Rx Bytes must be in terms of multiple of MPS for the corresponding end point. The MPS for the various packet types are as follows:	Isochronous Out Bit: 11 Bit ID: Reserved
[10:0] ¹⁾	Varies	<ul style="list-style-type: none"> • Control <ul style="list-style-type: none"> – LS - 8 bytes – FS - 8,16,32,64 bytes • Bulk <ul style="list-style-type: none"> – FS - 8,16,32,64 bytes • Interrupt <ul style="list-style-type: none"> – LS - up to 8 bytes – FS - up to 64 bytes <p>Note: In case of Interrupt packets, the MPS may not be a multiple of 4. If the MPS in an interrupt packet is not a multiple of 4, then a single interrupt packet corresponds to a single descriptor. If MPS is a multiple of 4 for an interrupt packet, then a single descriptor can have multiple MPS packets.</p>	<p>Isochronous Out Bit: Rx Bytes [10:0] Bit ID: Received number of bytes This 11-bit value can take values from 0 to (2K-1) bytes, depending on the packet size of data received from the USB host. Application programs the expected transfer size. When the descriptor is done this indicates remainder of the transfer size. The maximum payload size of each ISO packet as per USB specification 2.0 is as follows. FS - up to 1023 bytes Note: A Value of 0 indicates zero bytes of data, 1 indicates 1 byte of data and so on.</p>

1) The meaning of this field varies. See description.

Table 16-10 displays the matrix of L bit and MTRF bit options.

Table 16-10 OUT - L Bit and MTRF Bit

L Bit	MTRF bit	Functionality
1	1	Continue to process the list after the last descriptor encountered. Use DOEPDMAx as next descriptor. The Endpoint is not disabled.
1	0	For non-Isochronous endpoints, Stop processing list after last descriptor encountered. The application intervenes and programs the list pointer into DOEPDMAx register when a list is created in a new location otherwise enables the endpoint. Start processing when the endpoint is enabled again with DOEPDMAx register pointing to start of list. For Isochronous endpoints, the DMA engine always goes back to the base descriptor address after the last descriptor.
0	1	If a short packet is received or expected transfer is done, close the current descriptor, continue with the next descriptor. If a short packet or Zero length packet is received, the corresponding endpoint is not disabled.
0	0	After processing the current descriptor go to next descriptor. If a short packet OR zero length packet is received disable the endpoint and a transfer complete interrupt is generated irrespective of IOC bit setting for that descriptor.

Table 16-11 displays the out buffer pointer field description.

Note: For Bulk and Interrupt End Points, if MTRF bit is set for the last descriptor in a list, then all the descriptors in that list need to have their MTRF bit set.

Table 16-11 OUT Buffer Pointer

Buf Addr[31:0]	Buffer Address	The Buffer pointer field in the descriptor is 32 bits wide and contains the address where the received data is to be stored in the system memory. The starting buffer address must be DWORD aligned. The buffer size must be also DWORD aligned.
----------------	----------------	--

16.11.3.2 Isochronous OUT

- The application must create one descriptor per packet.
- End point is not disabled by the core based on L bit. The DMA always goes back to the base descriptor address after the last descriptor.
- The bit MTRF is not applicable.

16.11.3.3 Non-Isochronous OUT

- The core uses one descriptor per setup packet.
- The core closes the descriptor after receiving a short packet.
- Bit combinations for L and MTRF appear in [Table 16-10](#).
- Multiple Interrupt packets in the same buffer is allowed only if the MPS is multiple of 4.

16.11.3.4 IN Data Memory Structure

All endpoints that support IN direction transactions (transmitting data to the USB host) must implement the following memory structure. Each buffer must have a descriptor associated with it. The application fills the data buffer, updates its status in the descriptor, and enables the endpoint. The DMA fetches this descriptor and processes it, moving on in this fashion until it reaches the end of the descriptor chain. The buffer to which the descriptor points to hold packet data for non-isochronous endpoints and frame data for isochronous endpoints.

The definition of status quadlet bits for non-periodic and periodic end points are as shown in the figure. The status quadlet interpretation depends on the end point type field (DIEPCTLx.EPType) for the corresponding end point. For example, if an end point is IN and periodic, then the status quadlet is interpreted as "Status Quadlet for Isochronous IN".

The IN data memory structure is shown in **Figure 16-38**.

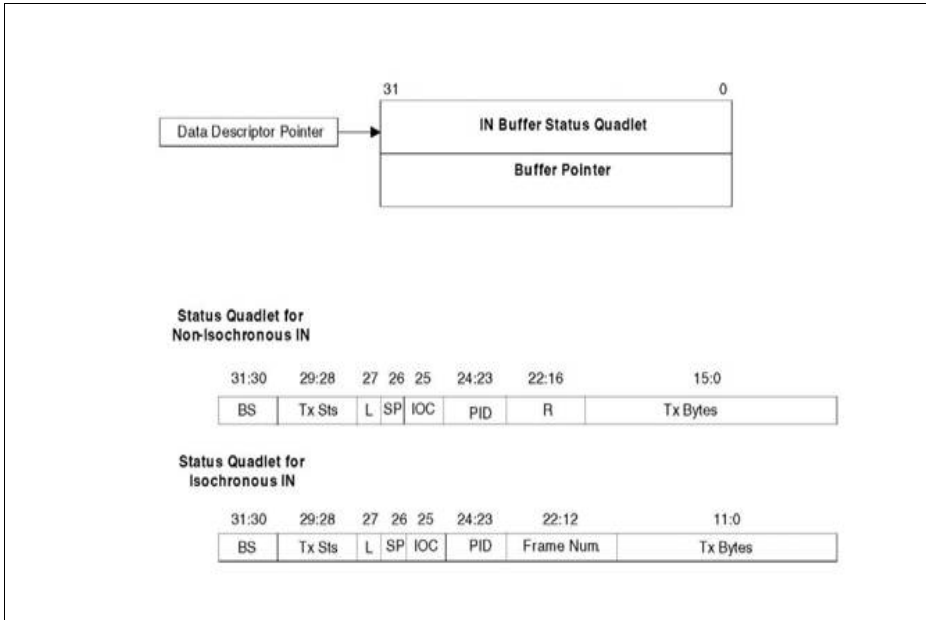


Figure 16-38 IN Data Memory Structure

Table 16-12 displays the IN Data Memory Structure fields.

Note: Some fields change depending on the mode

Table 16-12 IN Data Memory Structure Values

Bit	Bit ID	Description
BS [31:30]	Buffer Status	<p>This 2-bit value describes the status of the data buffer. The possible options are:</p> <ul style="list-style-type: none"> • 00_B Host ready • 01_B DMA busy • 10_B DMA done • 11_B Host busy <p>The application needs to make these bits as 00_B (Host Ready) as a last step after preparing the entire descriptor ready. Once the software makes these bits as HostReady then it must not alter the descriptor until DMA done</p>
Tx Sts [29:28]	Transmit Status	<p>The status of the transmitted data. This reflects if the IN data has been transmitted correctly or with errors. BUFERR is set by core when there is a AHB error during buffer access. When ilgnrFrmNum is not set, BUFFLUSH is set by the core when</p> <ul style="list-style-type: none"> • the core is fetching data pertaining to the current frame (N) and finds that the frame has incremented (N+1) during the data fetch • or • when it fetches a descriptor for which the frame number has already elapsed. The possible combinations are: <ul style="list-style-type: none"> • 00_B Success, No AHB errors • 01_B BUFFLUSH • 10_B Reserved • 11_B BUFERR
L [27]	Last	When set by the application, this bit indicates that this descriptor is the last one in the chain.
SP[26]	Short Packet	When set, this bit indicates that this descriptor points to a short packet or a zero length packet. If there is more than one packet in the descriptor, it indicates that the last packet is a short packet or a zero length packet.
IOC[25]	Interrupt On complete	When set by the application, this bit indicates that the core must generate a transfer complete interrupt after this descriptor is finished.

Universal Serial Bus (USB)

Table 16-12 IN Data Memory Structure Values (cont'd)

Bit	Bit ID	Description	
[24:23] ¹⁾	Varies	Non Isochronous In Bit: Reserved[24:16] Bit ID: Reserved	Isochronous In Bit: Reserved[24:23] Bit ID: Reserved
[22:12] ¹⁾	Varies		Isochronous In Bit: Frame Number [22:12] Bit ID: This field must correspond to the 11-bit full speed frame number.
[15:12] ¹⁾	Varies	Non Isochronous In Bit: Tx bytes [15:0] Bit ID: Number of bytes to be transmitted This 16-bit value can take values from 0 to (64K-1) bytes, indicating the number of bytes of data to be transmitted to the USB host. Note: A Value of 0 indicates zero bytes of data, 1 indicates 1 byte of data and so on.	
[11:0] ¹⁾	Varies		Isochronous In Bit: Tx bytes [11:0] Bit ID: Number of bytes to transmit Tx bytes [11:0] Number of bytes to be transmitted This 12-bit value can take values from 0 to (4K-1) bytes, indicating the number of bytes of data to be transmitted to the USB host. Note: A Value of 0 indicates zero bytes of data, 1 indicates 1 byte of data and so on.

1) The meaning of this field varies. See description.

Table 16-13 displays the matrix of IN - L Bit, SP Bit and Tx bytes options.

Table 16-13 IN - L Bit, SP Bit and Tx bytes

L Bit	SP bit	Tx Bytes	Functionality
0	1	Multiple of endpoint maximum packet size	Transmit a zero length packet after the last packet
0	1	Not multiple of maximum packet size	Send short packet at the end after normal packets are sent out. Then move onto next descriptor
0	1	0	Transmit zero length packet. Then move on to next descriptor.
0	0	Multiple of endpoint maximum packet size	Send normal packets and then move to next descriptor.
0	0	Not a multiple of maximum packet size	Transmit the normal packets and concatenate the remaining bytes with next buffer from the next descriptor. This combination is valid only for bulk end points.
0	0	0	Invalid. The behavior of the core is undefined.
1	1	Multiple of endpoint maximum packet size	Transmit a zero length packet after the last packet If this IN descriptor is for a ISO endpoint, then move onto the first descriptor in the list. If this IN descriptor is for a non-ISO endpoint, then stop processing this list and disable the corresponding end point.
1	1	Not multiple of maximum packet size	Send short packet after sending the normal packets If this IN descriptor is for a ISO endpoint, move onto the first descriptor in the list. If this IN descriptor is for a non-ISO endpoint, then stop processing this list and disable the corresponding end point.

Table 16-13 IN - L Bit, SP Bit and Tx bytes (cont'd)

L Bit	SP bit	Tx Bytes	Functionality
1	1	0	Transmit zero length packet If this IN descriptor is for a ISO endpoint, move onto the first descriptor in the list. If this IN descriptor is for a non-ISO endpoint, then stop processing this list and disable the corresponding end point.
1	0	Multiple of endpoint maximum packet size	Send normal packets If this IN descriptor is for a ISO endpoint, Move onto the first descriptor in the list after current transfer done. If this IN descriptor is for a non-ISO endpoint, then stop processing the list and disable the corresponding end point.
1	0	Not multiple of maximum packet size.	Invalid. The behavior of the core is undefined for these values.
1	0	0	invalid. The behavior of the core is undefined for these values.

The descriptions provided for the different combinations in [Table 16-13](#) depend on the previous descriptor L, SP, and Tx Bytes values. Consider [Table 16-14](#). The MPS for this example is 512.

Table 16-14 IN - Buffer Pointer

DESC NO	L bit	SP bit	Txbytes	Description
1	0	0	520	Send a normal packet of size 512, and concatenate the remaining 8 bytes with the next descriptor's buffer data
2	0	1	512	For this combination of L,SP and TxBytes, as per the above table, we need to send a zero length packet instead of a short packet. However, a normal packet followed by a short packet of length 8-bytes is sent. This is to illustrate the context dependency based on previous descriptor L,SP and TxByte combinations.

Table 16-15 displays the IN buffer pointer field description.

Table 16-15 IN Buffer Pointer

Bit	Bit ID	Description
Buf Addr[31:0]	Buffer Address	The Buffer pointer field in the descriptor is 32 bits wide and contains the address where the transmit data is stored in the system memory. The address can be non- DWORD aligned.

16.11.3.5 Descriptor Update Interrupt Enable Modes

If IOC bit is set for a descriptor and if the corresponding Transfer Completed Interrupt Mask (XferComplMask) is unmasked, this interrupt (DIOEPINTn.XferCompl) is asserted while closing that descriptor.

16.11.3.6 DMA Arbitration in Scatter/Gather DMA Mode

The arbiter grants receive higher priority than transmit. Within transmit, the priority is as follows.

- The highest priority is given to periodic endpoints. The periodic endpoints are serviced in a round robin fashion.
- The non periodic endpoints are serviced after the periodic scheduling interval has elapsed. The duration of the periodic scheduling interval is programmable, as specified by register bits DCFG[25:24]. When the periodic interval is active, the periodic endpoints are given priority.
- Amongst the periodic endpoints, the priority is round robin.
- Amongst the non periodic endpoints, the Global Multi Count field in the Device Control Register (DCTL) specifies the number of packets that need to be serviced for that end point before moving to the next endpoint.

The arbiter disables an endpoint and moves on to the next endpoint in the following scenarios as well, for all the endpoint types:

- Descriptor Fetch and AHB Error occurs.
- Buffer Not Available (BNA), such as when buffer status is Host busy.
- AHB Error during Descriptor update stage and Data transfer stage.

16.11.3.7 Buffer Data Access on AHB in Scatter/Gather DMA Mode

The buffer address whose data needs to be accessed in the system memory can be non DWORD aligned for transmit.

For buffer data read, the core arranges the buffer data to form a quadlet internally before populating the TXFIFO within the core as per the following scenarios

Universal Serial Bus (USB)

- The packet starts in a non DWORD aligned address, the core does two reads on AHB before appending the relevant bytes to form a quadlet internally. Hence the core stores the bytes before pushing to the TXFIFO.
- The packet ends in a non DWORD aligned address and it is not the end of the buffer or expected transfer, the core may switch to service another end point and come back to service the initial end point. In this case, the core reads the same DWORD location again and then samples only the relevant bytes. This eliminates the storage of the bytes for the initial end point.

For buffer data write, the core always performs DWORD accesses.

16.11.4 Control Transfer Handling

Control transfers (3-Stage Control R/WR or 2-Stage), can be handled effectively in the Descriptor-Based Scatter/Gather DMA mode by following the procedure explained in this section. By following this procedure the application is able to handle all normal control transfer flow and any of the following abnormal cases.

- More than one SETUP packet (back to back) — Host could send any number of SETUP packets back to back, before sending any IN/OUT token. In this case, the application is suppose to take the last SETUP packet, and ignore the others.
- More OUT/IN tokens during data phase than what is specified in the wlength field — If the host sends more OUT/IN data tokens than what is specified in the wlength field of the SETUP data, then the device must STALL.
- Premature SETUP packet during data/status phase — Device application must be able to handle this SETUP packet and ignore the previous control transfer.
- Lost ACK for the last data packet of a Three-Stage Control Read Status Stage.

16.11.5 Interrupt Usage for Control Transfers

The application checks the following OUT interrupts status bits for the proper decoding of control transfers.

- DIEPINTx.XferCompl (Transfer complete, based on IOC bit in the descriptor)
- DIEPINTx.InTknTxfEmp (In token received when Tx FIFO is empty)
- DOEPINTx.XferCompl (Transfer complete, based on IOC bit in the descriptor)
- DOEPINTx.SetUp (Setup Complete interrupt, generated when the core receives IN/OUT token after a SETUP packet.
- DOEPINTx.StsPhseRcvd (Status phase received interrupt (Also called SI), generated when host has switched to status phase of a Control Write transfer).

The core performs some optimization of these interrupt settings, when it sees multiple interrupt bits need to be set for OUT endpoints. This reduces the number of valid combinations of interrupts and simplifies the application.

Universal Serial Bus (USB)

The core gives priority for DOEPINTx.XferCompl over DOEPINTx.Setup and DOEPINTx.StsPhseRcvd (SI) interrupts. When setting the XferCompl interrupts, it clears the SetUP and SI interrupt bits.

- The core gives priority to DOEPINTx.SI interrupt over DOEPINTx.Setup. When setting DOEPINTx.StsPhseRcvd (SI), the core clears DOEPINTx.Setup interrupt bit.

Based on this, the application needs only to decode the combinations of interrupts for OUT endpoints shown in [Table 16-16](#).

Table 16-16 Combinations of OUT Endpoint Interrupts for Control Transfer

StsPhse Rcvd (SI)	Setup (SPD)	XferCompl (IOC)	Description	Template Used
0	0	1	Core has updated the OUT descriptor. Check the "SR" (Setup Received) bit in the descriptor to see if the data is for a SETUP or OUT transaction.	Case A
0	1	0	Setup Phase Done Interrupt for the previously decoded SETUP packet.	Case B
0	1	1	The core has updated the OUT descriptor for a SETUP packet, and the core is indicating a SETUP complete status also.	Case C
1	0	0	Host has switched to Status phase of a Control OUT transfer	Case D
1	0	1	Core has updated the OUT descriptor. Check the SR" (Setup Received) bit in the descriptor to see if the data is for a SETUP or OUT transaction. Also, the host has already switched to Control Write Status phase.	Case E

16.11.6 Application Programming Sequence

This section describes the application programming sequence to take care of normal and abnormal Control transfer scenarios.

All the control transfer cases can be handled by five separate descriptor lists. The descriptor lists are shown in [Figure 16-39](#).

- Three lists are for SETUP. The SETUP descriptors also take data for the Status stage of Control Read.
- The first two (index 0 and 1) act in a ping-pong fashion.

Universal Serial Bus (USB)

- The third list is an empty list, linked to one of the OUT descriptors when premature SETUP comes during the data/ status phase.
- Two lists are for IN and OUT data respectively.
- **Figure 16-39** displays setup_index 0, 1, and 2 as elements of array of pointers called setup_index. The first two elements of this array point to SETUP descriptors. The third element of this array is initially a NULL pointer, but is eventually linked to a SETUP descriptor. These array elements could also point to a descriptor for Control Read Status phase.

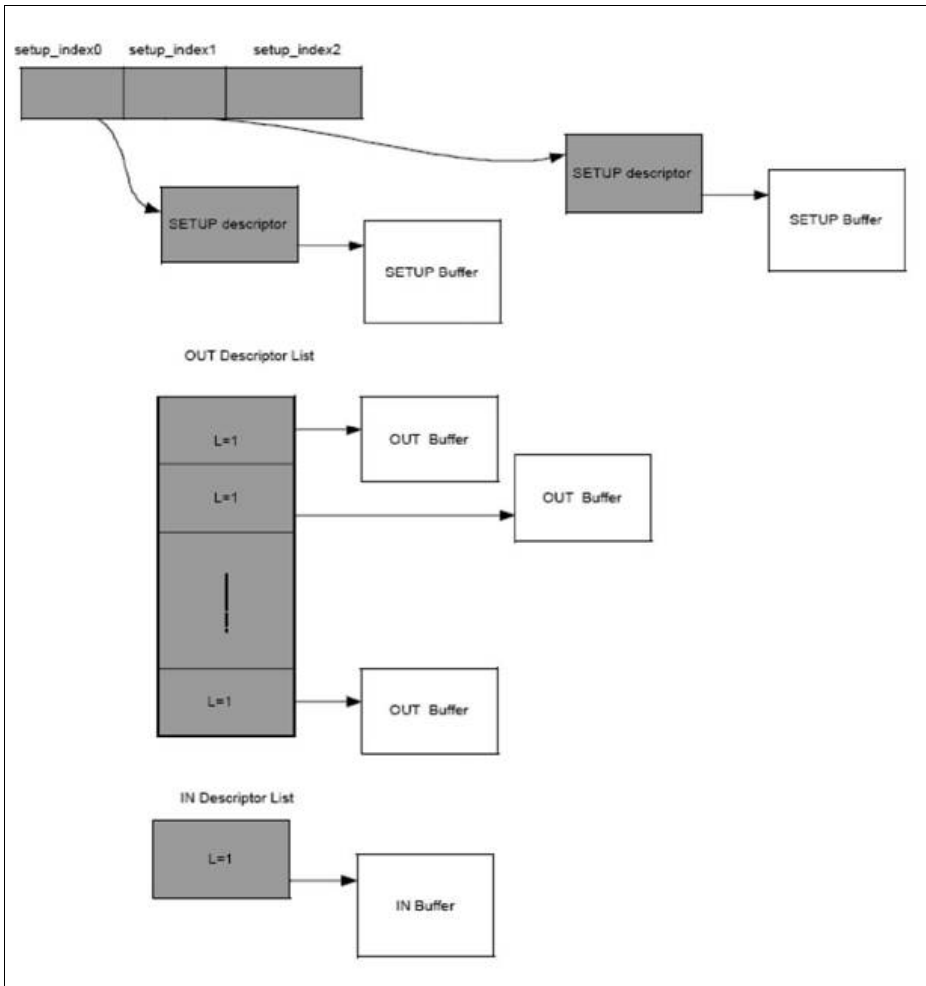


Figure 16-39 Descriptor Lists for Handling Control Transfers

The following are the steps that need to be followed by the application driver.

1. **Set up Desc for SETUP/Ctrl-Rd-Sts** — Setup 2 descriptor lists in memory for taking in SETUP packets. Each of this list must have only one descriptor, with the descriptor fields set to the following
 - a) `Rx_bytes` — Set it to Max packet size of the control endpoint.
 - b) `IOC` =1.
 - c) `MTRF`=0.

- d) L=1.
2. **Enable DMA**—If current `setup_index = 0`, then `setup_index = 1`. The application pings between these two descriptors. Program the address of the current setup descriptor (specified by `setup_index`) to `DOEPDMAx`. Write to `DOEPCTLx` with the following fields.
 - a) `DOEPCTL.MPS` — Max Packet size of the endpoint
 - b) `DOEPCTL.EPEna` — Set to 1 to enable the DMA for the endpoint.
 3. **Wait for Interrupt**—Wait for OUT endpoint interrupt (`GINTSTS.OEPInt`). Then read the corresponding `DOEPINT`.
 4. If Control Read Data Stage in progress
 - a) Case A—Check SR bit (In this case SR bit is set, because the host cannot send OUT at this point. If it sends OUT it is NAKed. GOTO Step 24.
 - b) Case B —GOTO Step 26.
 - c) Case C:-Check SR bit (In this case SR bit is set because host cannot send OUT packets without SETUP at this stage). GOTO Step 24.
 - d) Case D — Cannot happen at this stage because SI cannot come alone without a SETUP, at this stage.
 - e) Case E — Indicates that host has switched to another SETUP (Three-Stage control write) and then has switched to status phase without and data phase (core clears SUP with SI in this case). Decode SETUP packet and if ok, GOTO Step 11.

else If Ctrl Write Status Stage in progress OR Two-Stage Status Stage in progress

 - f) Case A—Check SR bit (In this case SR bit is set, because the host cannot send OUT at this point. If it sends OUT it is NAKed.) GOTO Step 24.
 - g) Case B — (Could happen for Two-Stage Ctrl Transfer.) GOTO Step 26.
 - h) Case C—GOTO Step 24.
 - i) Case D — Clear SI interrupt and wait Step 3.
 - j) Case E — Cannot happen at this stage.

else

 - k) Case A—GOTO Check Desc.
 - l) Case B — Normally, this does not occur at this stage. Either IOC comes first or IOC comes with SUP (Case C).
 - m)Case C— GOTO Check Desc.
 - n) Case D — Cannot happen at this point.
 - o) Case E — If `SR==1`, Indicates Three stage control Transfer SETUP and that the host has switched to status phase. Decode the SETUP packet and Goto Step 11.
 - p) Check Desc

Read the Descriptor status quadlet corresponding to the `setup_index` and check the SR field. (Application might also want to check the BS and RxSts fields and

Universal Serial Bus (USB)

- take necessary actions if there is any abnormalities). If SR field is 1 GOTO Step 5 (If Step Step 20 is active, terminate it). If SR field is 0 GOTO Step 22 (Control Rd Status phase) (This must also terminate Step 20).
5. Decode SETUP—Decode the SETUP packet. If it is a Three-Stage Control Write, GOTO Step 20. If it is a Three-Stage Control Read, GOTO Step 15. If it is a Two-Stage Control transfer, GOTO Step 11 (Same as Status stage for 3-Stage Control Write).
 6. Desc list for Ctrl Wr data— Setup descriptor list for Control write data phase. This must be based on the Wlength field in the SETUP data. The descriptors in the list must be setup such that there must be one descriptor per packet. Each of these descriptors must have the control fields set as follows.
 - a) Rx_Bytes — Set to the Max Packet Size of the control Endpoint.
 - b) IOC = 1
 - c) MTRF = 0.
 - d) L=1.
 - e) At this point we are not enabling and clearing the NAK for the IN endpoint for status phase. This is because, the status phase for Control Write can be ACKed only after decoding the complete data for the data phase.
GOTO Step 7.
 7. Enable DMA for Ctrl Wr Data—Write the start address of this list to DOEPDMAx. Program the DOEPCTLx with the following bits set
 - a) DOEPCTL.MPS — Max Packet size of the endpoint
 - b) DOEPCTL.EPEna — Set to 1 to enable the DMA for the endpoint. GOTO Step 8.
 8. Wait for Ctrl Wr Data Interrupt—Wait for OUT endpoint interrupt (GINTSTS.OEPInt). Then read the corresponding DOEPINTx.
 - a) Case A—check the SR field. Also clear DOEPINTx.XferCompl by writing to DOEPINTx.(Application might also want to check the BS and RxSts fields and take necessary actions if there is any abnormalities). If SR field is 0 GOTO Step 9.If SR field is 1, GOTO Step 23. (This indicates that the host has switched to a new control transfer).
 - b) Case B —GOTO Step 25.
 - c) Case C—GOTO Step 23. (This indicates that the host has switched to a new control transfer).
 - d) Case D — Host has switched to status phase. Decode the data received so far. GOTO Step 10.
 - e) Case E — Check SR bit. If SR==0, decode the data received so far. GOTO Step 10. If SR==1, decode the SETUP packet and Goto Step10.
 9. Check Desc — If it's not the last packet of data phase, Re-enable the endpoint and clear the Nak. This is because the core sets NAK after receiving each OUT packet for control write data phase. This is to allow application to STALL in case the host sends more data than what is specified in the Wlength field. GOTO Step 8. Re-enabling and clearing the NK involves the following steps.
 - a) Write to DOEPDMA with the new descriptor address.

- b) Write to DOEPCTLx with the following fields.
 - DOEPCTL.MPS — Max Packet size of the endpoint
 - DOEPCTL.CNAK—Set to 1 to clear the NAK.
 - DOEPCTL.EPEna — Set to 1 to enable the DMA for the endpoint.
 If it is the last packet of the data phase, GOTO Step 10.
- 10. **STALL Extra Bytes**— Write to DOEPCTLx with Stall set so that the core could STALL any further OUT tokens from host.If the received Bytes so far is greater than what is specified in Wlength field OR is there were any unsupported commands in the data phase, then write to DIEPCTLx with the Stall bit set so that the Status phase could be Stalled.(The STALL bit is automatically cleared by the core with the next SETUP). GOTO Step 11.
- 11. **Disc list for Ctrl Wr Sts**— The following two process must run in parallel. This is because, we are preparing for the status phase (IN) of Control write but at the same time the host could send another SETUP. So IN and OUT descriptor list must be ready.
 - a) Do Step 2— Step 5 (This is for handling SETUP or Ctrl Wr Status). If the OUT DMA is already enabled (OUT DMA was enabled for data phase of Three-Stage Control Write, but there was a premature status phase), GOTO Step 3.
 - b) Setup descriptor list for Status phase IN, depending on the data in the status phase. Normally it is always a zero length packet.
 - c) Tx_Bytes — Size of status phase,
 - d) BS — Host Ready,
 - e) L=1.
 - f) IOC=1.
 - g) SP=1 (Depending on the Tx_Bytes).
 - h) Write to DIEPDMAx with the start address of the descriptor.Write to DIEPCTLx clear the NAK and enable the endpoint. Flush the corresponding TX FIFO.
 - i) If SI has not been received in the data stage prior to the status stage, then wait for SI before clearing the NAK(DIEPCTLx.CNAK=1)
 - j) DIEPCTLx.EpEna=1.
 - k) GOTO Step 12.
- 12. **Wait for Interrupt**—Wait for IN endpoint interrupt (GINTSTS.IEPInt).
- 13. If IN endpoint INterrupt, and DIEPINTx.XferCompl, then GOTO Step 14.
- 14. **Check Desc** —Read the Status field of the descriptor. Check Tx_bytes in the descriptor.(Application might also want to check the BS and RxSts fields and take necessary actions if there is any abnormalities). This is end of Three-Stage Control Write OR Two-Stage Control transfer. We are now ready for the next control transfer (Already taken care by process "a" is Step 11.
- 15. **Desc for Ctrl Rd Data**—The following two steps must be run in parallel. This is because, we are preparing for Data phase of Control read, but at the same time, the host could abnormally abort this control transfer and send a SETUP, or switch to status phase.

Universal Serial Bus (USB)

16. Do Step 2— Step 5 (This is for handling SETUP and also Control Read Status phase.).
17. Setup descriptor list for Data phase IN, depending on the WLength field in the SETUP data. The setup can be for a single descriptor OR multiple descriptors. If it is multiple descriptors, ensure that IOC for the last descriptor is set.
 - a) Tx_Bytes — Size of data phase (Wlength field).
 - b) BS — Host Ready
 - c) L=1.
 - d) IOC=1. It is mandatory to set the IOC when it is the last descriptor.
 - e) SP=1 (Depending on the Tx_Bytes).
 - f) Write to DIEPDMAx with the start address of the descriptor list.
 - g) Write to DIEPCTLx clear the NAK and enable the endpoint.
 - h) Flush the corresponding TX FIFO.
 - i) DIEPCTLx.MPS = Max_packet size of the endpoint,
 - j) DIEPCTLx.CNAK=1 only if SPD already set (Case C in Step 3).
 - k) Also set the DOEPTLx.CNAK for the corresponding OUT endpoint after SPD because a premature status stage (OUT) can come which must be acked.
 - l) DIEPCTLx.EpEna=1.
 - m)GOTO Step 18.
18. **Wait for Interrupt**—Wait for IN endpoint interrupt (GINTSTS.IEPInt)
19. If IN endpoint interrupt, read the corresponding DIEPINTx and if XferCompl is set GOTO Step 20.
20. **Check_Desc**—Wait for the DIEPINTx.IOC interrupt. Go to Step 21.
21. **Set_Stall**—Write to DIEPCTLx with STALL bit set. (The STALL bit is automatically cleared by the core with the next SETUP). The function of this process initiated in step Step 15 is over, and must be terminated. The next control transfer is already taken care by the process that is running from Step 2.
22. **Ctrl Rd Sts Desc Check** — Read the descriptor to check the Rxbytes and also check the SP field. The Three-Stage control Read is complete here. GOTO Step 2, in preparation for the next SETUP.
23. The unexpected SETUP packet now received during the control write data phase, is sitting in the descriptor allocated for Data. Link this to the setup descriptor pointer. setup_desc_index = 2. Point setup_desc_index to the current OUT descriptor (which has the SETUP). GOTO Step 5.
24. Disable IN Endpoint DMA. Core flushes the corresponding Tx FIFO in order to flush the data that was meant for Control Write Status phase OR Control Read data phase. If Step 12 or Step 18 is active, terminate it. GOTO Step .
25. Read Modify write DOEPTLx to clear the NAK. Then GOTO Step 8 again.
 - a) DOEPTLx.CNAK—Set to 1 to clear the NAK.
26. Read Modify write DIEPCTLx to clear the NAK. Then GOTO Step 3 again.
 - a) DIEPCTLx.CNAK—Set to 1 to clear the NAK.
27. Read Modify write DIEPCTLx to clear the NAK. Then Step 12 again
 - a) DIEPCTLx.CNAK—Set to 1 to clear the NAK.

- b) DOEPCTLx.CNAK: Set to 1 to clear the NAK for the out endpoint. This clears the NAK to accept status stage data in case of control read.

16.11.7 Internal Data Flow

This section explains the cores internal data flow for control transfers.

16.11.7.1 Three-Stage Control Write

Figure 16-40 displays the core behavior for Three-Stage control write transfers.

1. On receiving SETUP, the data is pushed into the Rx FIFO and the core sets NAK on both IN and OUT endpoint of that control endpoint. Additionally, the clearing of the NAK bit is blocked by the core until the following SPD or SI is read by the application and cleared.
2. The DMA detects the Rx FIFO as non-empty and does the following:
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
3. On receiving the first data phase OUT token after the SETUP, the core push a SETUP_COMPLETE status into the Rx FIFO. Core NAKs the data phase OUT tokens because of the NAK set on receiving the SETUP packet.
4. The core generates DOEPINT.XferCompl interrupt after having transferred the SETUP packet into memory (Step 2).
5. The core generates DOEPINT.SetUp interrupt after the DMA has popped the SETUP_COMPLETE status out of the Rx FIFO.
6. Application clears NAK for the data phase, after receiving DOEPINTx.SetUp interrupt.
7. The core ACKs and the next OUT token because the NAK has been cleared (provided there is enough space in the Rx FIFO).
8. DMA detects the OUT packet in Rx FIFO and starts transferring the OUT packet to the system memory.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the OUT packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close descriptor with DMA_DONE status.
9. The core NAKs the next OUT token because the core internally sets the NAK after every control write data phase packets. This is to allow application to Stall any extra tokens.
10. The core generates DOEPINT.XferCompl after closing the OUT descriptor (Step 8).
11. Application clear NAK on receiving DOEPINTx.XferCompl interrupt.
12. Host starts the Status phase by sending the IN token which is NAKed by the core. The core push DATA_PHASE_DONE status into the Rx FIFO.

Universal Serial Bus (USB)

13. The core generates DOEPINTx.XferCompl for the last OUT packet transfer to system memory.
14. The core generates DOEPINT.StsPhsRcvd interrupt after the DMA has popped the DATA_PHASE_DONE status from the RxFIFO.
15. Application clears the NAK and enables the IN endpoint for status phase.
16. The core starts fetching the data for the Status phase
 - a) Fetch the descriptor pointed by DIEPDMA.
 - b) Fetch the packet (if size >0) to Tx fifo.
 - c) Close the descriptor with DMA_DONE status
 - d) The core generates DIEPINTx.XferCompl interrupt after closing the descriptor.
17. The core sends out data in response to the Status Phase IN token.

Universal Serial Bus (USB)

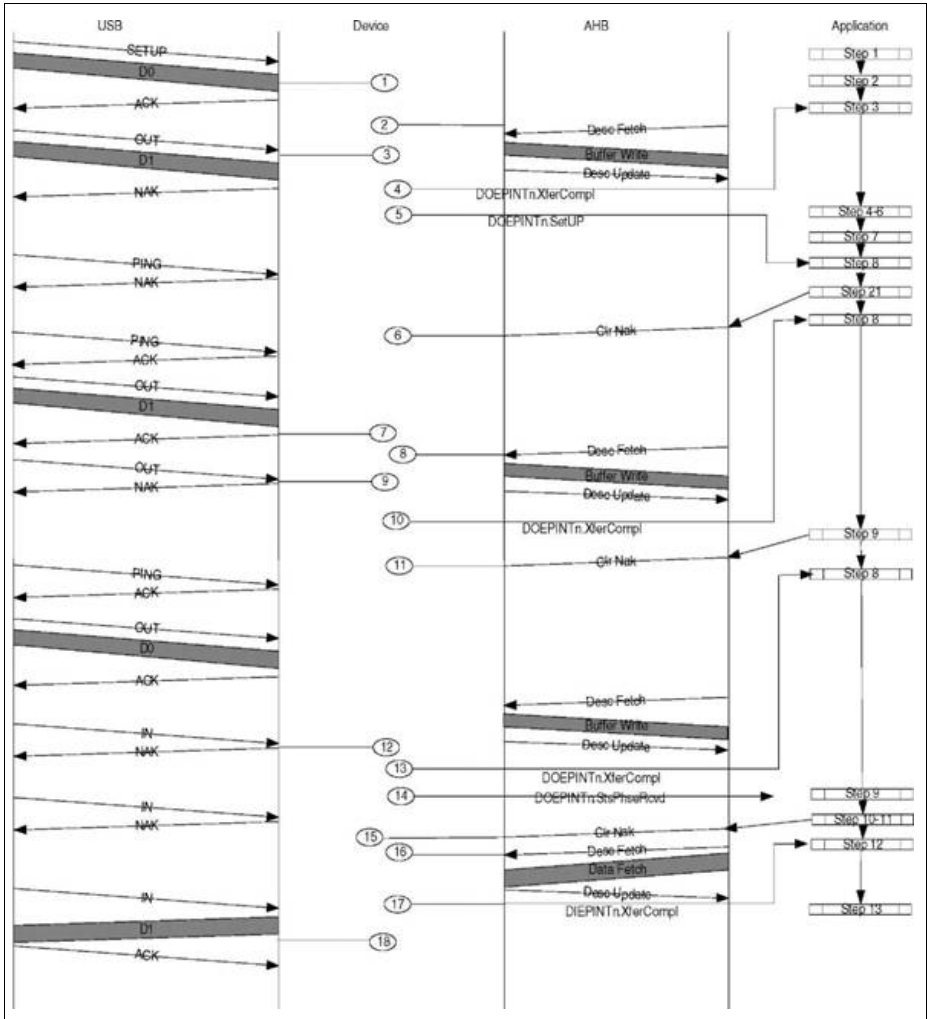


Figure 16-40 Three-Stage Control Write

16.11.7.2 Three-Stage Control Read

Figure 16-41 displays the core flow for three-stage control read transfers

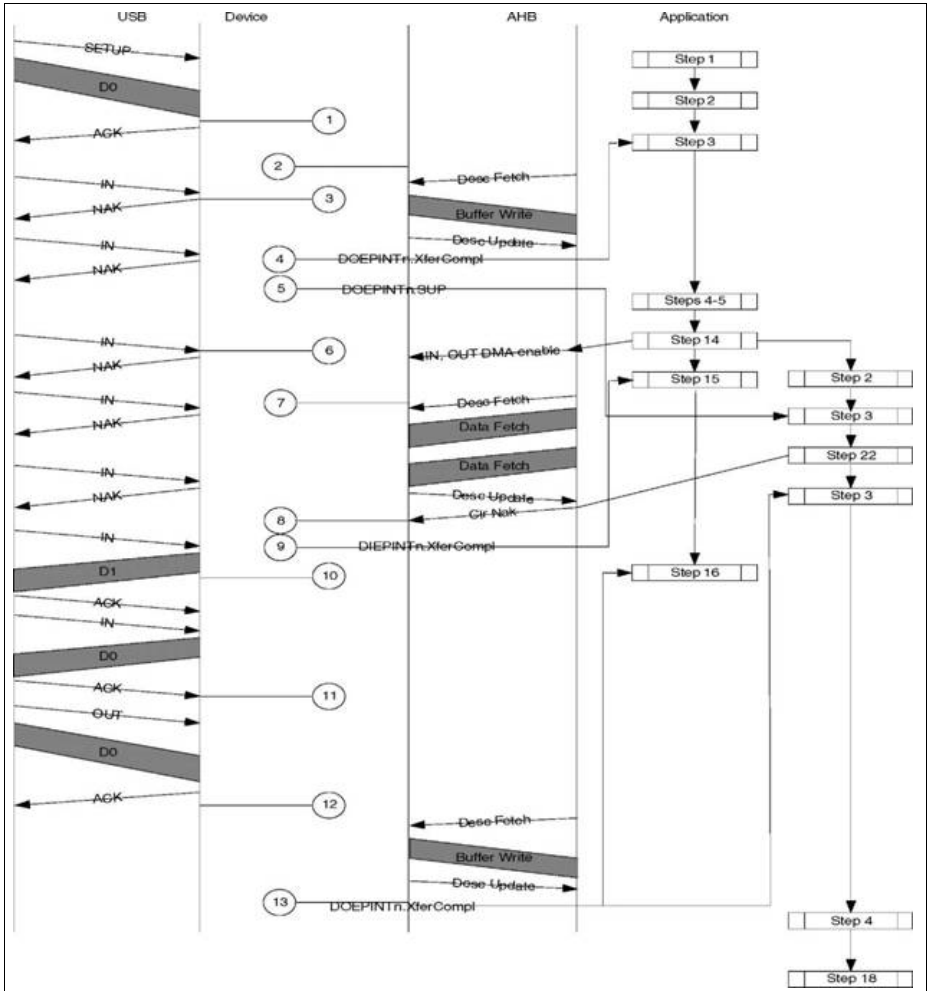


Figure 16-41 Three-Stage Control Read

In this example, it is assumed that the data phase consists of 2 packets, and the application allocates these two packets in a single buffer.

1. On receiving SETUP, the data is pushed into the Rx FIFO and the core sets NAK on both IN and OUT endpoint of that control endpoint.

Universal Serial Bus (USB)

2. The DMA detects the RxFIFO as non-empty and does the following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the RxFIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
3. On receiving the first data phase OUT token after the SETUP, the core push a SETUP_COMPLETE status into the RxFIFO. Core NAKs the data phase OUT tokens because of the NAK set on receiving the SETUP packet.
4. The core generates DOEPINT.XferCompl interrupt after having transferred the SETUP packet into memory (Step 2).
5. The core generates DOEPINT.SetUp interrupt after the DMA has popped the SETUP_COMPLETE status out of the RxFIFO.
6. Data phase IN tokens are NAKed until this point because the NAK has not yet been cleared by the application.
7. The core starts fetching the IN data after the application enables IN DMA (In this example it is assumed that multiple packets are in the same buffer. But it could also be in different buffers). This involves the following steps
 - a) Fetch the descriptor pointed by DIEPDMA.
 - b) Fetch the data into the corresponding Tx FIFO.
 - c) Close the descriptor with DMA_DONE status...
8. The application clears the NAK after receiving the setup complete (DOEPINT.SetUp) interrupt. The application also clears NAK of the OUT End point to accept the status phase.
9. After all the data has been fetched for the descriptor (Step 7), core generates DIEPINT.XferCompl interrupt.
10. The core sends data in response to the IN token for the data phase.
11. The core sends out the last packet of the IN data phase.
12. The core ACKs the status phase.
13. The core generates DOEPINTx.XferCompl interrupt after transferring the data received for the status phase to system memory.

16.11.7.3 Two-Stage Control Transfer

Figure 16-42 displays the core behavior for Two Stage control read transfers.

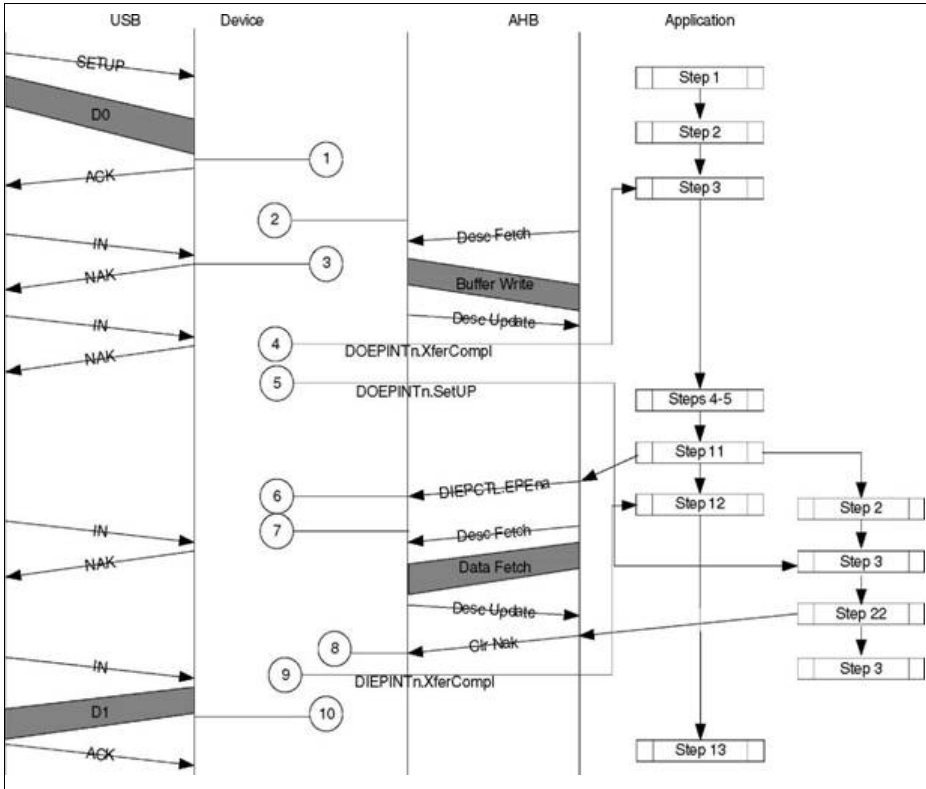


Figure 16-42 Two-Stage Control Transfer

This example shows the core behavior for a Two-Stage Control transfer.

1. On receiving SETUP, the data is pushed into the Rx FIFO and the core sets NAK on both IN and OUT endpoint of that control endpoint.
2. The DMA detects the Rx FIFO as non-empty and does the following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
 - d) The core receives the status phase IN token, which it NAKs. Core also pushes SETUP_COMPLETE status into the Rx FIFO.

Universal Serial Bus (USB)

- e) The core generates DOEPINT.XferCompl interrupt after having transferred the SETUP packet into memory (Step 2)
- f) The core generates DOEPINT.SetUp interrupt after the DMA has popped the SETUP_COMPLETE status out of the RxFIFO.
3. Application enables the IN endpoint for status phase.
4. The core starts fetching the descriptor for IN endpoint.
5. Application clears the NAK for IN endpoint after getting the DOEPINTx.SetUp interrupt (Step 4).
6. The core generates DIEPINTx.XferCompl after updating the descriptor after IN data fetch.
7. The core sends out data for the status phase IN token from host.

16.11.7.4 Back to Back SETUP During Control Write

This example shows the core receiving 2 Back to Back SETUP tokens for 3 Three-Stage Control write.

1. On receiving SETUP, the data is pushed into the Rx FIFO and the core sets NAK on both IN and OUT endpoint of that control endpoint.
2. The DMA detects the RxFIFO as non-empty and does the following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the RxFIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
 - d) The core receives another SETUP, and pushes the data into the Rx FIFO, also sets the NAK.
 - e) The core generates DOEPINT.XferCompl interrupt after having transferred the SETUP packet into memory (Step 2).
3. On receiving the first data phase OUT token after the SETUP, the core push a SETUP_COMPLETE status into the RxFIFO. Core NAKs the data phase OUT tokens because of the NAK set on receiving the SETUP packet.
4. The DMA detects the RxFIFO as non-empty (because of the 2nd SETUP packet) and does the following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the RxFIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
 - d) The core generates DOEPINT.XferCompl interrupt after having transferred the second SETUP packet into memory (Step 6)
 - e) The core generates DOEPINT.SetUp interrupt after the DMA has popped the SETUP_COMPLETE status out of the RxFIFO.
5. Application clears NAK for the data phase, after receiving DOEPINTx.SetUp interrupt.

Universal Serial Bus (USB)

6. The core ACKs the next OUT/Ping token after the NAK has been cleared by the application.
7. The DMA detects the RxFIFO as non-empty (because of the OUT packet) and does the following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the RxFIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
 - d) The core generates DOEPINT.XferCompl interrupt after having transferred the OUT packet into memory (Step 11) and closing the descriptor.

The remaining steps are similar to Steps 11-18 of **“Application Programming Sequence” on Page 16-164**.

This example shows the core behavior for a Two-Stage Control transfer.

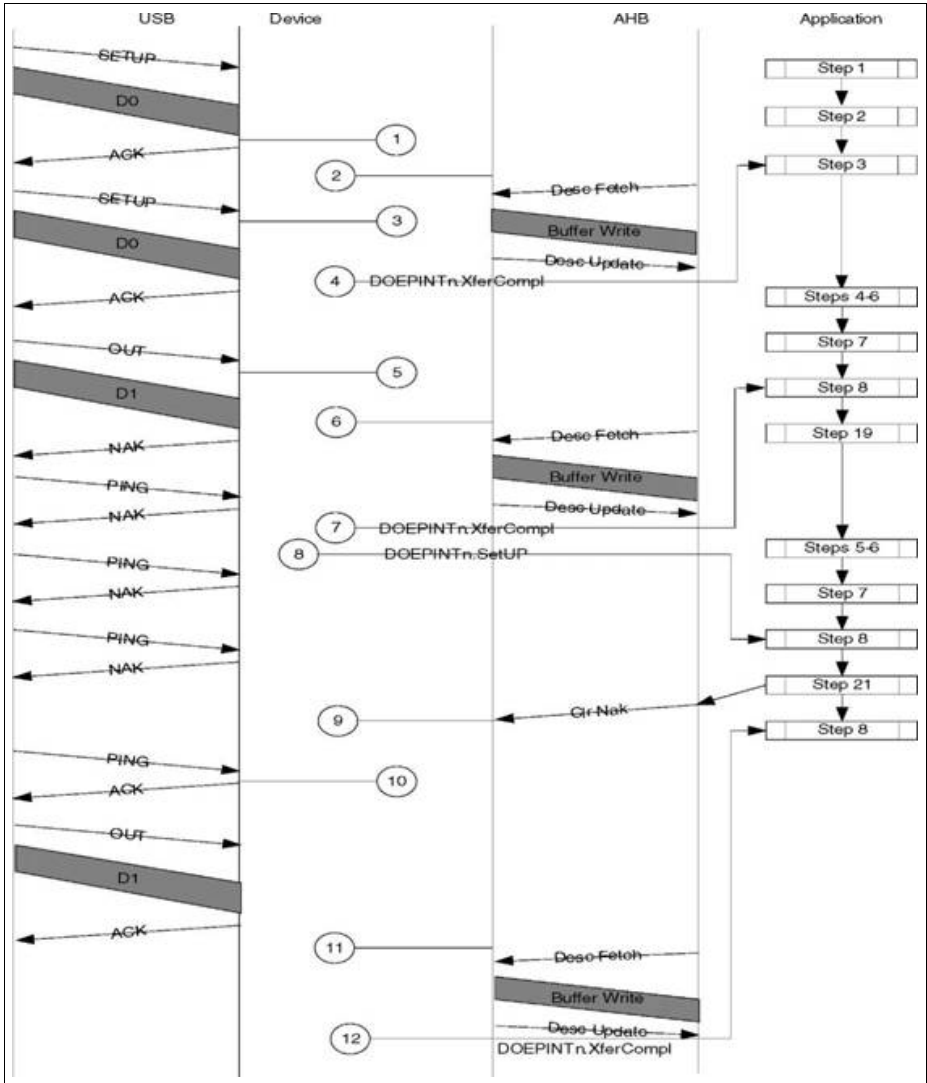


Figure 16-43 Back-to-Back SETUP Packet Handling During Control Write

16.11.7.5 Back-to-Back SETUPs During Control Read

1. On receiving SETUP, the data is pushed into the Rx FIFO and the core sets NAK on both IN and OUT endpoint of that control endpoint.
2. The DMA detects the Rx FIFO as non-empty and does the following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
 - d) The core receives another SETUP, and pushes the data into the Rx FIFO, also sets the NAK.
3. The core generates DOEPINT.XferCompl interrupt after having transferred the SETUP packet into memory (Step 2).
4. Host sends IN token for the data phase which is NAKed by the core, because NAK is set in Step3. The core pushes SETUP_COMPLETE status into Rx FIFO.
5. After the application has re-enabled the OUT DMA (Application flow Step 2) core detects Rx FIFO as non-empty because of the second SETUP packet and does the following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
 - d) The core generates DOEPINTx.XferCompl interrupt after having transferred the SETUP packet into memory (Step6).
 - e) The core starts fetching data for IN endpoint because the IN endpoint was enabled by application in Step-14.
6. On seeing DOEPINTx.XferCompl (Step 7) and finding that it is a SETUP packet, application disables the endpoint in Step 20.
7. The core generates DOEPINTx.SetUP (Setup complete) interrupt after popping the SETUP_COMPLETE status from the Rx FIFO.
8. The core generates endpoint disabled interrupt (as a result of application setting disable bit in step 9)
9. The core generates DIEPINTx.XferCompl after completing the IN data fetch and updating the descriptor.
10. application clears NAK after seeing setup_complete interrupt (generated in Step 10). The flow after this is same as steps 9 - 13 of **“Internal Data Flow” on Page 16-171**

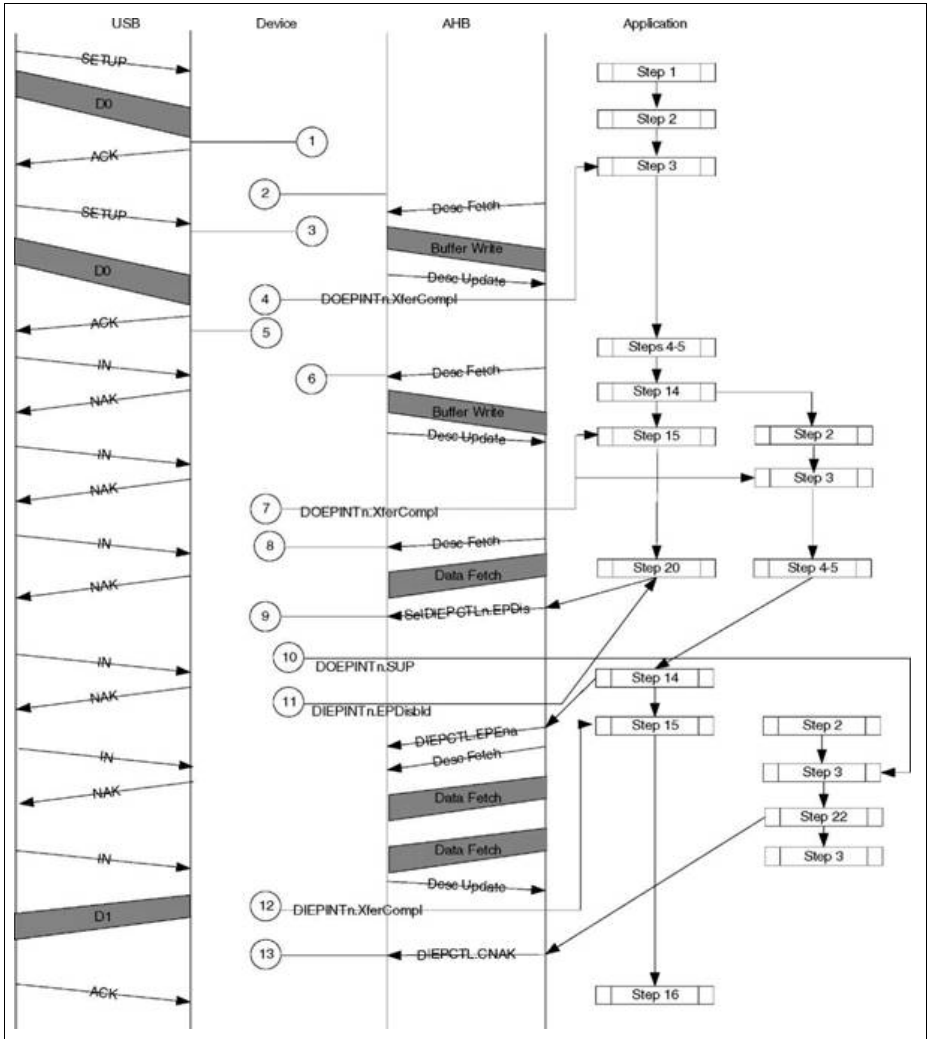


Figure 16-44 Back-to-Back SETUP During Control Read

16.11.7.6 Extra Tokens During Control Write Data Phase

This example assumes a three-stage control write transfer with only WLength field in the SETUP indicating only 1 packet in the data phase. But the host sends an additional OUT packets which the core STALLs.

1. On receiving SETUP, the data is pushed into the Rx FIFO and the core sets NAK on both IN and OUT endpoint of that control endpoint.
2. The DMA detects the Rx FIFO as non-empty and does the following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
3. On receiving the first data phase OUT token after the SETUP, the core push a SETUP_COMPLETE status into the Rx FIFO. Core NAKs the data phase OUT tokens because of the NAK set on receiving the SETUP packet.
4. The core generates DOEPINT.XferCompl interrupt after having transferred the SETUP packet into memory (Step 2).
5. The core generates DOEPINT.SetUp interrupt after the DMA has popped the SETUP_COMPLETE status out of the Rx FIFO.
6. Application clears NAK for the data phase, after receiving DOEPINTx.SetUp interrupt.
7. The core ACKs and the next OUT token because the NAK has been cleared (provided there is enough space in the Rx FIFO).
8. DMA starts transferring the OUT packet to the system memory.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the OUT packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close descriptor with DMA_DONE status.
 - d) The core generates DOEPINTx.XferCompl interrupt after having transferred the OUT packet to the system memory. Since there were only one packet in the data phase, the data phase is complete here.
 - e) The core initially NAK's the extra tokens send by the host, because the core internally sets NAK after each OUT packet for the data phase of control write.
9. Application sets STALL to stall any extra tokens.
10. The core stalls the next OUT/PING token.
11. Host switches to next control transfer, core ACKs the SETUP. This SETUP packet is transferred to the system memory buffer originally allocated for Status phase.
12. The core generates DOEPINTx.XferCompl interrupt after transferring the SETUP packet to the system memory.

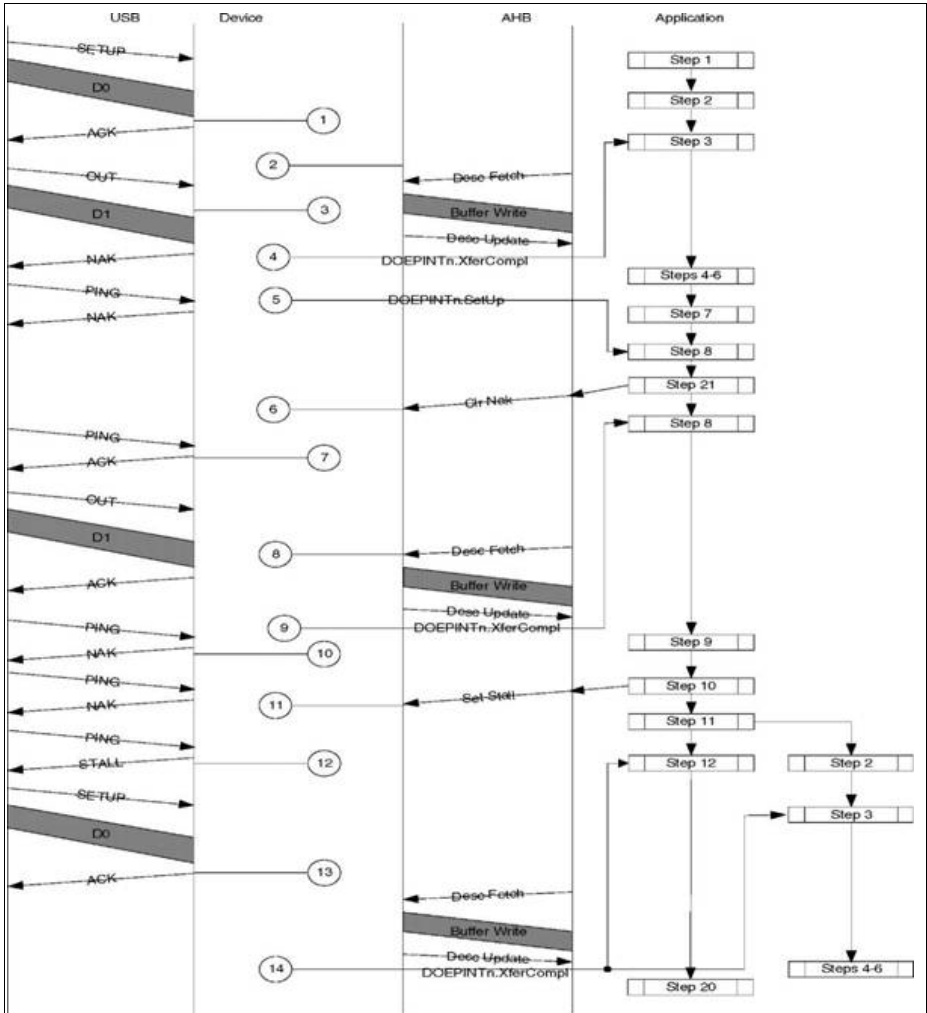


Figure 16-45 Extra Tokens During Control Write Data Phase

16.11.7.7 Extra Tokens During Control Read Data Phase

In this example, it is assumed that the data phase consists of 2 packets, and the application allocates these two packets in a single buffer. After the data phase is complete and the two packets have been transferred, the core sends an extra IN token and then the application sets Stall.

1. On receiving SETUP, the data is pushed into the Rx FIFO and the core sets NAK on both IN and OUT endpoint of that control endpoint.
2. The DMA detects the Rx FIFO as non-empty and does the following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
3. On receiving the first data phase OUT token after the SETUP, the core push a SETUP_COMPLETE status into the Rx FIFO. Core NAKs the data phase OUT tokens because of the NAK set on receiving the SETUP packet.
4. The core generates DOEPINT.XferCompl interrupt after having transferred the SETUP packet into memory (Step 2).
5. The core generates DOEPINT.SetUp interrupt after the DMA has popped the SETUP_COMPLETE status out of the Rx FIFO.
6. Data phase IN tokens are NAKed until this point because the NAK has not yet been cleared by the application.
7. The core starts fetching the IN data after the application enables IN DMA (In this example it is assumed that multiple packets are in the same buffer. But it could also be in different buffers). This involves the following steps
 - a) Fetch the descriptor pointed by DIEPDMA.
 - b) Fetch the data into the corresponding Tx FIFO.
 - c) Close the descriptor with DMA_DONE status.
8. The application clear the NAK after receiving the setup complete (DOEPINT.SetUp) interrupt. Set the Stall bit after all the Data has been pushed in the FIFO
9. After all the data has been fetched for the descriptor (Step 7), core generates DIEPINT.XferCompl interrupt.
10. The core sends data in response to the IN token for the data phase.
11. The core sends out the last packet of the IN data phase.
12. Host sends an extra token.
13. The core Stalls the IN token and also automatically Stalls the Status phase if the Host switches to the Status phase.

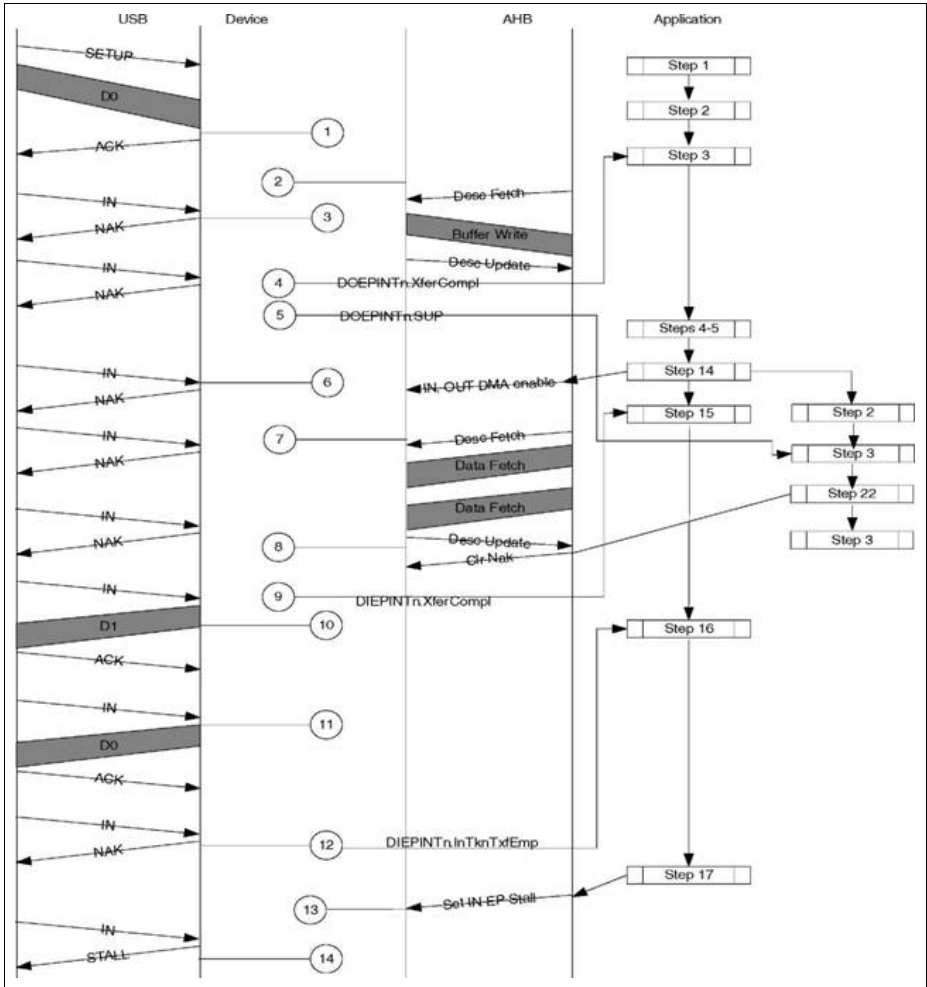


Figure 16-46 Extra IN Tokens During Control Read Data Phase

16.11.7.8 Premature SETUP During Control Write Data Phase

This example shows a Three-Stage Control Write transfer with host sending a premature Control Write SETUP packet during the data phase.

1. On receiving SETUP, the data is pushed into the Rx FIFO and the core sets NAK on both IN and OUT endpoint of that control endpoint.
2. The DMA detects the Rx FIFO as non-empty and does the following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
3. On receiving the first data phase OUT token after the SETUP, the core push a SETUP_COMPLETE status into the Rx FIFO. Core NAKs the data phase OUT tokens because of the NAK set on receiving the SETUP packet.
4. The core generates DOEPINT.XferCompl interrupt after having transferred the SETUP packet into memory (Step 2).
5. The core generates DOEPINT.SetUp interrupt after the DMA has popped the SETUP_COMPLETE status out of the Rx FIFO.
6. The core receives a SETUP packet during the data phase. This is an unexpected SETUP packet. On receiving this SETUP, the SETUP data is pushed into the Rx FIFO and the core again sets NAK on both IN and OUT endpoints of the control endpoint (NAK was already set because of the first SETUP packet received).
7. Application decodes the previous DOEPINT.SetUp interrupt and clears the NAK, unaware of the fact that there is another SETUP packet sitting in the Rx FIFO for the same control endpoint. On seeing this condition, core does not allow clearing of the NAK bit, and masks the clearing of NAK. The core takes this decision based on the fact that a SETUP_COMPLETE status is pending in the RXFIFO.
8. The DMA detects the Rx FIFO as non-empty (because of the unexpected SETUP) and does following
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
 - d) The core NAKs the data phase OUT token because NAK bit clearing by the application did not take effect (as explained in Step 7).
 - e) The core generates DOEPINT.XferCompl interrupt after having transferred the SETUP packet into memory (Step 8).
 - f) The core generates DOEPINT.SetUp interrupt after the DMA has popped the SETUP_COMPLETE status (for the unexpected SETUP packet received) out of the Rx FIFO.
9. Application clears the NAK after decoding the latest SETUP packet. This time, the core does not mask the clearing of the NAK because there are no more SETUP_COMPLETE status sitting in the Rx FIFO.

10. The core ACKs the next OUT/PING token of the data phase.
11. DMA starts transferring the OUT packet to the system memory.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the OUT packet from the RxFIFO to the buffer pointed by the descriptor.
 - c) Close descriptor with DMA_DONE status.
 - d) The core generates DOEPINTx.XferCompl interrupt after having transferred the OUT packet to the system memory.
 - e) The remaining steps are similar to Steps 11-18 of **“Application Programming Sequence” on Page 16-164**

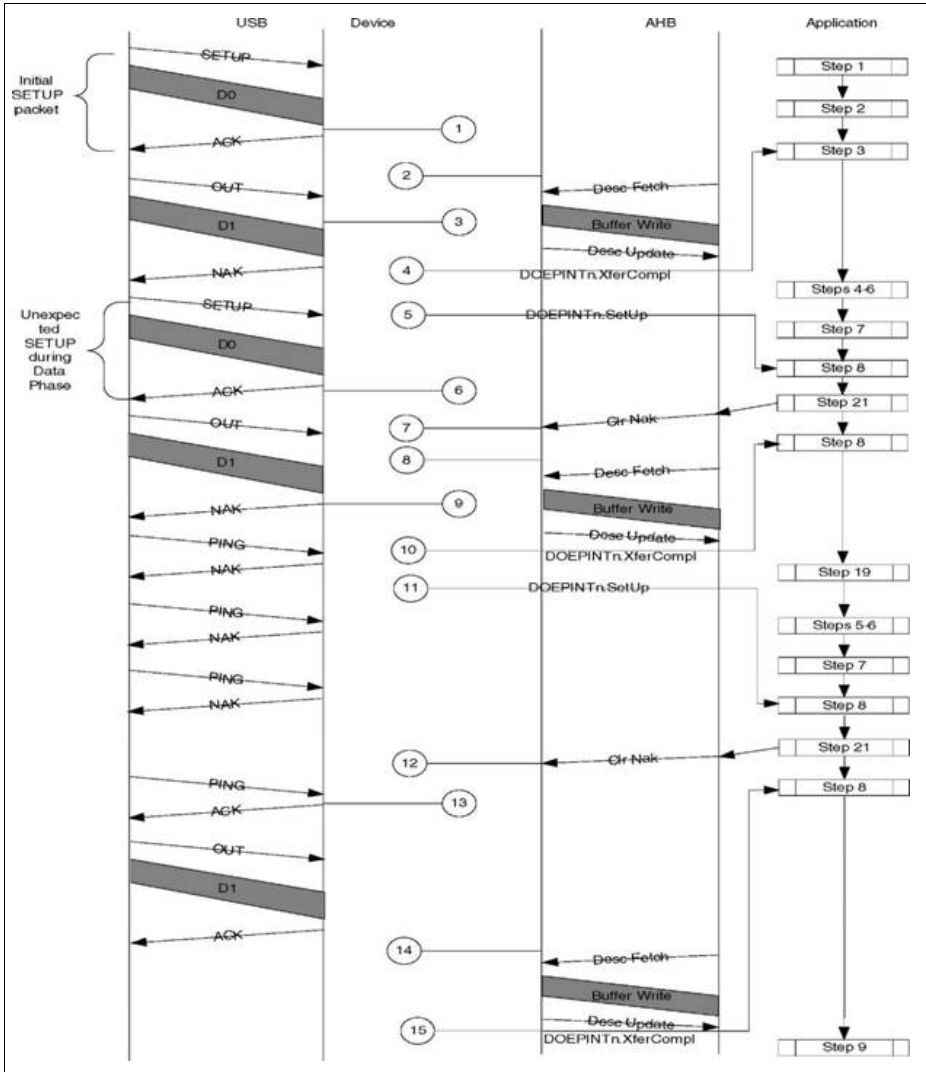


Figure 16-47 Premature SETUP During Control Write Data Phase

16.11.7.9 Premature SETUP During Control Read Data Phase

In this example, it is assumed that the data phase consists of 2 packets, and the application allocates these two packets in a single buffer. The host switches to a new control read command after having send two IN tokens during the data phase.

1. On receiving SETUP, the data is pushed into the Rx FIFO and the core sets NAK on both IN and OUT endpoint of that control endpoint.
2. The DMA detects the Rx FIFO as non-empty and does the following.
 - a) Fetch the descriptor pointed by DOEPMa.
 - b) Transfer the SETUP packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
3. On receiving the first data phase IN token after the SETUP, the core push a SETUP_COMPLETE status into the Rx FIFO. Core NAKs the data phase IN tokens because of the NAK set on receiving the SETUP packet.
4. The core generates DOEPINT.XferCompl interrupt after having transferred the SETUP packet into memory (Step 2).
5. The core generates DOEPINT.SetUp interrupt after the DMA has popped the SETUP_COMPLETE status out of the Rx FIFO.
6. Host switches to a new Control transfer by sending a SETUP token. This is the premature SETUP packet. Core sets NAK on both IN and OUT control endpoints.
7. The core fetch the data for IN control endpoint after application enables the IN endpoint.
8. The core push SETUP_COMPLETE status into Rx FIFO on seeing the IN token for data phase.
9. Application clears NAK as a result of DOEPINT.SetUP (Setup complete) interrupt generated in Step 5. But core masks this clearing of setup_complete interrupt, because there is already one SETUP packet sitting in the Rx FIFO.
10. The core generates DIEPINTx.XFERCompl after closing the IN endpoint descriptor (for Step 7)
11. The core generates DOEPINTx.XferCompl after transferring the premature SETUP packet to system memory and closing the descriptor.
12. The core generates SETUP complete interrupt.
13. Application enables IN endpoint DMA for data phase.
14. The core fetches descriptor and data for IN endpoint.
15. Application clears IN endpoint NAK after receiving DOEPINTx.SetUP (Setup complete) interrupt. This time, the core does not mask the clearing of the Nak because NO SETUP packet is remaining in the Rx FIFO.
16. The core generates DIEPINTx.XferCompl interrupt after fetching the data and closing the descriptor. The remaining steps are same as steps 11 to 13 of **“Internal Data Flow” on Page 16-171.**

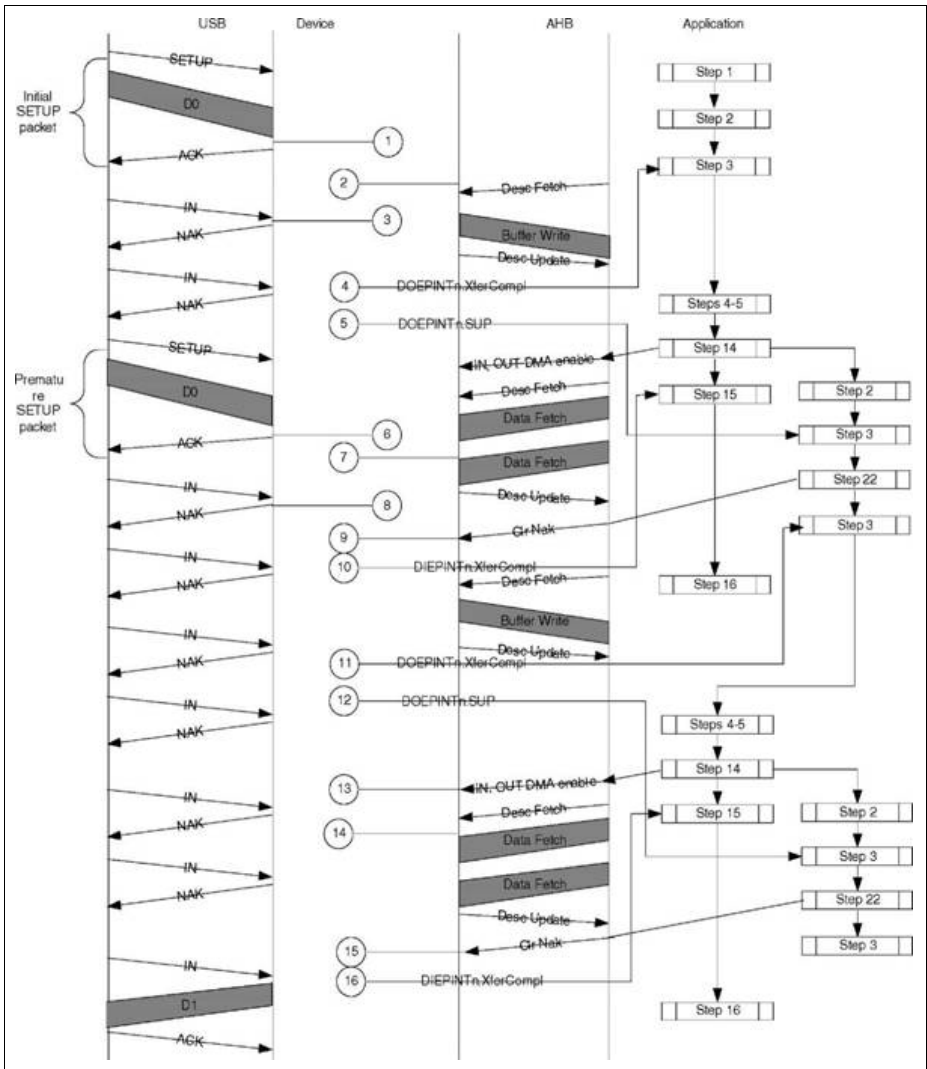


Figure 16-48 Premature SETUP During Control Read Data Phase

16.11.7.10 Premature Status During Control Write

This example assumes a Three-Stage control write transfer with only Wlength field in the SETUP indicating two packets in the data phase. But the host switch to data phase after the first packet of the data phase is complete.

1. On receiving SETUP, the data is pushed into the Rx FIFO and the core sets NAK on both IN and OUT endpoint of that control endpoint.
2. The DMA detects the Rx FIFO as non-empty and does the following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
3. On receiving the first data phase OUT token after the SETUP, the core push a SETUP_COMPLETE status into the Rx FIFO. Core NAKs the data phase OUT tokens because of the NAK set on receiving the SETUP packet.
4. The core generates DOEPINT.XferCompl interrupt after having transferred the SETUP packet into memory (Step 2).
5. The core generates DOEPINT.SetUp interrupt after the DMA has popped the SETUP_COMPLETE status out of the Rx FIFO.
6. Application clears NAK for the data phase, after receiving DOEPINTx.SetUp interrupt (Step 5).
7. The core ACKs and the next OUT token because the NAK has been cleared (provided there is enough space in the Rx FIFO).
8. DMA sees Tx FIFO non empty and starts transferring the OUT packet to the system memory.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the OUT packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close descriptor with DMA_DONE status.
9. Host switch to status phase (IN token) without completing the data phase.
10. The core generates DOEPINTx.XferComp after closing the descriptor after the data fetch.
11. Application sets up descriptor, enables IN endpoint and clear NAK.
12. The core starts to fetch the descriptor and data for the status phase once application has enabled the IN endpoint.
13. The core generates DIEPINTx.XferCompl after doing the data fetch and the descriptor update (step 12)
14. The core sends data out in response to status phase IN token.

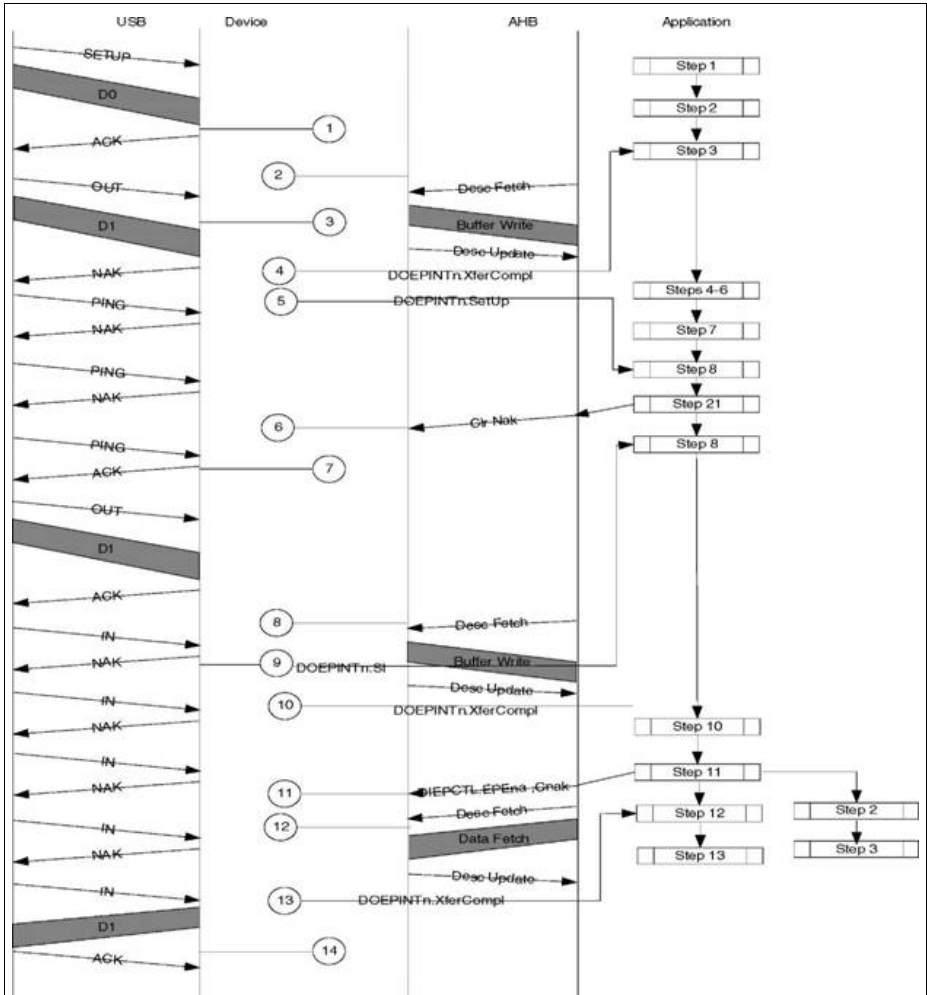


Figure 16-49 Premature Status Phase During Control Write

16.11.7.11 Premature Status During Control Read

In this example, it is assumed that the data phase consists of two packets, and the application allocates these two packets in a single buffer. After one packet in the data phase, host switches to status phase.

1. On receiving SETUP, the data is pushed into the Rx FIFO and the core sets NAK on both IN and OUT endpoint of that control endpoint.
2. The DMA detects the Rx FIFO as non-empty and does the following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
3. On receiving the first data phase IN token after the SETUP, the core push a SETUP_COMPLETE status into the Rx FIFO. Core NAKs the data phase IN tokens because of the NAK set on receiving the SETUP packet.
4. The core generates DOEPINT.XferCompl interrupt after having transferred the SETUP packet into memory (Step 2).
5. The core generates DOEPINT.SetUp interrupt after the DMA has popped the SETUP_COMPLETE status out of the Rx FIFO.
6. Data phase IN tokens are NAKed until this point because the NAK has not yet been cleared by the application.
7. The core starts fetching the IN data after the application enables IN DMA (In this example it is assumed that multiple packets are in the same buffer. But it could also be in different buffers). This involves the following steps
 - a) Fetch the descriptor pointed by DIEPDMA.
 - b) Fetch the data into the corresponding Tx FIFO.
 - c) Close the descriptor with DMA_DONE status.
8. Application clear the NAK after receiving the setup complete (DOEPINT.SetUp) interrupt.
9. After all the data has been fetched for the descriptor (Step 7), core generates DIEPINT.XferCompl interrupt.
10. The core sends data in response to the IN token for the data phase.
11. Host switches to status phase and sends the status phase OUT token. Core ACKs the OUT packet because the NAK has already been cleared.
12. The DMA detects the Rx FIFO as non-empty (because of the status phase data) and does following.
 - a) Fetch the descriptor pointed by DOEPMA.
 - b) Transfer the SETUP packet from the Rx FIFO to the buffer pointed by the descriptor.
 - c) Close the descriptor with DMA_DONE status.
 - d) The core generates DOEPINT.XferCompl after transferring the status phase data to system memory and closing the descriptor.

Universal Serial Bus (USB)

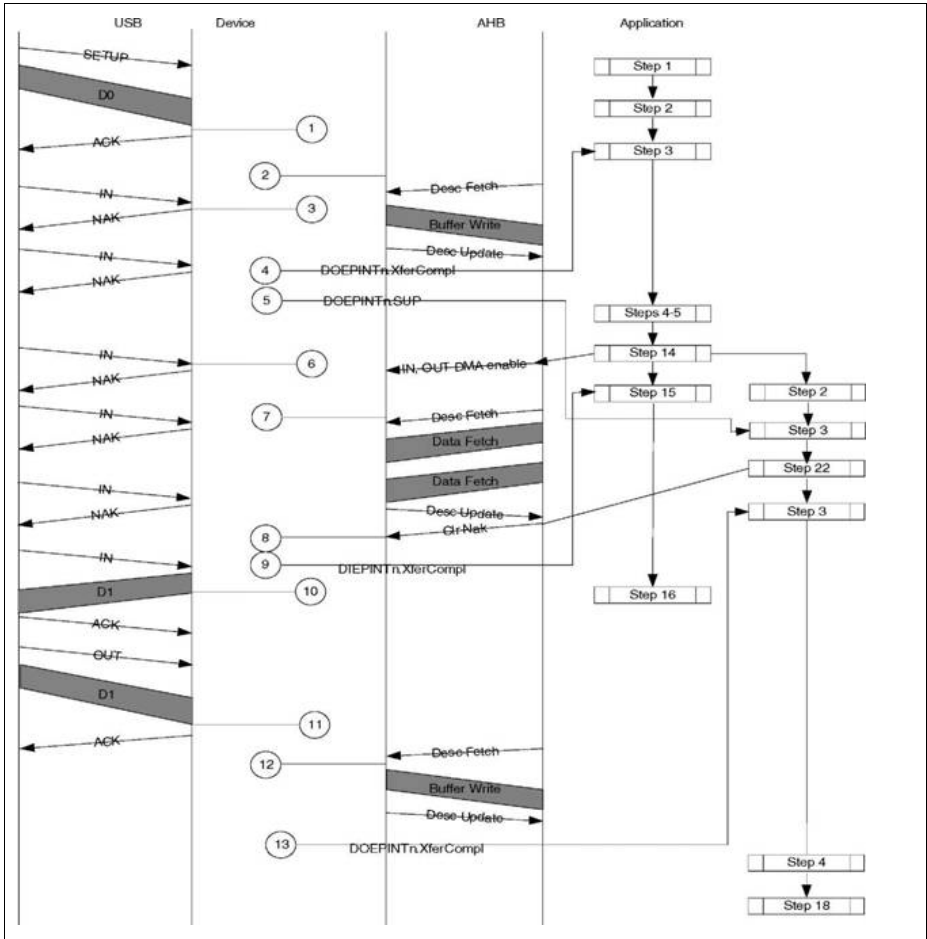


Figure 16-50 Premature Status Phase During Control Read

16.11.7.12 Lost ACK During Last Packet of Control Read

This is similar to the previous section. [Figure 16-51](#) shows this.

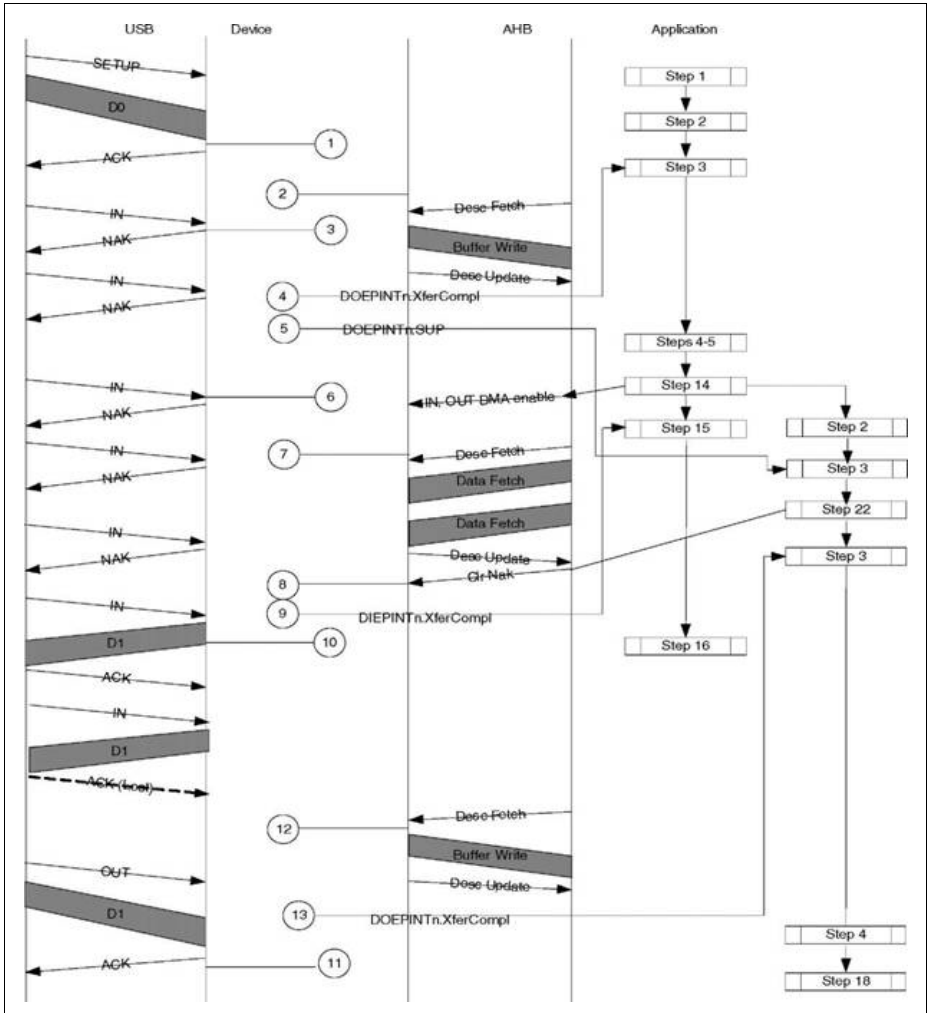


Figure 16-51 Lost ACK During Last Packet of Control Read

16.11.8 Bulk Transfer Handling in Scatter/Gather DMA Mode

16.11.8.1 Bulk IN Transfer in Scatter-Gather DMA Mode

Interrupt usage

The following interrupts are of relevance.

1. DIEPINTx.XferCompl (Transfer complete, based on IOC bit in the descriptor)
2. DIEPINTx.BNA (Buffer Not Available)

Application Programming Sequence

This section describes the application programming sequence for Bulk IN transfer scenarios.

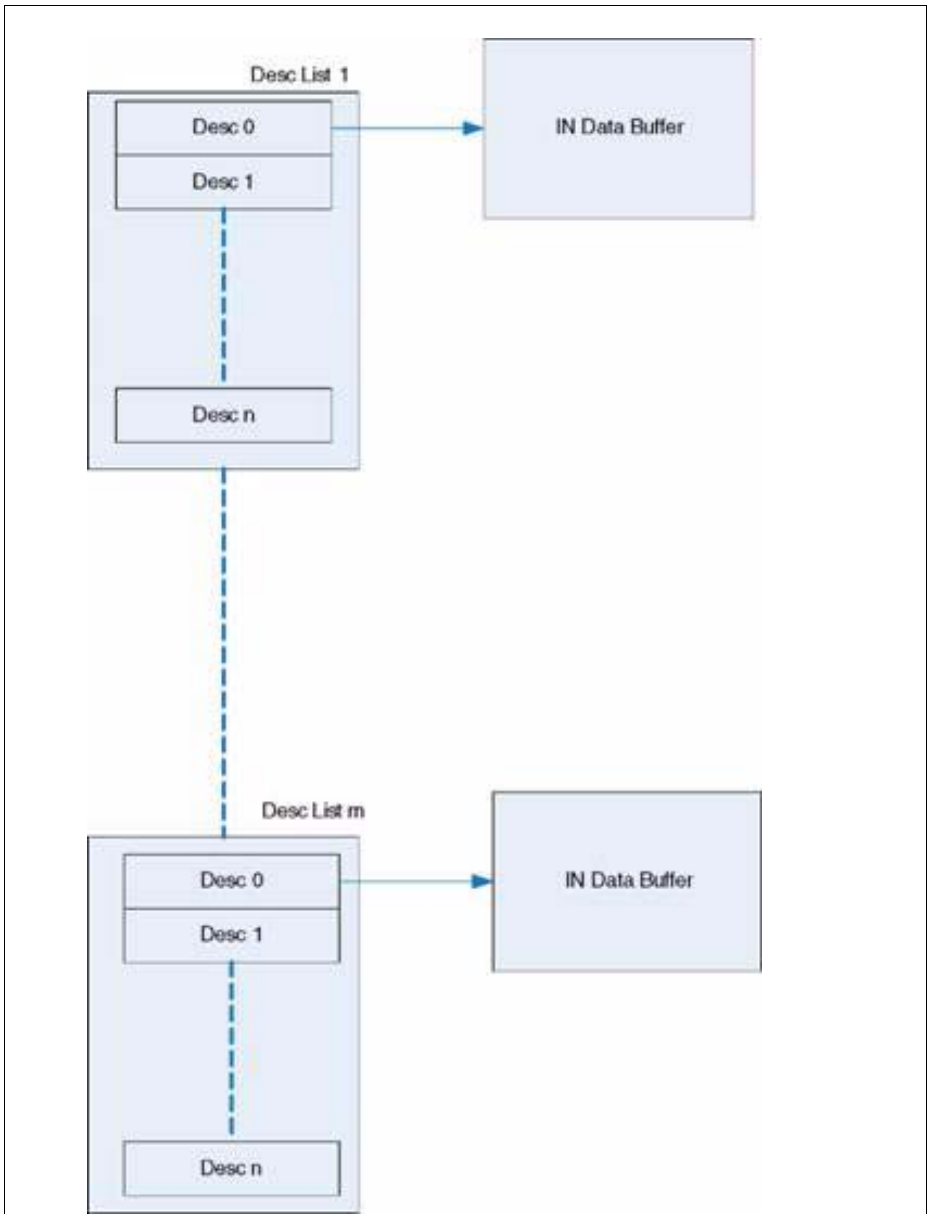


Figure 16-52 IN Descriptor List

1. Prepare Descriptor(s):
2. The application creates descriptor list(s) in the system memory pertaining to an Endpoint.
3. Each descriptor list may have up to n descriptors and there may be up to m descriptor lists.
4. Application may choose to set the IOC bit of the corresponding descriptor. If the IOC is set for the last descriptor of the list, the core generates DIEPINTx.XferCompl interrupt after the entire list is processed.
5. Program DIEPDMAx:
 - a) Application programs the base address of the descriptor in the corresponding IN Endpoint DIEPDMAx register.
6. Enable DMA:
 - a) Application programs the corresponding endpoint DIEPCTLx register with the following
 - DIEPCTLx.MPS — Max Packet size of the endpoint
 - DIEPCTLx.CNAK—Set to 1 to clear the NAK
 - DIEPCTLx.EPEna — Set to 1 to enable the DMA for the endpoint.
7. Wait for Interrupt:
 - a) On reception of DIEPINTx.XferCompl, application must check the Buffer status and Tx Status field of the descriptor to ascertain that the descriptor closed normally.

DIEPINTx.BNA interrupt gets generated by the core when it encounters a descriptor in the list whose Buffer Status field is not Host Ready. In this case, the application is suppose to read the DIEPDMAx register to ascertain the address for which the BNA interrupt is asserted to take corrective action.

Internal Flow

Bulk IN Transfers

The core handles Bulk IN transfers internally as functionally depicted in [Figure 16-53](#) (Non ISO IN Descriptor/Data Processing). [Figure 16-54](#) depicts this flow.

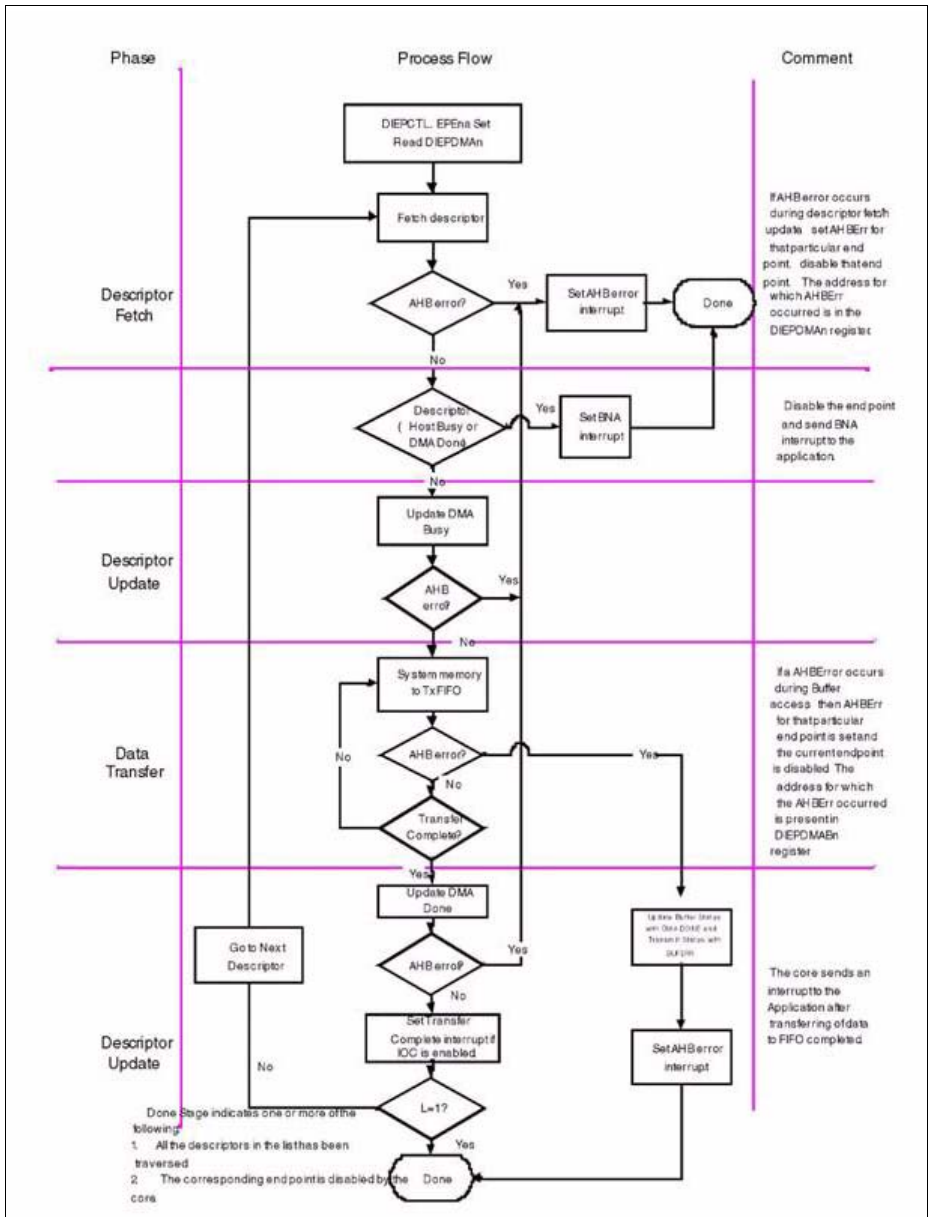


Figure 16-53 Non ISO IN Descriptor/Data Processing

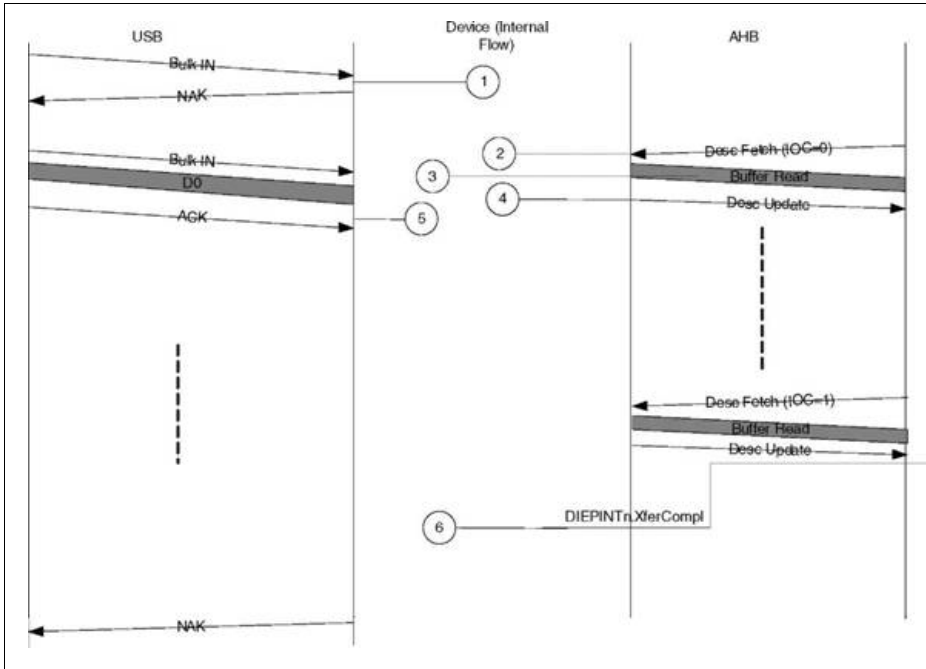


Figure 16-54 Bulk IN Transfers

1. When a BULK IN token is received on an end point before the corresponding DMA is enabled, (DIEPCTLx.EPEna = 0_B), it is NAKed on USB.
2. As a result of application enabling the DMA for the corresponding end point (DIEPCTLx.EPEna=1), the core fetches the descriptor and processes it.
3. The DMA fetches the data from the system memory and populates its internal FIFO with this data.
4. After fetching all the data from a descriptor, the core closes the descriptor with a DMA_DONE status.
5. On reception of BULK IN tokens on USB, data is sent to the USB Host.
6. After the last descriptor in the chain is processed, the core generates DIEPINTx.XferCompl interrupt provided the IOC bit for the last descriptor is set.

16.11.8.2 Bulk OUT Transfer in Scatter-Gather DMA Mode

Interrupt Usage

The following interrupts are of relevance.

1. DOEPINTx.XferCompl (Transfer complete, based on IOC bit in the descriptor)
2. DOEPINTx.BNA (Buffer Not Available)

Application Programming Sequence

This section describes the application programming sequence to take care of Bulk OUT transfer scenarios.

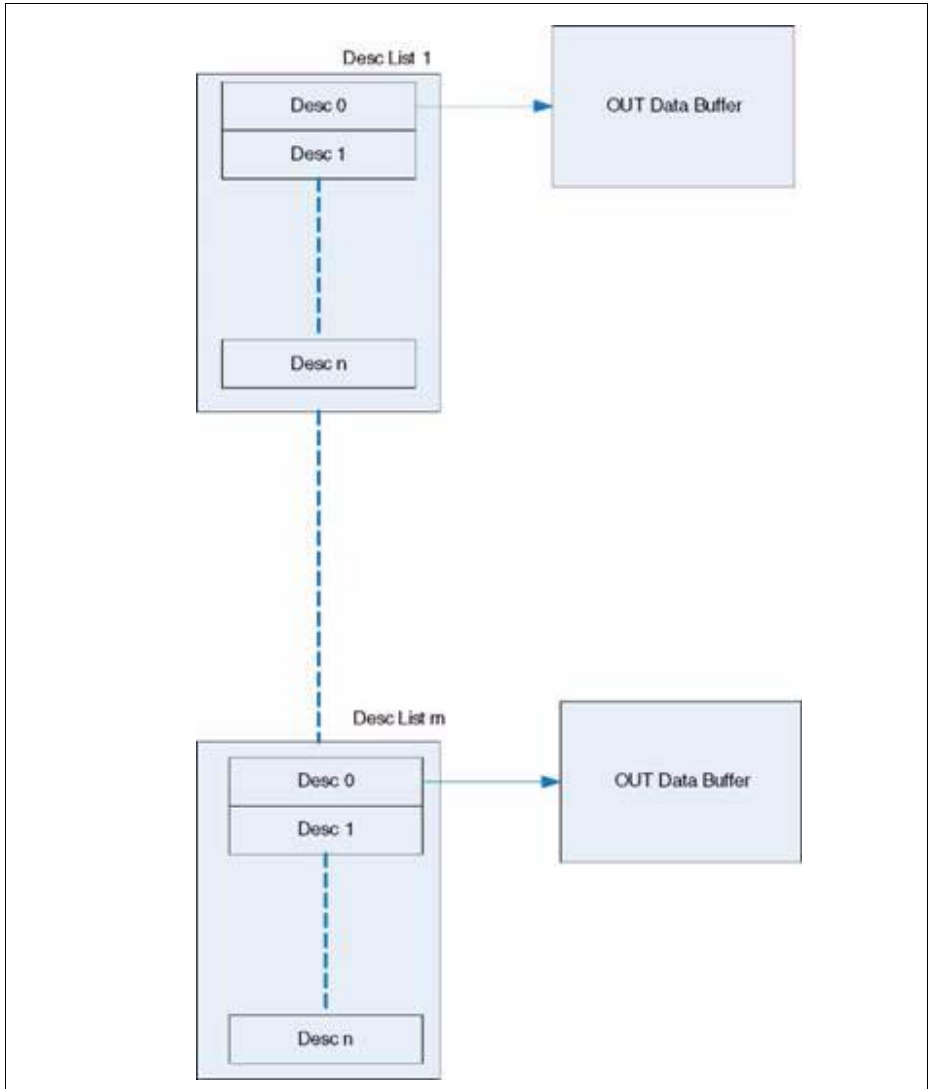


Figure 16-55 OUT Descriptor List

1. Prepare Descriptor(s):
2. The application creates descriptor list(s) in the system memory pertaining to an Endpoint.
3. Each descriptor list may have up to n descriptors and there may be up to m descriptor lists.
4. Application may choose to set the IOC bit of the corresponding descriptor. If the IOC is set for the last descriptor of the list, the core generates DOEPINTx.XferCompl interrupt after the entire list is processed.
 - a) a. Based on L bit and MTRF bit combinations, the core may disable the end point. Refer to **Table 16-9 “OUT Data Memory Structure Values” on Page 16-151** for bit field descriptions.
 - b) If the application programs the NAK bit, the core sets NAK for the endpoint after the descriptor is processed by the DMA. The application must set DIEPCTLn.CNAK to clear the NAK.
5. Program DOEPDMAx:
 - a) Application programs the base address of the descriptor in the corresponding OUT Endpoint DOEPDMAx register.
6. Enable DMA:
 - a) Application programs the corresponding endpoint DOEPCTLx register with the following:
 - DOEPCTL.MPS — Max Packet size of the endpoint
 - DOEPCTL.CNAK—Set to 1 to clear the NAK
 - DOEPCTL.EPEna — Set to 1 to enable the DMA for the endpoint.
7. Wait for Interrupt:
 - a) On reception of DOEPINTx.XferCompl, application must check the Buffer status and Rx Status field of the descriptor to ascertain that the descriptor closed normally.
 - b) On receiving DOEPINTn.BNA interrupt (generated by the core when it encounters a descriptor in the list whose Buffer Status field is not Host Ready), the application must read the DOEPDMA_n register to ascertain the address for which the BNA interrupt is asserted to take one of the following corrective actions:
 - If the application has programmed DCTL.EnContOnBNA = 1_B, then the application must read the DOEPDMA register, identify which descriptor received BNA status, process the descriptor, and re-enable the endpoint. The core continues to process descriptors from the descriptor which received BNA status.
 - If the application has programmed DCTL.EnContOnBNA = 1_B, then every time the endpoint is disabled, the core automatically sets the NAK for the endpoint, the application must set DIEPCTLn.CNAK = 1 while enabling the endpoint. This ensures that the core does not accept packets when the DMA is disabled for an endpoint.
 - If the application has programmed DCTL.EnContOnBNA = 0_B, then the application must read the DOEPDMA register, identify which descriptor received BNA status, process the descriptor, and re-enable the endpoint. The core

Universal Serial Bus (USB)

continues to process descriptors from the start of the descriptor chain (that is, the descriptor programmed in the DOEPDMA_n register).

Internal Flow

The core handles Bulk OUT transfers internally as depicted in [Figure 16-56](#). [Figure 16-57](#) also diagrams this flow.

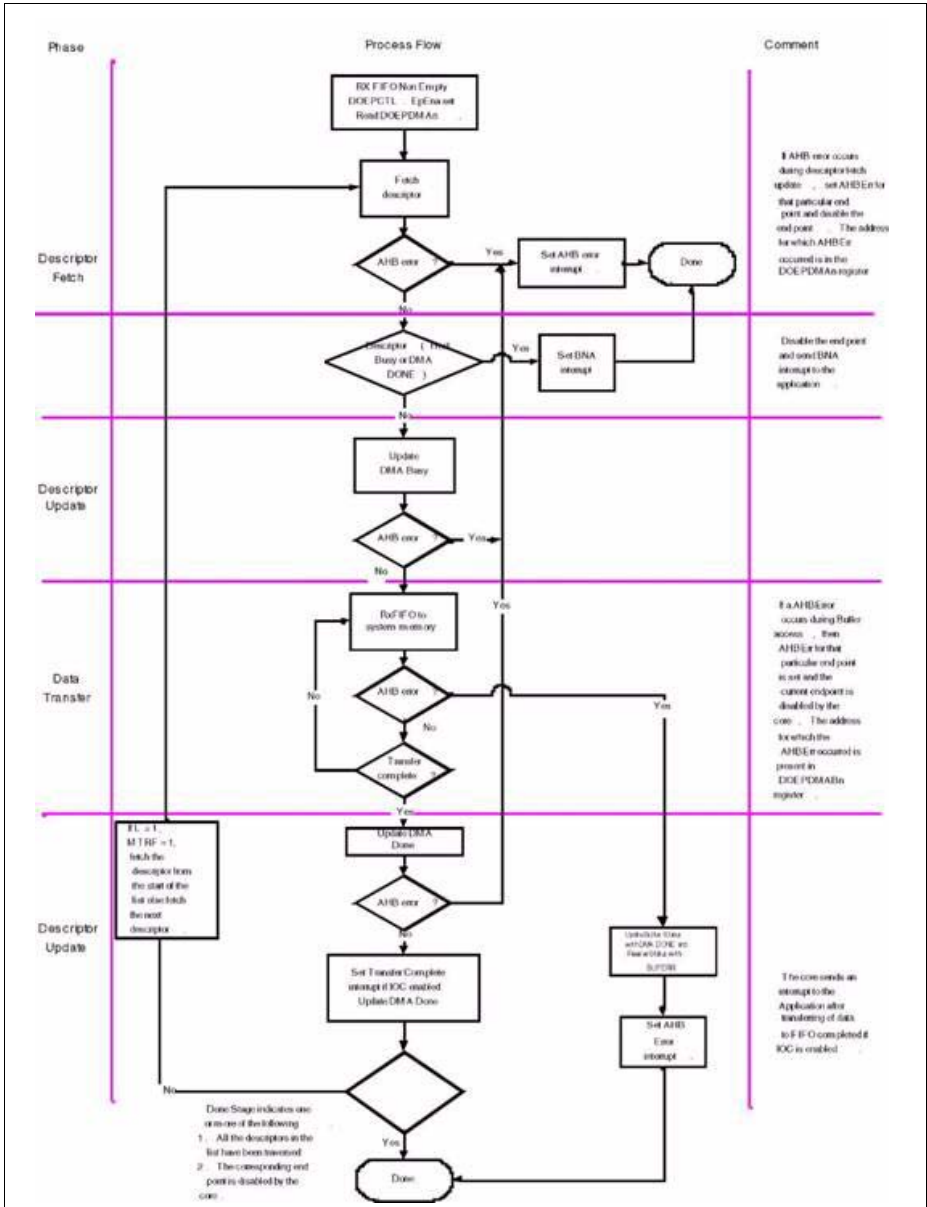


Figure 16-56 Non ISO OUT Descriptor/Data Buffer Processing

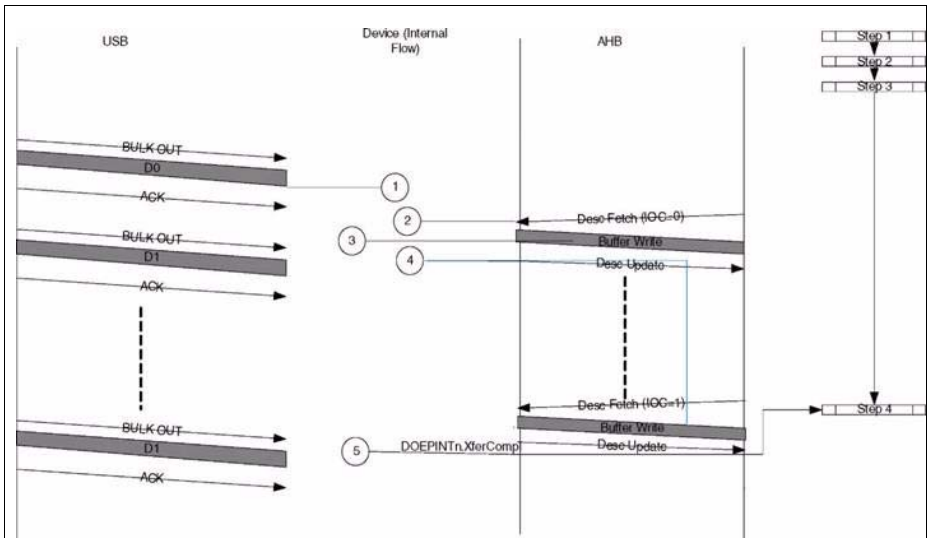


Figure 16-57 Bulk OUT Transfers

1. When a BULK OUT token is received on an end point, the core stores the received data internally in a FIFO.
2. As a result of application enabling the DMA for the corresponding end point (DOEPCTLx.EPEna=1), the core fetches the descriptor and processes it.
3. If the descriptor Buffer Status is HOST_READY:
 - a) The DMA transfers the data from the internal FIFO to system memory.
 - b) After transferring all the data from the FIFO, the core closes the descriptor with a DMA_DONE status.
 - c) After the last descriptor in the chain is processed, the core generates DOEPINTn.XferComp interrupt provided the IOC bit for the last descriptor is set.
4. If the descriptor Buffer Status is not HOST_READY:
 - a) The DMA generates Buffer Not Available (BNA) interrupt
 - b) After the application re-enables the corresponding endpoint (DOEPCTLn.EPEna=1), the core does the following:
 - Fetches the base descriptor from DOEPDMA_n (DCTL.EnContOnBNA = 0_B).
 - Fetches the descriptor that received the BNA interrupt is (DCTL.EnContOnBNA = 1_B).

16.11.9 Interrupt Transfer Handling in Scatter/Gather DMA Mode

16.11.9.1 Interrupt IN Transfer in Scatter/Gather DMA Mode

Application programming for Interrupt IN transfers is as with the Bulk IN transfer sequence. The core handles Interrupt IN transfers internally in the same way it handles Bulk IN transfers

16.11.9.2 Interrupt OUT Transfer in Scatter/Gather DMA Mode

Application programming for Interrupt OUT transfers is as with the Bulk OUT transfer sequence. The core handles Interrupt OUT transfers internally in the same way it handles Bulk OUT Transfers

16.11.10 Isochronous Transfer Handling in Scatter/Gather DMA Mode

16.11.10.1 Isochronous IN Transfer in Scatter/Gather DMA Mode

The application programming for Isochronous IN transfers is in the same manner as Bulk IN transfer sequence.

The following behavior is of importance while working with Isochronous IN end points

DCTL.IgnrFrmNum = 1_B

The way the core handles Isochronous IN transfers internally in the same way as it handles Bulk IN Transfers.

DCTL.IgnrFrmNum = 0_B

The core closes the descriptor and clears the corresponding fetched data in the FIFO if the USB frame number to which the descriptor belongs is elapsed.

Isochronous Transfers in Scatter/Gather (Descriptor DMA) Mode

This topic includes descriptions of both isochronous IN and OUT transfers

Isochronous IN

In the case of ISO IN After descriptor is fetched, the frame number field M is compared with current USB frame number N.

If the frame number in the fetched descriptor is already elapsed ($M < N$) then the descriptor is closed with status changed to DMA Done.

If the frame number in the fetched descriptor is for future ($N > M + 1$) then the descriptor is left untouched. The Core suspends and re-look at this descriptor contents in the next frame.

- If the frame number in the fetched descriptor is for current or next frame ($N = M$ or $M + 1$) then the descriptor is further processed as per the flow chart. At the end of data

Universal Serial Bus (USB)

transfer from memory to Tx FIFO the above check must be performed. And if the data fetch finished in the subsequent frame, data must be flushed and descriptor must be closed (DMA Done) with BUFLUSH status.

- For ISO IN, the application creates a series of descriptors (D,D+1,D+2) for a given periodic end point corresponding to successive frames (N,N+1,N+2).

Note: The series of descriptors does not correspond to the series of frames in the same order.

For example, D and D + 1 may correspond to N, D + 2 may correspond to N + 1 and so on except in the case where the application can create more than one descriptor for the same frame. The core fetches the descriptor and compares the frame/ ^frame number field with the current frame/ ^frame number.

If the fetched descriptor corresponds to a frame which has already elapsed, the core updates the descriptor with DMA Done Buffer status and proceeds to the next descriptor.

If the next descriptor fetched indicates that it corresponds to frame number N or N + 1, it services it. In the process of fetching the descriptors, if the core determines that the descriptor corresponds to a future frame/ ^frame (> N + 1), it does not service the descriptor in that frame/ ^frame. Instead, it moves on to the next periodic endpoint or non-periodic endpoint without disabling the current periodic endpoint. It revisits this endpoint in the next frame/ ^frame and repeats the process.

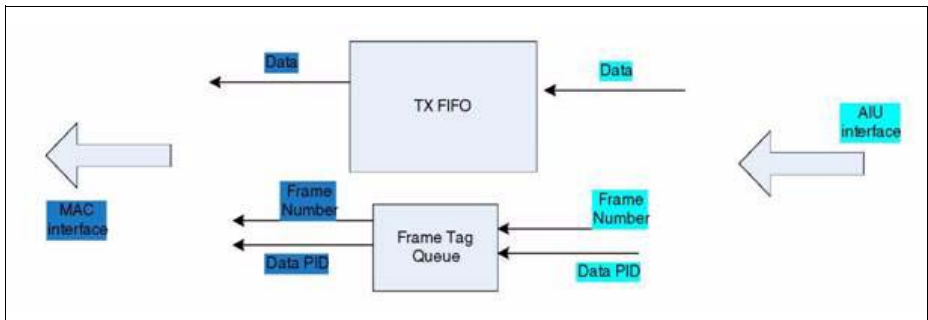


Figure 16-58 ISO IN Data Flow

Application Programming Sequence

This section describes the application programming sequence for Isochronous IN transfer scenarios.

Prepare Descriptor(s)

The application creates descriptor list(s) in the system memory pertaining to an Endpoint.

Universal Serial Bus (USB)

Each descriptor list may have up to n descriptors and there may be up to m descriptor lists.

Application may choose to set the IOC bit of the corresponding descriptor. If the IOC is set for the last descriptor of the list, the core generates DIEPINTx.XferCompl interrupt after the entire list is processed.

1. Program DIEPDMAx:
 - a) Application programs the base address of the descriptor in the corresponding IN Endpoint DIEPDMAx register.
2. Enable DMA:
 - a) Application programs the corresponding endpoint DIEPCTLx register with the following
 - DIEPCTLx.MPS — Max Packet size of the endpoint
 - DIEPCTLx.CNAK—Set to 1 to clear the NAK
 - DIEPCTLx.EPEna — Set to 1 to enable the DMA for the endpoint.
3. Wait for Interrupt:
 - a) On reception of DIEPINTx.XferCompl, application must check the Buffer status and Tx Status field of the descriptor to ascertain that the descriptor closed normally.

DIEPINTx.BNA interrupt gets generated by the core when it encounters a descriptor in the list whose Buffer Status field is not Host Ready. In this case, the application is suppose to read the DIEPDMAx register to ascertain the address for which the BNA interrupt is asserted to take corrective action.

Internal Flow

The core handles isochronous IN transfers internally as functionally depicted in [Figure 16-59](#). [Figure 16-60](#) also diagrams this flow.

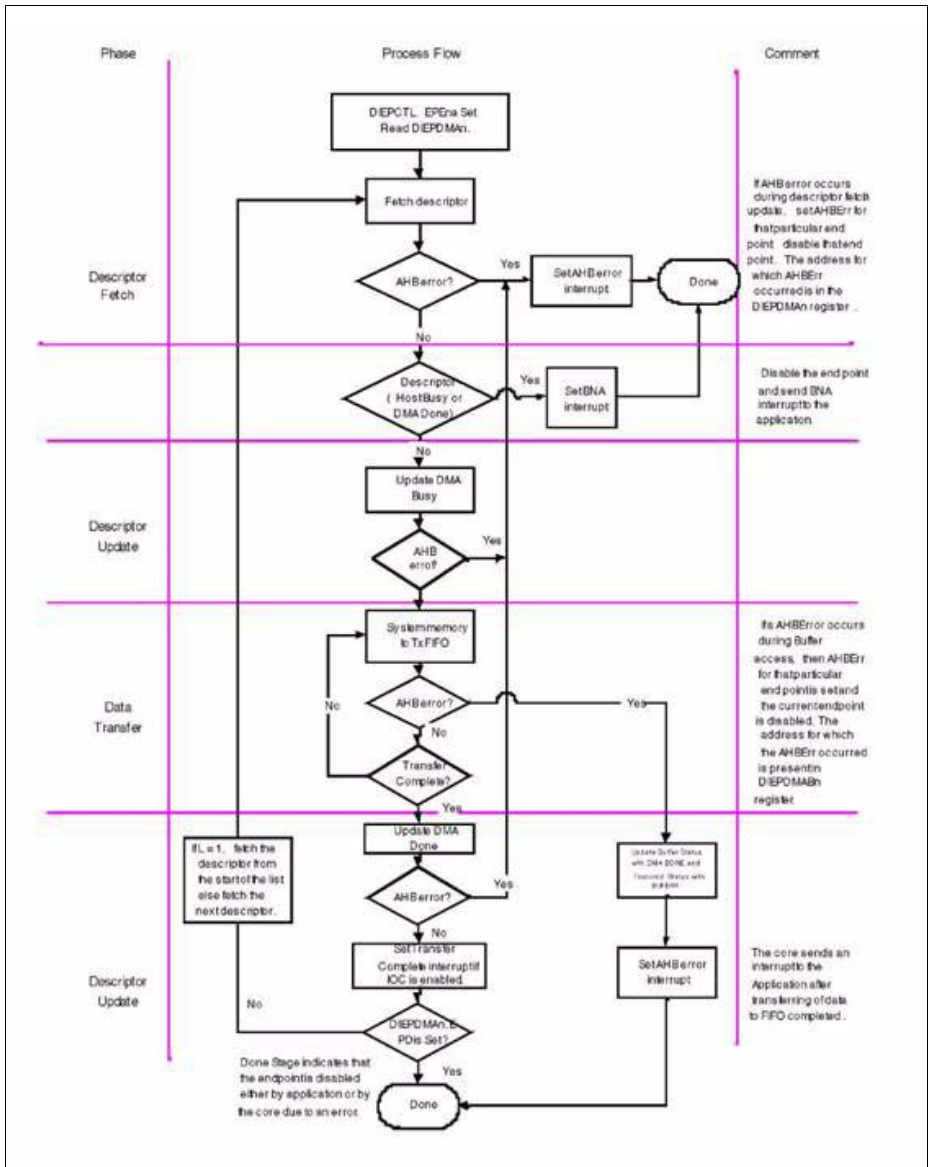


Figure 16-59 ISO IN Descriptor/Data Processing

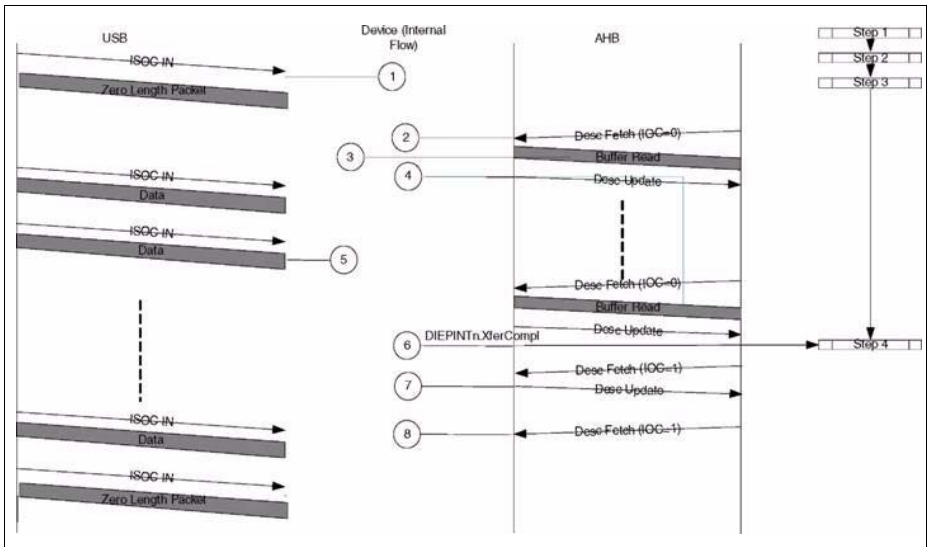


Figure 16-60 Isochronous IN Transfers

1. When an Isochronous IN token is received on an end point before the corresponding DMA is enabled, ($DIEPCTLx.EPEna = 0_B$), zero length packet is sent on USB.
2. As a result of application enabling the DMA for the corresponding end point ($DIEPCTLx.EPEna=1$), the core fetches the descriptor. If the descriptor belongs to the current or the next USB frame number, the core processes it.
3. The DMA fetches the data pointed by the above descriptor from the system memory and populates its internal FIFO with this data.
4. After fetching all the data, the core closes the descriptor with a `DMA_DONE` status.
5. On reception of Isochronous IN tokens on USB, data is sent to the USB Host.
6. After the last descriptor in the chain is processed, the core generates `DIEPINTx.XferCompl` interrupt provided the IOC bit for the last descriptor is set.
7. When the DMA fetches a descriptor whose USB frame number has been already elapsed, it closes that descriptor with a `DMA_DONE` status without fetching the data for that descriptor.
8. When the DMA fetches a descriptor which has a future USB frame number, it does not service it in the current context. It services it in the future.

16.11.10.2 Isochronous OUT Transfer in Scatter/Gather DMA Mode

The application programming for isochronous out transfers is in the same manner as Bulk OUT transfer sequence, except that the application creates only 1 packet per descriptor for an isochronous OUT endpoint. The core handles isochronous OUT transfers internally in the same way it handles Bulk OUT transfers, and as depicted in [Figure 16-61](#).

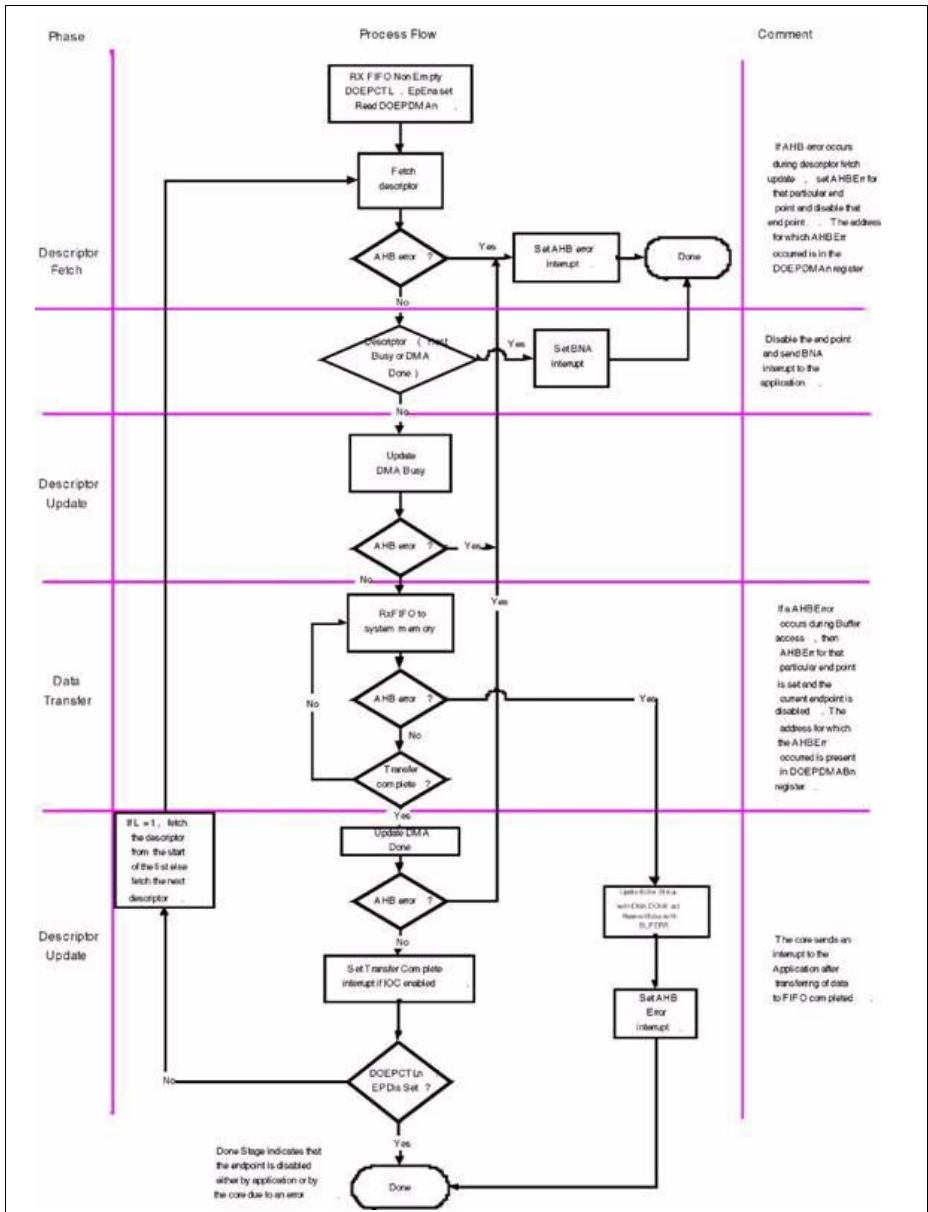


Figure 16-61 Isochronous OUT Descriptor/Data Buffer Processing

Isochronous OUT

For ISO OUT transactions, the core transfers the packets from the Rx FIFO to the system memory and updates the frame number field of the descriptor with the frame number in which the packet was received. The frame number for which data is received is extracted from the Receive Status queue and written back to the descriptor.

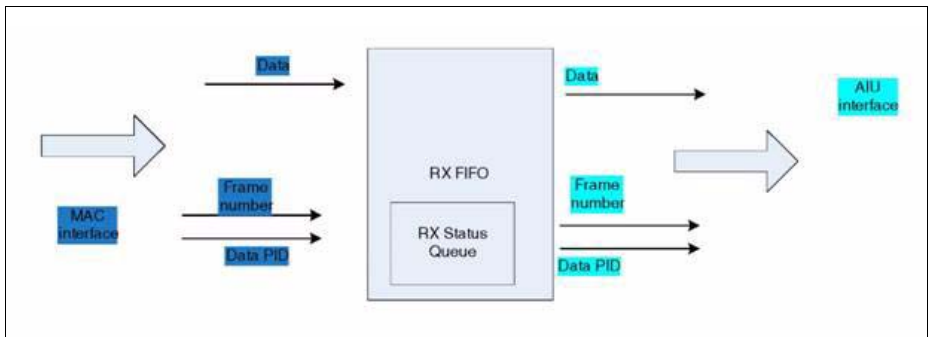


Figure 16-62 ISO Out Data Flow

Note: Incomplete Isochronous Interrupt (GINTSTS.incomplete) is not generated in Scatter/Gather DMA mode. Received isochronous packets are sent unmodified to the application memory, with the corresponding frame number updated in the descriptor status.

16.12 OTG Revision 1.3 Programming Model

This section describes the OTG programming model when the OTG core is configured to support OTG Revision 1.3 of the specification.

The USB core is an OTG device supporting HNP and SRP. When the core is connected to an "A" plug, it is referred to as an A-device. When the core is connected to a "B" plug it is referred to as a B-device. In Host mode, the USB core turns off VBUS to conserve power. SRP is a method by which the B-device signals the A-device to turn on VBUS power. A device must perform both data-line pulsing and VBUS pulsing, but a host can detect either data-line pulsing or VBUS pulsing for SRP. HNP is a method by which the B-device negotiates and switches to host role. In Negotiated mode after HNP, the B-device suspends the bus and reverts to the device role.

16.12.1 A-Device Session Request Protocol

The application must set the SRP-Capable bit in the Core USB Configuration register. This enables the USB core to detect SRP as an A-device.

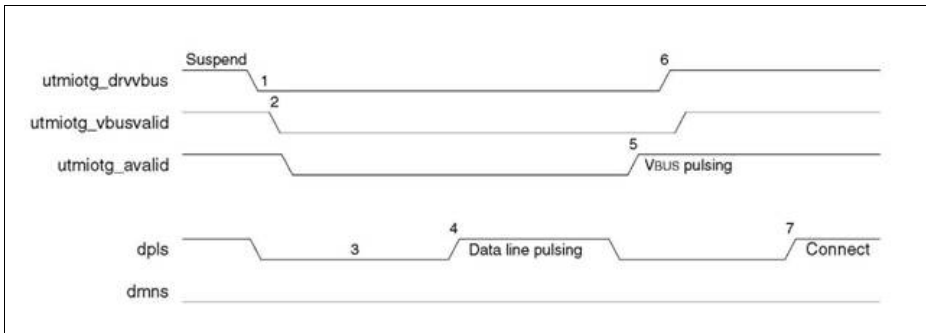


Figure 16-63 A-Device SRP

1. To save power, the application suspends and turns off port power when the bus is idle by writing the port Suspend and Port Power bits in the Host Port Control and Status register.
2. PHY indicates port power off by deasserting the `utmi_vbusvalid` signal.
3. The device must detect SE0 for at least 2 ms to start SRP when Vbus power is off.
4. To initiate SRP, the device turns on its data line pull-up resistor for 5 to 10 ms. The USB core detects data-line pulsing.
5. The device drives VBUS above the A-device session valid (2.0 V minimum) for VBUS pulsing.
The USB core interrupts the application on detecting SRP. The Session Request Detected bit is set in Global Interrupt Status register (GINTSTS.SessReqInt).
6. The application must service the Session Request Detected interrupt and turn on the Port Power bit by writing the Port Power bit in the Host Port Control and Status register. The PHY indicates port power-on by asserting `utmi_vbusvalid` signal.
7. When the USB is powered, the device connects, completing the SRP process.

16.12.2 B-Device Session Request Protocol

The application must set the SRP-Capable bit in the Core USB Configuration register. This enables the USB core to initiate SRP as a B-device. SRP is a means by which the USB core can request a new session from the host.

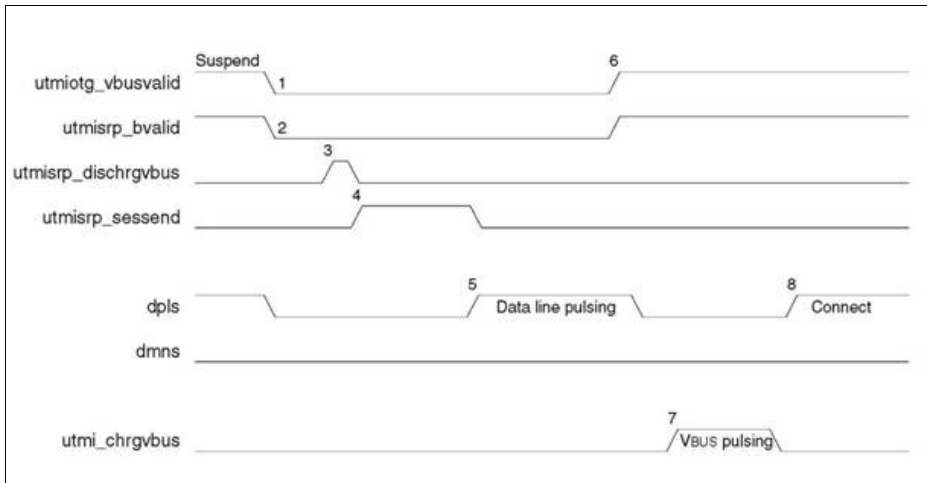


Figure 16-64 B-Device SRP

1. To save power, the host suspends and turns off port power when the bus is idle. PHY indicates port power off by deasserting the utmi_vbusvalid signal. The USB core sets the Early Suspend bit in the Core Interrupt register after 3 ms of bus idleness. Following this, the USB core sets the USB Suspend bit in the Core Interrupt register. The PHY indicates the end of the B-device session by deasserting the utmi_bvalid signal.
2. The USB core asserts the utmi_dischrgvbus signal to indicate to the PHY to speed up VBUS discharge.
3. The PHY indicates the session's end by asserting the utmi_sessend signal. This is the initial condition for SRP. The USB core requires 2 ms of SE0 before initiating SRP.
For a USB 1.1 full-speed serial transceiver, the application must wait until VBUS discharges to 0.2 V after GOTGCTL.BSesVld is deasserted.
4. The application initiates SRP by writing the Session Request bit in the OTG Control and Status register. The USB core perform data-line pulsing followed by VBUS pulsing.
5. The host detects SRP from either the data-line or VBUS pulsing, and turns on VBUS. The PHY indicates VBUS power-on by asserting utmi_vbusvalid.
6. The USB core performs VBUS pulsing by asserting utmi_chrgvbus.
The host starts a new session by turning on VBUS, indicating SRP success. The USB core interrupts the application by setting the Session Request Success Status Change bit in the OTG Interrupt Status register. The application reads the Session Request Success bit in the OTG Control and Status register.
7. When the USB is powered, the USB core connects, completing the SRP process.

16.12.3 A-Device Host Negotiation Protocol

HNP switches the USB host role from the A-device to the B-device. The application must set the HNP- Capable bit in the Core USB Configuration register to enable the USB core to perform HNP as an A-device.

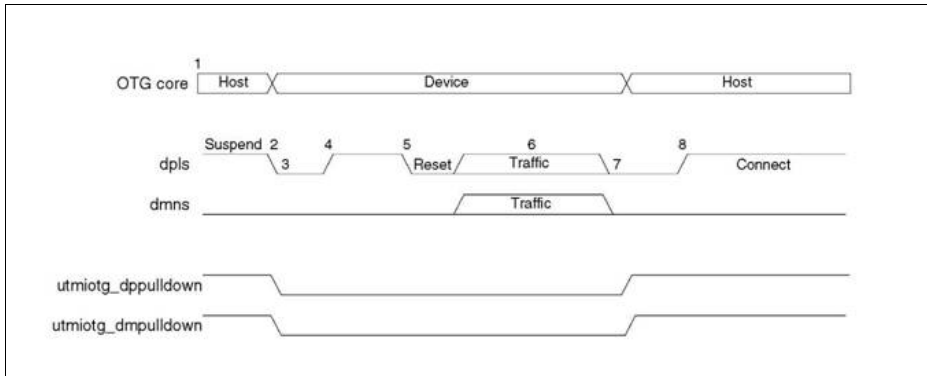


Figure 16-65 A-Device HNP

1. The USB core sends the B-device a SetFeature b_hnp_enable descriptor to enable HNP support. The B-device's ACK response indicates that the B-device supports HNP. The application must set Host Set HNP Enable bit in the OTG Control and Status register to indicate to the USB core that the B-device supports HNP.
2. When it has finished using the bus, the application suspends by writing the Port Suspend bit in the Host Port Control and Status register.
3. When the B-device observes a USB suspend, it disconnects, indicating the initial condition for HNP. The B-device initiates HNP only when it must switch to the host role; otherwise, the bus continues to be suspended.

The USB core sets the Host Negotiation Detected interrupt in the OTG Interrupt Status register, indicating the start of HNP.

The USB core deasserts the utmiotg_dppulldown and utmiotg_dmpulldown signals to indicate a device role. The PHY enable the D+ pull-up resistor indicates a connect for B-device.

The application must read the Current Mode bit in the OTG Control and Status register to determine Device mode operation.
4. The B-device detects the connection, issues a USB reset, and enumerates the USB core for data traffic.
5. The B-device continues the host role, initiating traffic, and suspends the bus when done.

The USB core sets the Early Suspend bit in the Core Interrupt register after 3 ms of bus idleness. Following this, the USB core sets the USB Suspend bit in the Core Interrupt register.

Universal Serial Bus (USB)

6. In Negotiated mode, the USB core detects the suspend, disconnects, and switches back to the host role. The USB core asserts the `utmio_tg_dppulldown` and `utmio_tg_dmpulldown` signals to indicate its assumption of the host role.
7. The USB core sets the Connector ID Status Change interrupt in the OTG Interrupt Status register. The application must read the connector ID status in the OTG Control and Status register to determine the USB core's operation as an A-device. This indicates the completion of HNP to the application. The application must read the Current Mode bit in the OTG Control and Status register to determine Host mode operation.
8. The B-device connects, completing the HNP process.

16.12.4 B-Device Host Negotiation Protocol

HNP switches the USB host role from B-device to A-device. The application must set the HNP-Capable bit in the Core USB Configuration register to enable the USB core to perform HNP as a B-device.

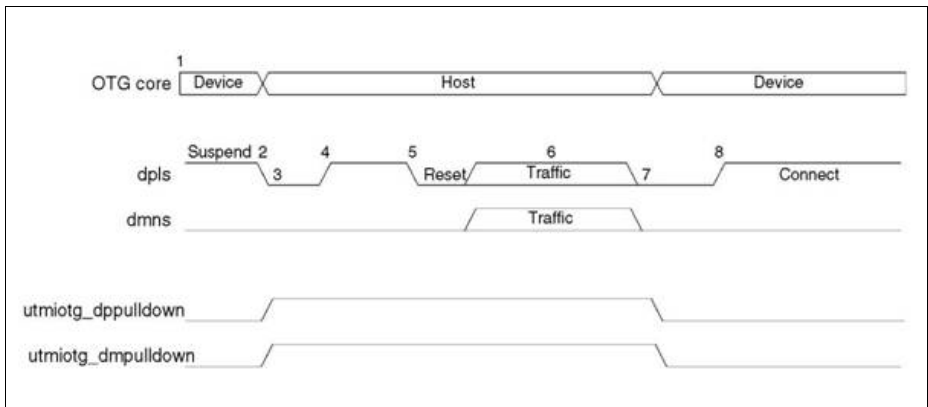


Figure 16-66 B-Device HNP

1. The A-device sends the SetFeature `b_hnp_enable` descriptor to enable HNP support. The USB core's ACK response indicates that it supports HNP. The application must set the Device HNP Enable bit in the OTG Control and Status register to indicate HNP support.
The application sets the HNP Request bit in the OTG Control and Status register to indicate to the USB core to initiate HNP.
2. When it has finished using the bus, the A-device suspends by writing the Port Suspend bit in the Host Port Control and Status register.
The USB core sets the Early Suspend bit in the Core Interrupt register after 3 ms of bus idleness. Following this, the USB core sets the USB Suspend bit in the Core Interrupt register.

Universal Serial Bus (USB)

The USB core disconnects and the A-device detects SE0 on the bus, indicating HNP. The USB core asserts the `utmio_tg_dppulldown` and `utmio_tg_dmpulldown` signals to indicate its assumption of the host role.

The A-device responds by activating its D+ pull-up resistor within 3 ms of detecting SE0. The USB core detects this as a connect.

The USB core sets the Host Negotiation Success Status Change interrupt in the OTG Interrupt Status register, indicating the HNP status. The application must read the Host Negotiation Success bit in the OTG Control and Status register to determine host negotiation success. The application must read the Current Mode bit in the Core Interrupt register (GINTSTS) to determine Host mode operation.

3. The application sets the reset bit (HPRT.PrtRst) and the USB core issues a USB reset and enumerates the A-device for data traffic
4. The USB core continues the host role of initiating traffic, and when done, suspends the bus by writing the Port Suspend bit in the Host Port Control and Status register.
5. In Negotiated mode, when the A-device detects a suspend, it disconnects and switches back to the host role. The USB core deasserts the `utmio_tg_dppulldown` and `utmio_tg_dmpulldown` signals to indicate the assumption of the device role.
6. The application must read the Current Mode bit in the Core Interrupt (GINTSTS) register to determine the Host mode operation.
7. The USB core connects, completing the HNP process.

16.13 Clock Gating Programming Model

When the USB is suspended or the session is not valid, the PHY is driven into Suspend mode, and the PHY clock is stopped to reduce power consumption in the PHY and the USB core. The PHY clock is turned off for as long as the core asserts the suspend signal.

To further reduce power consumption, the USB core also supports AHB clock gating. The AHB clock to some of the USB internal modules can be gated by writing to the Gate Hclk bit in the Power and Clock Gating Control register.

The following sections show the procedures to use the clock gating feature.

16.13.1 Host Mode Suspend and Resume With Clock Gating

Sequence of operations:

1. The application sets the Port Suspend bit in the Host Port Control and Signal register, and the core drives a USB suspend.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core asserts the suspend signal to the PHY, and the PHY clock stops. The application sets the Gate hclk bit in the Power and Clock Gating Control register, the core gates the hclk (`hclk_gated`) to AHB- domain modules other than the BIU.
3. The core remains in Suspend mode.
4. The application clears the Gate hclk and Stop PHY Clock bits, and the PHY clock is generated.

Universal Serial Bus (USB)

5. The application sets the Port Resume bit, and the core starts driving Resume signaling.
6. The application clears the Port Resume bit after at least 20 ms.
7. The core is in normal operating mode.

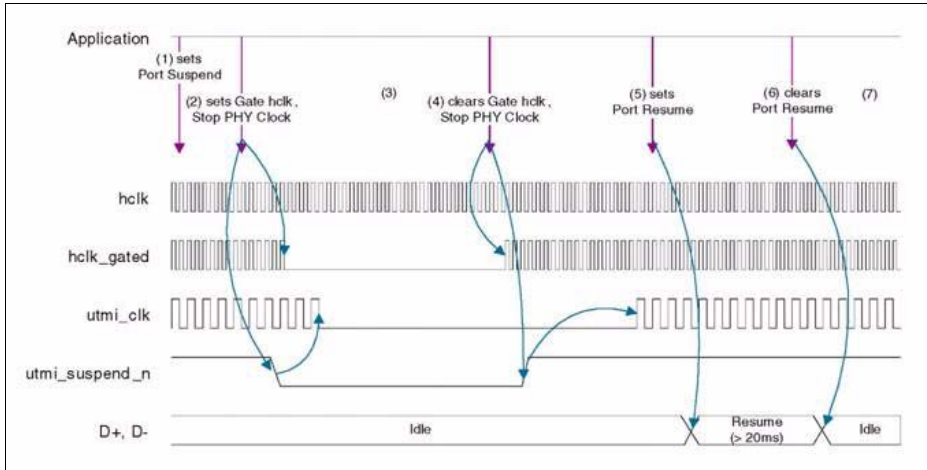


Figure 16-67 Host Mode Suspend and Resume With Clock Gating

16.13.2 Host Mode Suspend and Remote Wakeup With Clock Gating

Sequence of operations:

1. The application sets the Port Suspend bit in the Host Port CSR, and the core drives a USB suspend.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core asserts the suspend_n signal to the PHY, and the PHY clock stops. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates the hclk (hclk_ctl) to AHB- domain modules other than the BIU.
3. The core remains in Suspend mode.
4. The Remote Wakeup signaling from the device is detected. The core deasserts the suspend_n signal to the PHY to generate the PHY clock. The core generates a Remote Wakeup Detected interrupt.
5. The application clears the Gate hclk and Stop PHY Clock bits. The core sets the Port Resume bit.
6. The application clears the Port Resume bit after at least 20 ms.
7. The core is in normal operating mode.

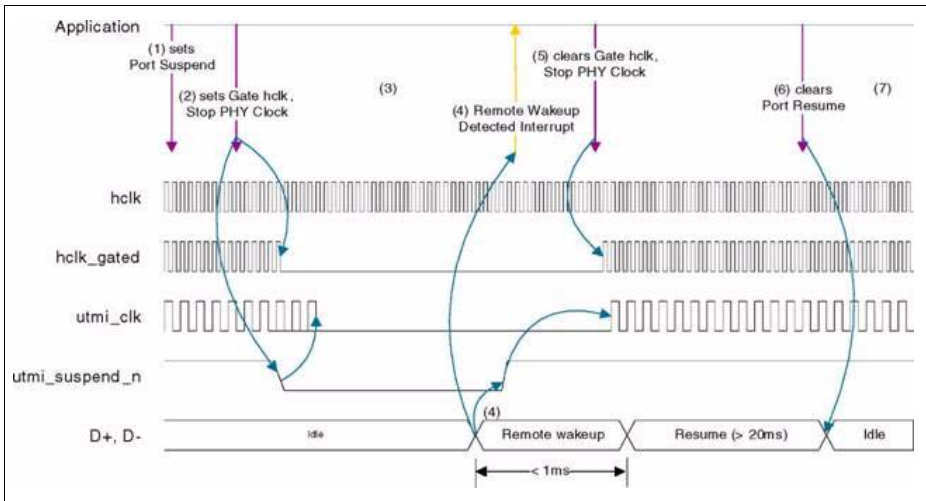


Figure 16-68 Host Mode Suspend and Remote Wakeup With Clock Gating

16.13.3 Host Mode Session End and Start With Clock Gating

Sequence of operations:

1. The application sets the Port Suspend bit in the Host Port CSR, and the core drives a USB suspend.
2. The application clears the Port Power bit. The core turns off VBUS.
3. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core asserts the suspend_n signal to the PHY, and the PHY clock stops. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates the hclk (hclk_ctl) to AHB- domain modules other than the BIU.
4. The core remains in Low-Power mode.
5. The application clears the Gate hclk bit and the application clears the Stop PHY Clock bit to start the PHY clock.
6. The application sets the Port Power bit to turn on VBUS.
7. The core detects device connection and drives a USB reset.
8. The core is in normal operating mode.

16.13.4 Host Mode Session End and SRP With Clock Gating

Sequence of operations:

1. The application sets the Port Suspend bit in the Host Port CSR, and the core drives a USB suspend.
2. The application clears the Port Power bit. The core turns off VBUS.

Universal Serial Bus (USB)

3. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core asserts the suspend_n signal to the PHY, and the PHY clock stops. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates the hclk (hclk_ctl) to AHB- domain modules other than the BIU.
4. The core remains in Low-Power mode.
5. SRP (data line pulsing) from the device is detected. The core deasserts the suspend_n signal to the PHY to generate the PHY clock. An SRP Request Detected interrupt is generated.
6. The application clears the Gate hclk bit and the Stop PHY Clock bit.
7. The core sets the Port Power bit to turn on VBUS.
8. The core detects device connection and drives a USB reset.
9. The core is in normal operating mode.

16.13.5 Device Mode Suspend and Resume With Clock Gating

Sequence of operations:

1. The core detects a USB suspend and generates a Suspend Detected interrupt.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core asserts the suspend_n signal to the PHY, and the PHY clock stops. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates the hclk (hclk_ctl) to AHB- domain modules other than the BIU.
3. The core remains in Suspend mode.
4. The Resume signaling from the host is detected. The core deasserts the suspend_n signal to the PHY to generate the PHY clock. A Resume Detected interrupt is generated.
5. The application clears the Gate hclk bit and the Stop PHY Clock bit.
6. The host finishes Resume signaling.
7. The core is in normal operating mode.

16.13.6 Device Mode Suspend and Remote Wakeup With Clock Gating

Sequence of operations:

1. The core detects a USB suspend and generates a Suspend Detected interrupt.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core asserts the suspend_n signal to the PHY, and the PHY clock stops. The application sets the Gate hclk bit in the Power and Clock Gating Control register, the core gates the hclk (hclk_ctl) to AHB-domain modules other than the BIU.
3. The core remains in Suspend mode.
4. The application clears the Gate hclk bit and the Stop PHY Clock bit.
5. The application sets the Remote Wakeup bit in the Device Control register, the core starts driving Remote Wakeup signaling.
6. The host drives Resume signaling.
7. The core is in normal operating mode.

16.13.7 Device Mode Session End and Start With Clock Gating

Sequence of operations:

1. The core detects a USB suspend, and generates a Suspend Detected interrupt. The host turns off VBUS.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core asserts the suspend_n signal to the PHY, and the PHY clock stops. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates the hclk (hclk_ctl) to AHB- domain modules other than the BIU.
3. The core remains in Low-Power mode.
4. The new session is detected (bsessvld is high). The core deasserts the suspend_n signal to the PHY to generate the PHY clock. A New Session Detected interrupt is generated.
5. The application clears the Gate hclk and Stop PHY Clock bits.
6. The core detects USB reset.
7. The core is in normal operating mode

16.13.8 Device Mode Session End and SRP With Clock Gating

Sequence of operations:

1. The core detects a USB suspend, and generates a Suspend Detected interrupt. The host turns off VBUS.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core asserts the suspend_n signal to the PHY, and the PHY clock stops. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates the hclk (hclk_ctl) to AHB- domain modules other than the BIU.
3. The core remains in Low-Power mode.
4. The application clears the Gate hclk and Stop PHY Clock bits.
5. The application sets the SRP Request bit, and the core drives data line and VBUS pulsing.
6. The host turns on Vbus, detects device connection, and drives a USB reset.
7. The core is in normal operating mode.

16.14 FIFO RAM Allocation

16.14.1 Data FIFO RAM Allocation

The RAM must be allocated among different FIFOs in the core before any transactions can start. The application must follow this procedure every time it changes core FIFO RAM allocation.

The application must allocate data RAM per FIFO based on the following criteria:

Universal Serial Bus (USB)

- AHB's operating frequency
- PHY Clock frequency
- Available AHB bandwidth
- Performance required on the USB

Based on the above criteria, the application must provide a table with RAM sizes for each FIFO in each mode.

USB core shares a single SPRAM between transmit FIFO(s) and receive FIFO.

In DMA mode — The SPRAM is also used for storing some register information:

- **In non Scatter Gather mode** — The Device mode Endpoint DMA address registers (DI/OEPDMA_n) and Host mode Channel DMA registers (HCDMA) are stored in the SPRAM.
- **In Scatter Gather mode** — The Base descriptor address, the Current descriptor address, the current buffer address and the descriptor status quadlet information for each endpoint/channel are stored in the SPRAM.

These register information are stored at the end of the SPRAM after the space allocated for receive and Transmit FIFO. These register space must also be taken into account when allocating the RAM among the different FIFOs.

In Slave mode — No registers are stored in the SPRAM. Therefore no additional space needs to be allocated in the SPRAM for register information.

The following rules apply while calculating how much RAM space must be allocated to store these registers.

Table 16-17 RAM Space Allocation

Mode	Configuration	RAM Space Allocation
Host	Slave mode	No space required
	Buffer DMA mode	One location per channel
	Scatter/Gather DMA mode	Four locations per channel as follows: - Location for storing current descriptor address - Location for storing current buffer address - Location for storing the status quadlet that is used by the List processor - Location for storing the transfer size used by the token request block

Table 16-17 RAM Space Allocation

Mode	Configuration	RAM Space Allocation
Device	Slave mode	No space required
	Buffer DMA mode	One location per endpoint direction
	Scatter/Gather DMA mode	Four locations per endpoint direction as follows: - Location for storing base descriptor address - Location for storing current descriptor address - Location for storing current buffer address - Location for storing descriptor status quadlet

For example in Scatter/Gather DMA mode, if there are five bidirectional endpoints, then the last forty SPRAM locations are reserved for storing these values.

16.14.1.1 Device Mode RAM Allocation

Considerations for allocating data RAM for Device Mode FIFOs are listed here:

1. Receive FIFO RAM Allocation:
 - a) RAM for SETUP Packets: 10 locations must be reserved in the receive FIFO to receive up to n SETUP packets on the control endpoint. The core does not use these locations, which are reserved for SETUP packets, to write any other data.
 - b) One location for Global OUT NAK
 - c) Status information is written to the FIFO along with each received packet. Therefore, a minimum space of $(\text{Largest Packet Size} / 4) + 1$ must be allotted to receive packets. If a high-bandwidth endpoint is enabled, or multiple isochronous endpoints are enabled, then at least two $(\text{Largest Packet Size} / 4) + 1$ spaces must be allotted to receive back-to-back packets. Typically, two $(\text{Largest Packet Size} / 4) + 1$ spaces are recommended so that when the previous packet is being transferred to AHB, the USB can receive the subsequent packet. If AHB latency is high, enough space must be allocated to receive multiple packets. This is critical to prevent dropping any isochronous packets.
 - d) Along with each endpoint's last packet, transfer complete status information is also pushed to the FIFO. Typically, one location for each OUT endpoint is recommended.
2. Transmit FIFO RAM Allocation:
 - a) The minimum RAM space required for each IN Endpoint Transmit FIFO is the maximum packet size for that particular IN endpoint.
 - b) More space allocated in the transmit IN Endpoint FIFO results in a better performance on the USB and can hide latencies on the AHB.

Table 16-18 FIFO Name - Data RAM Size

FIFO Name	Data RAM Size
Receive data FIFO	rx_fifo_size. This must include RAM for setup packets, OUT endpoint control information and data OUT packets.
Transmit FIFO 0	tx_fifo_size[0]
Transmit FIFO 1	tx_fifo_size[1]
Transmit FIFO 2	tx_fifo_size[2]
...	...
Transmit FIFO i	tx_fifo_size[i]

With this information, the following registers must be programmed as follows:

1. Receive FIFO Size Register (GRXFSIZ)
GRXFSIZ.Receive FIFO Depth = rx_fifo_size;
2. Device IN Endpoint Transmit FIFO0 Size Register (GNPTXFSIZ)
GNPTXFSIZ.non-periodic Transmit FIFO Depth = tx_fifo_size[0];
GNPTXFSIZ.non-periodic Transmit RAM Start Address = rx_fifo_size;
3. Device IN Endpoint Transmit FIFO#1 Size Register (DIEPTXF1)
DIEPTXF1. Transmit RAM Start Address = GNPTXFSIZ.FIFO0 Transmit RAM Start Address + tx_fifo_size[0];
4. Device IN Endpoint Transmit FIFO#2 Size Register (DIEPTXF2)
DIEPTXF2.Transmit RAM Start Address = DIEPTXF1.Transmit RAM Start Address + tx_fifo_size[1];
5. Device IN Endpoint Transmit FIFO#i Size Register (DIEPTXFi)
DIEPTXFm.Transmit RAM Start Address = DIEPTXFi-1.Transmit RAM Start Address + tx_fifo_size[i-1];
6. The transmit FIFOs and receive FIFO must be flushed after the RAM allocation is done, for the proper functioning of the FIFOs.
 - a) GRSTCTL.TxFNum = 10_H
 - b) GRSTCTL.TxFFlush = 1_B
 - c) GRSTCTL.RxFFlush = 1_B
 - d) The application must wait until the TxFFlush bit and the RxFFlush bits are cleared before performing any operation on the core.

See also [Figure 16-69](#).

Universal Serial Bus (USB)

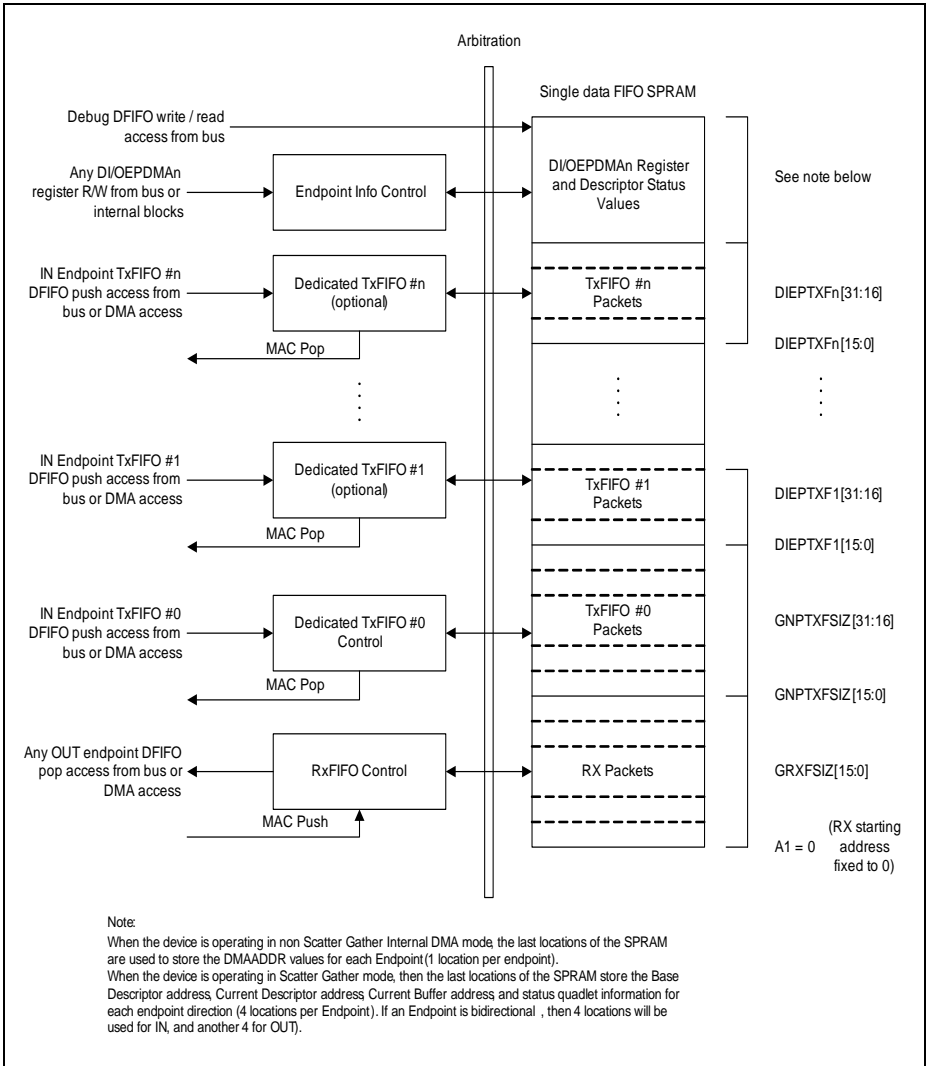


Figure 16-69 Device Mode FIFO Address Mapping

16.14.1.2 Host Mode RAM Allocation

Considerations for allocating data RAM for Host Mode FIFOs are listed here:

Receive FIFO RAM allocation:

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of $(\text{Largest Packet Size} / 4) + 2$ must be allotted to receive packets. If a high-bandwidth channel is enabled, or multiple isochronous channels are enabled, then at least two $(\text{Largest Packet Size} / 4) + 2$ spaces must be allotted to receive back-to-back packets. Typically, two $(\text{Largest Packet Size} / 4) + 2$ spaces are recommended so that when the previous packet is being transferred to AHB, the USB can receive the subsequent packet. If AHB latency is high, enough space must be allocated to receive multiple packets.

Along with each host channel's last packet, information on transfer complete status and channel halted is also pushed to the FIFO. So two locations must be allocated for this.

For handling NAK/NYET in Buffer DMA mode, the application must determine the number of Control/Bulk OUT endpoint data that must fit into the TX_FIFO at the same instant. Based on this, one location each is required for Control/Bulk OUT endpoints.

For example, when the host addresses one Control OUT endpoint and three Bulk OUT endpoints, and all these must fit into the non-periodic TX_FIFO at the same time, then four extra locations are required in the RX FIFO to store the rewind status information for each of these endpoints.

Transmit FIFO RAM allocation

The minimum amount of RAM required for the Host Non-periodic Transmit FIFO is the largest maximum packet size among all supported non-periodic OUT channels.

More space allocated in the Transmit Non-periodic FIFO results in better performance on the USB and can hide AHB latencies. Typically, two Largest Packet Sizes' worth of space is recommended, so that when the current packet is under transfer to the USB, the AHB can get the next packet. If the AHB latency is large, then enough space must be allocated to buffer multiple packets.

The minimum amount of RAM required for Host periodic Transmit FIFO is the largest maximum packet size among all supported periodic OUT channels. If there is at least one High Bandwidth Isochronous OUT endpoint, then the space must be at least two times the maximum packet size of that channel.

Internal Register Storage Space Allocation

When operating in Buffer DMA mode, the DMA address register for each host channel (HCDMAx) is stored in the SPRAM. One location for each channel must be reserved for this.

When operating in Scatter/Gather DMA mode, four locations per channel must be reserved.

Table 16-19 FIFO Name - Data RAM Size

FIFO Name	Data RAM Size
Receive Data FIFO	rx_fifo_size
Non-periodic Transmit FIFO	tx_fifo_size[0]
IN Endpoint Transmit FIFO	tx_fifo_size[1]

With this information, the following registers must be programmed:

1. Receive FIFO Size Register (GRXFSIZ)
 - a) GRXFSIZ.RxFDep= rx_fifo_size;
2. Non-periodic Transmit FIFO Size Register (GNPTXFSIZ)
 - a) GNPTXFSIZ.NPTxFDe=tx_fifo_size[0];
 - b) GNPTXFSIZ.NPTxFStAddr = rx_fifo_size;
3. Host Periodic Transmit FIFO Size Register (HPTXFSIZ)
 - a) HPTXFSIZ.PTxFSize = tx_fifo_size[1];
 - b) HPTXFSIZ.PTxFStAddr= GNPTXFSIZ.NPTxFStAddr + tx_fifo_size[0];
4. The transmit FIFOs and receive FIFO must be flushed after RAM allocation for proper FIFO function.
 - a) GRSTCTL.TxFNum = 10_H
 - b) GRSTCTL.TxFFlush = 1_B
 - c) GRSTCTL.RxFFlush = 1_B
 - d) The application must wait until the TxFFlush bit and the RxFFlush bits are cleared before performing any operation on the core.

See also [Figure 16-70](#).

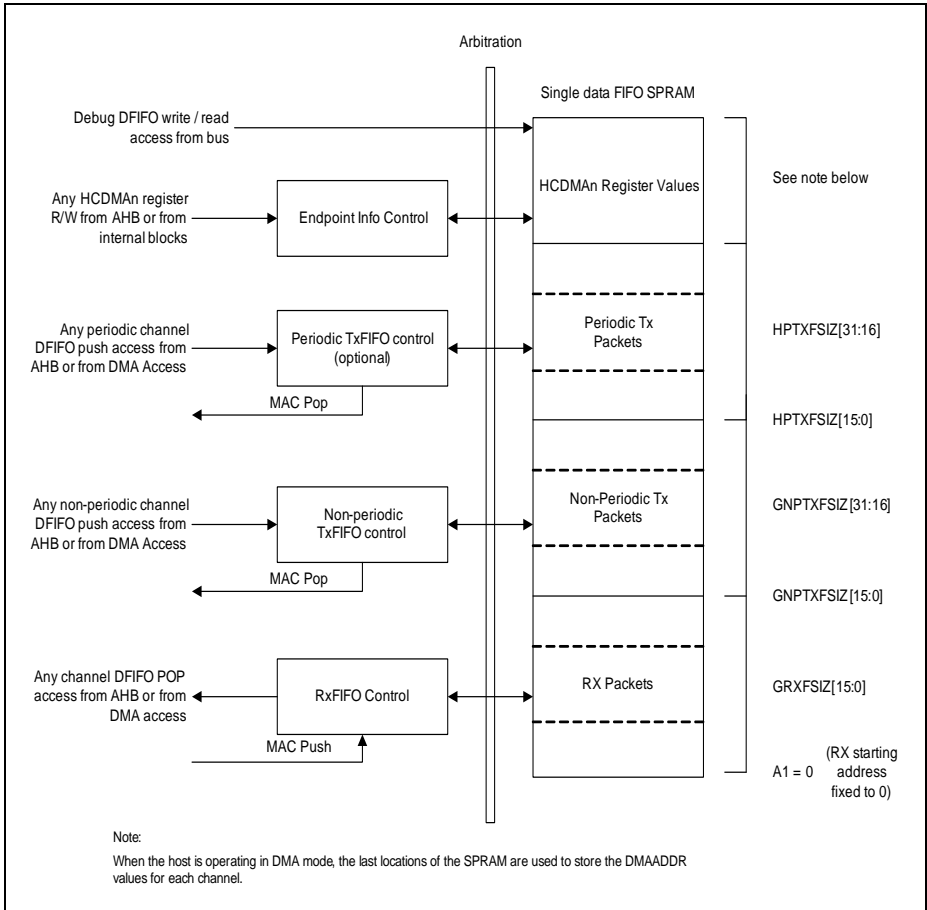


Figure 16-70 Host Mode FIFO Address Mapping

16.14.2 Dynamic FIFO Allocation

The application can change the RAM allocation for each FIFO during the operation of the core.

16.14.2.1 Dynamic FIFO Reallocation in Host Mode

In Host mode, before changing FIFO data RAM allocation, the application must determine the following:

- All channels are disabled
- All FIFOs are empty

Once these conditions are met, the application can reallocate FIFO data RAM as explained in **“Data FIFO RAM Allocation” on Page 16-223**.

After reallocating the FIFO data RAM, the application must flush all FIFOs in the core using the GRSTCTL.TxFIFO Flush and GRSTCTL.RxFIFO Flush fields. Flushing is required to reset the pointers in the FIFOs for proper FIFO operation after reallocation. For more information on flushing TxFIFO, see **Flushing TxFIFOs in the Core**.

16.14.2.2 Dynamic FIFO Reallocation in Device Mode

Dynamic FIFO re-allocation in device mode occurs when there is Power On Reset or a USB Reset.

In Device mode, before changing FIFO data RAM allocation,

1. The application must determine the following:
 - a) DIEPCTLn/DOEPCTLn.EPEna = 0_B
 - b) DIEPCTLn/DOEPCTLn.NAKSts = 1_B

If the bits are not set as above, follow the procedure in **“Transfer Stop Programming for OUT Endpoints” on Page 16-79** or **“Transfer Stop Programming for IN Endpoints” on Page 16-82** to ensure that all transfers on that endpoint are stopped.
2. Once these conditions are met, the application can reallocate FIFO data RAM as explained in **“Data FIFO RAM Allocation” on Page 16-223**.
3. Flush the TxFIFO in the core using the GRSTCTL.TxFIFO field. For more information on flushing TxFIFO, see **Flushing TxFIFOs in the Core**.

Note: The GlobalOUTNak process to disable OUT endpoints ensures that the Rx FIFO does not have any data, so an Rx FIFO flush is not required.

16.14.2.3 Flushing TxFIFOs in the Core

The application can flush all TxFIFOs in the core using GRSTCTL.TxFFish as follows:

1. Check that GINTSTS.GINNAkEff=0. If this bit is cleared then set DCTL.SGNPInNak=1.
NAK Effective interrupt = H indicating that the core is not reading from the FIFO.
2. Wait for GINTSTS.GINNAkEff=1, which indicates the NAK setting has taken effect to all IN endpoints.
3. Poll GRSTCTL.AHBIdle until it is 1.
AHBIdle = H indicates that the core is not writing anything to the FIFO.
4. Check that GRSTCTL.TxFFish =0. If it is 0, then write the TxFIFO number you want to flush to GRSTCTL.TxFNum.
5. Set GRSTCTL.TxFFish=1 and wait for it to clear.
6. Set the DCTL.GCNPIInNak bit.

16.15 Service Request Generation

The USB module provides a single service request output connected to an interrupt node in the Nested Vectored Interrupt Controller (NVIC)

Figure 16-71 displays the USB core interrupt hierarchy.

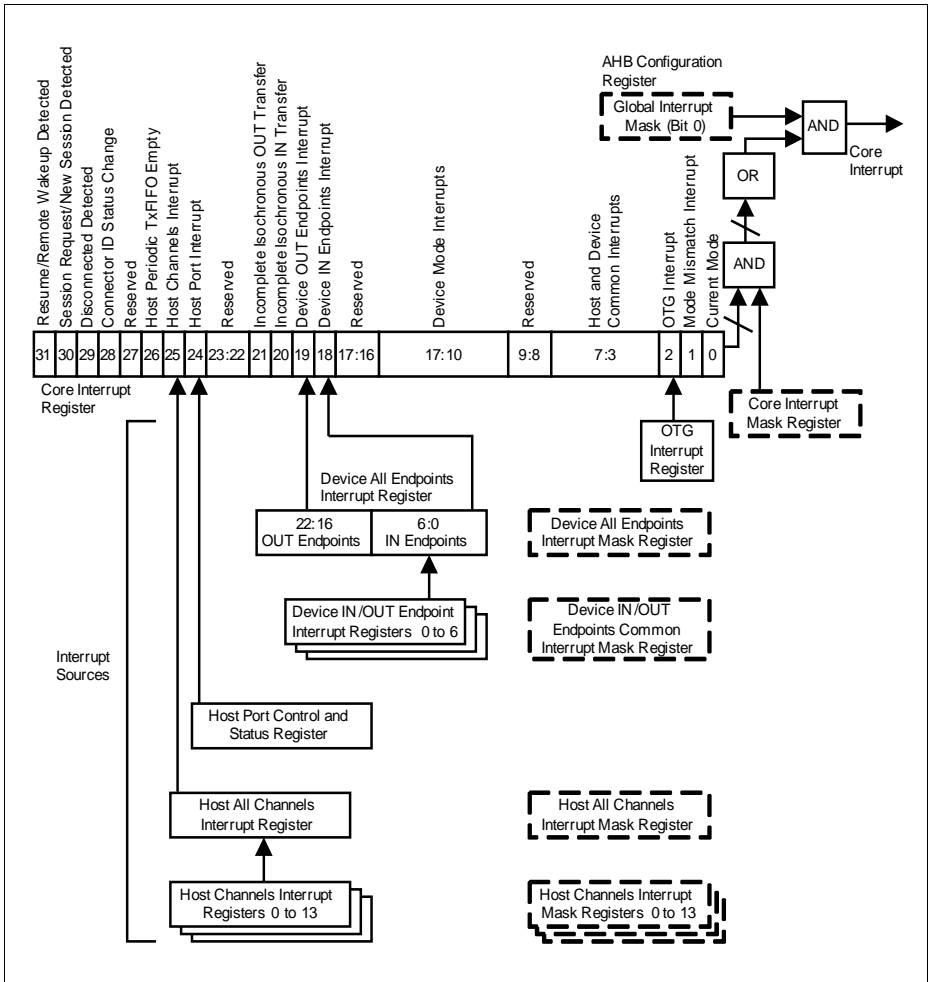


Figure 16-71 Interrupt Hierarchy

The Core Interrupt Handler

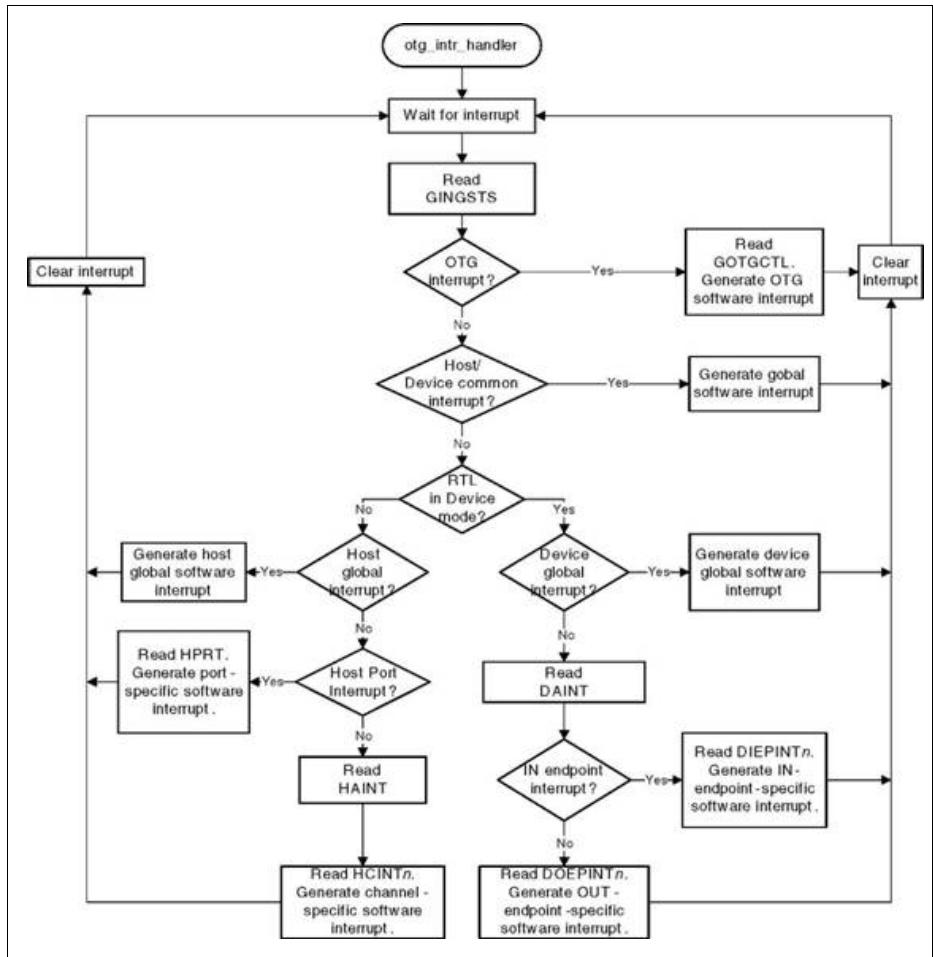


Figure 16-72 Core Interrupt Handler

16.16 Debug Behaviour

The USB module is not affected when the CPU enters HALT mode.

16.17 Power, Reset and Clock

When the USB module is programmed as a host, an external charge pump is required to drive the VBUS.

The module, including all registers, can be reset to its default state by a system reset or a software reset triggered through the setting of corresponding bits in PRSETx registers.

The module has the following input clocks:

- clk_ahbm: the module clock, which is also referred to as hclk in this chapter
- clk_usb: the 48 MHz PHY clock., which is also referred to as phy_clk in this chapter

In addition, the module internally generates:

- hclk_gated: hclk gated for power optimization

16.18 Initialization and System Dependencies

The USB core is held in reset after a start-up from a system or software reset. The USB PHY is also by default in the power-down state. Therefore, the application has to apply the following initialization sequence before programming the USB core:

- Release reset of USB core by writing a 1 to the USBRS bit in SCU_PRCLR2 register
- Enable the 48 MHz PHY clock by configuring the USB PLL in SCU, see clock control section in SCU chapter
- Remove the USB PHY from power-down by writing a 1 to the USBOTGEN and USBPHYPDQ bits in SCU_PWRSET register

16.19 Registers

Register Overview

The application controls the USB core by reading from and writing to the Control and Status Registers (CSRs) through the AHB Slave interface. These registers are 32 bits wide and the addresses are 32-bit block aligned.

Only the Core Global, Power and Clock Gating, Data FIFO Access, and Host Port registers can be accessed in both Host and Device modes. When the USB core is operating in one mode, either Device or Host, the application must not access registers from the other mode. If an illegal access occurs, a Mode Mismatch interrupt is generated and reflected in the Core Interrupt register (GINTSTS.ModeMis).

When the core switches from one mode to another, the registers in the new mode must be reprogrammed as they would be after a power-on reset.

The absolute register address is calculated by adding:

Module Base Address + Offset Address

Table 16-20 Registers Address Space

Module	Base Address	End Address	Note
USB0	5004 0000 _H	5007 FFFF _H	

Figure 16-73 shows the CSR address map. Host and Device mode registers occupy different addresses.

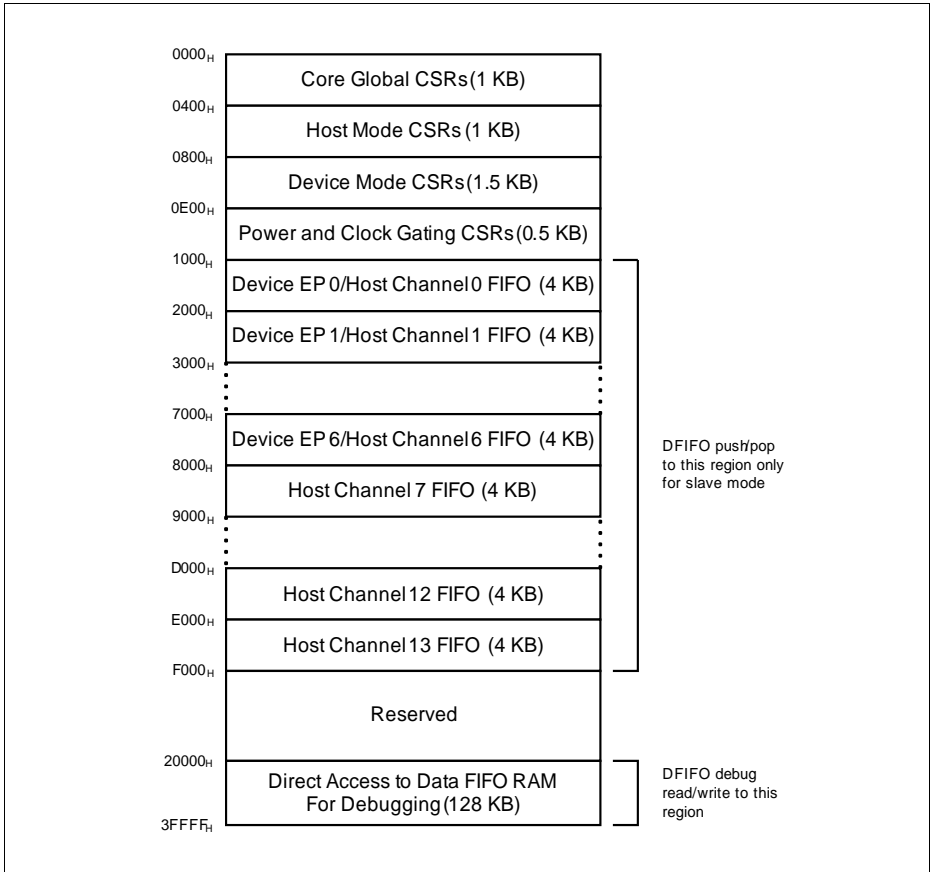


Figure 16-73 CSR Memory Map

The first letter of the register name is a prefix for the register type:

- G: Core Global
- H: Host mode
- D: Device mode

Note: FIFO size and FIFO depth are used interchangeably.

Table 16-21 Register Overview

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
USB Global Registers					
GOTGCTL	Control and Status Register	000 _H	U, PV	U, PV	Page 16-242
GOTGINT	OTG Interrupt Register	004 _H	U, PV	U, PV	Page 16-247
GAHBCFG	AHB Configuration Register	008 _H	U, PV	U, PV	Page 16-248
GUSBCFG	USB Configuration Register	00C _H	U, PV	U, PV	Page 16-250
GRSTCTL	Reset Register	010 _H	U, PV	U, PV	Page 16-253
GINTSTS	Interrupt Register	014 _H	U, PV	U, PV	Page 16-256
GINTMSK	Interrupt Mask Register	018 _H	U, PV	U, PV	Page 16-263
GRXSTSR	Receive Status Debug Read Register	01C _H	U, PV	U, PV	Page 16-266
GRXSTSP	Status Read and Pop Register	020 _H	U, PV	U, PV	Page 16-266
GRXFSIZ	Receive FIFO Size Register	024 _H	U, PV	U, PV	Page 16-269
GNPTXFSIZ	Non-Periodic Transmit FIFO Size Register	028 _H	U, PV	U, PV	Page 16-270
GNPTXSTS	Non-Periodic Transmit FIFO/Queue Status Register	02C _H	U, PV	U, PV	Page 16-271
Reserved	Reserved	030 _H -038 _H	nBE	nBE	
GUID	User ID Register	03C _H	U, PV	U, PV	Page 16-272
Reserved	Reserved	040 _H -058 _H	nBE	nBE	
GDFIFOCFG	DFIFO Software Config Register	05C _H	U, PV	U, PV	Page 16-273
Reserved	Reserved	060 _H -0FC _H	nBE	nBE	
HPTXFSIZ	Host Periodic Transmit FIFO Size Register	100 _H	U, PV	U, PV	Page 16-274
DIEPTXFn	Device IN Endpoint Transmit FIFO Size Register	104 _H -124 _H	U, PV	U, PV	Page 16-275

Table 16-21 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
Reserved	Reserved	128 _H - 3FF _H	nBE	nBE	
USB Host Mode Registers					
HCFG	Host Configuration Register	400 _H	U, PV	U, PV	Page 16-276
HFIR	Host Frame Interval Register	404 _H	U, PV	U, PV	Page 16-278
HFNUM	Host Frame Number/Frame Time Remaining Register	408 _H	U, PV	U, PV	Page 16-279
Reserved	Reserved	40C _H	nBE	nBE	
HPTXSTS	Host Periodic Transmit FIFO/Queue Status Register	410 _H	U, PV	U, PV	Page 16-280
HAINT	Host All Channels Interrupt Register	414 _H	U, PV	U, PV	Page 16-281
HAINTMSK	Host All Channels Interrupt Mask Register	418 _H	U, PV	U, PV	Page 16-282
HFLBADDR	Host Frame List Base Address Register	41C _H	U, PV	U, PV	Page 16-283
Reserved	Reserved	420 _H - 43C _H	nBE	nBE	
HPRT	Host Port Control and Status Register	440 _H	U, PV	U, PV	Page 16-283
Reserved	Reserved	444 _H - 4FC _H	nBE	nBE	
HCCHARx	Host Channel-n Characteristics Register	500 _H + n*20	U, PV	U, PV	Page 16-287
Reserved	Reserved	504 _H + n*20	nBE	nBE	
HCINTx	Host Channel-n Interrupt Register	508 _H + n*20	U, PV	U, PV	Page 16-289
HCINTMSKx	Host Channel-n Interrupt Mask Register	50C _H + n*20	U, PV	U, PV	Page 16-292
HCTSIZx	Host Channel-n Transfer Size Register	510 _H + n*20	U, PV	U, PV	Page 16-294

Table 16-21 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
HCDMAx	Host Channel-n DMA Address Register	514 _H + n*20	U, PV	U, PV	Page 16-297
Reserved	Reserved	518 _H + n*20	nBE	nBE	
HCDMABx	Host Channel-n DMA Buffer Address Register	51C _H + n*20	U, PV	U, PV	Page 16-300
Reserved	Reserved	780 _H -7FF _H	nBE	nBE	

USB Device Mode Registers

DCFG	Device Configuration Register	800 _H	U, PV	U, PV	Page 16-300
DCTL	Device Control Register	804 _H	U, PV	U, PV	Page 16-304
DSTS	Device Status Register	808 _H	U, PV	U, PV	Page 16-307
Reserved	Reserved	80C _H	nBE	nBE	
DIEPMSK	Device IN Endpoint Common Interrupt Mask Register	810 _H	U, PV	U, PV	Page 16-309
DOEPMSK	Device OUT Endpoint Common Interrupt Mask Register	814 _H	U, PV	U, PV	Page 16-310
DAINT	Device All Endpoints Interrupt Register	818 _H	U, PV	U, PV	Page 16-311
DAINTMSK	Device All Endpoints Interrupt Mask Register	81C _H	U, PV	U, PV	Page 16-312
Reserved	Reserved	820 _H -824 _H	nBE	nBE	
DVBUSDIS	Device VBUS Discharge Time Register	828 _H	U, PV	U, PV	Page 16-312
DVBUSPULSE	Device VBUS Pulsing Time Register	82C _H	U, PV	U, PV	Page 16-313
Reserved	Reserved	830 _H	nBE	nBE	
DIEPEMPMSK	Device IN Endpoint FIFO Empty Interrupt Mask Register	834 _H	U, PV	U, PV	Page 16-314

Table 16-21 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
Reserved	Reserved	838 _H -8FC _H	nBE	nBE	
DIEPCTL0	Device Control IN Endpoint 0 Control Register	900 _H	U, PV	U, PV	Page 16-314
DIEPCTLx	Device Endpoint n Control Register	900 _H + n*20 _H	U, PV	U, PV	Page 16-319
Reserved	Reserved	904 _H + n*20 _H	nBE	nBE	
DIEPINTx	Device Endpoint-n Interrupt Register	908 _H + n*20 _H	U, PV	U, PV	Page 16-329
Reserved	Reserved	90C _H + n*20 _H	nBE	nBE	
DIEPTSIZ0	Device Endpoint 0 Transfer Size Register	910 _H	U, PV	U, PV	Page 16-334
DIEPTSIZx	Device Endpoint-n Transfer Size Register	910 _H + n*20 _H	U, PV	U, PV	Page 16-336
DIEPDMAx	Device Endpoint-n DMA Address Register	914 _H + n*20 _H	U, PV	U, PV	Page 16-340
DTXFSTSx	Device IN Endpoint Transmit FIFO Status Register	918 _H + n*20 _H	U, PV	U, PV	Page 16-342
DIEPDMABx	Device Endpoint-n DMA Buffer Address Register	91C _H + n*20 _H	U, PV	U, PV	Page 16-341
Reserved	Reserved	9E0 _H -AFC _H	nBE	nBE	
DOEPCTL0	Device Control OUT Endpoint 0 Control Register	B00 _H	U, PV	U, PV	Page 16-317
DOEPCTLx	Device Endpoint-n Control Register	B00 _H + n*20 _H	U, PV	U, PV	Page 16-319
Reserved	Reserved	B04 _H + n*20 _H	nBE	nBE	
DOEPINTx	Device Endpoint-n Interrupt Register	B08 _H + n*20 _H	U, PV	U, PV	Page 16-329

Table 16-21 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
Reserved	Reserved	B0C _H + n*20 _H	nBE	nBE	
DOEPTSIZE0	Device Endpoint 0 Transfer Size Register	B10 _H	U, PV	U, PV	Page 16-334
DOEPTSIZEx	Device Endpoint-n Transfer Size Register	B10 _H + n*20 _H	U, PV	U, PV	Page 16-336
DOEPDMAx	Device Endpoint-n DMA Address Register	B14 _H + n*20 _H	U, PV	U, PV	Page 16-340
Reserved	Reserved	B18 _H + n*20 _H	nBE	nBE	
DOEPDMAx	Device Endpoint-n DMA Buffer Address Register	B1C _H + n*20 _H	U, PV	U, PV	Page 16-341
Reserved	Reserved	BE0 _H -DFC _H	nBE	nBE	

USB Power and Gating Register

PCGCR	Power and Clock Gating Control Register	E00 _H	U, PV	U, PV	Page 16-343
Reserved	Reserved	E04 _H -FFC _H	nBE	nBE	

Data FIFO (DFIFO) Access Register Map

These registers, available in both Host and Device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

Table 16-22 Data FIFO (DFIFO) Access Register Map

FIFO Access Register Section	Address Range	Access
Device IN Endpoint 0/Host OUT Channel 0: DFIFO Write Access Device OUT Endpoint 0/Host IN Channel 0: DFIFO Read Access	1000 _H -1FFC _H	WO/RO
Device IN Endpoint 1/Host OUT Channel 1: DFIFO Write Access Device OUT Endpoint 1/Host IN Channel 1: DFIFO Read Access	2000 _H -2FFC _H	WO/RO
...

Table 16-22 Data FIFO (DFIFO) Access Register Map (cont'd)

FIFO Access Register Section	Address Range	Access
Device IN Endpoint 6/Host OUT Channel 6: DFIFO Write Access Device OUT Endpoint 6/Host IN Channel 6: DFIFO Read Access	7000 _H - 7FFC _H	WO/RO
Host OUT Channel 7: DFIFO Write Access Host IN Channel 7: DFIFO Read Access	8000 _H - 8FFC _H	WO/RO
...
Host OUT Channel 13: DFIFO Write Access Host IN Channel 13: DFIFO Read Access	E000 _H - EFFF _H	WO/RO

Access Restriction

Note: The USB registers are accessible only through word accesses. Half-word and byte accesses on USB registers will not generate a bus error. Write to unused address space will not cause an error but be ignored.

16.19.1 Register Description

This section describes Core Global, Device Mode, Host Mode, and Power and Clock Gating CSRs.

Note: Always program Reserved fields with 0s. Treat read values from Reserved fields as unknowns (Xs).

Global Registers

These registers are available in both Host and Device modes, and do not need to be reprogrammed when switching between these modes.

Control and Status Register (GOTGCTL)

The OTG Control and Status register controls the behavior and reflects the status of the OTG function of the core.

Universal Serial Bus (USB)

GOTGCTL

Control and Status Register

(000_H)

Reset Value: 0001 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											OTG Ver	BSe sVld	ASe sVld	Dbn cTime	Conl DSts
r											rw	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				Dev HNP En	HstS etHN PEn	HNP Req	HstN egSc s	Bvali dOv Val	Bvali dOv En	Avali dOv Val	Avali dOv En	Vbva lidO vVal	Vbva lidO vEn	Ses Req	Ses Req Scs
r				rw	rw	rw	rh	rw	rw	rw	rw	rw	rw	rw	rh

Field	Bits	Type	Description
SesReqScs	0	rh	<p>Session Request Success</p> <p>The core sets this bit when a session request initiation is successful.</p> <p>0_B Session request failure</p> <p>1_B Session request success</p> <p>This bit is used in Device only.</p>
SesReq	1	rw	<p>Session Request</p> <p>The application sets this bit to initiate a session request on the USB. The application can clear this bit by writing a 0 when the Host Negotiation Success Status Change bit in the OTG Interrupt register (GOTGINT.HstNegSucStsChng) is set. The core clears this bit when the HstNegSucStsChng bit is cleared.</p> <p>Since the USB 1.1 Full-Speed Serial Transceiver interface is used to initiate the session request, the application must wait until the Vbus discharges to 0.2 V, after the B-Session Valid bit in this register (GOTGCTL.BSesVld) is cleared.</p> <p>0_B No session request</p> <p>1_B Session request</p> <p>This bit is used in Device only.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
VbvalidOvEn	2	rw	<p>VBUS Valid Override Enable</p> <p>This bit is used to enable/disable the software to override the vbus valid signal using the GOTGCTL.VbvalidOvVal.</p> <p>0_B Override is disabled and vbus valid signal from the PHY is used internally by the core.</p> <p>1_B Internally vbus valid received from the PHY is overridden with GOTGCTL.VbvalidOvVal.</p> <p>This bit is used in Host only.</p>
VbvalidOvVal	3	rw	<p>VBUS Valid Override Value</p> <p>This bit is used to set the override value for vbus valid signal when GOTGCTL.VbvalidOvEn is set.</p> <p>0_B vbusvalid value is 0_B when GOTGCTL.VbvalidOvEn = 1</p> <p>1_B vbusvalid value is 1_B when GOTGCTL.VbvalidOvEn = 1</p> <p>This bit is used in Host only.</p>
AvalidOvEn	4	rw	<p>A-Peripheral Session Valid Override Enable</p> <p>This bit is used to enable/disable the software to override the Avalid signal using the GOTGCTL.AvalidOvVal.</p> <p>0_B Override is disabled and Avalid signal from the PHY is used internally by the core.</p> <p>1_B Internally Avalid received from the PHY is overridden with GOTGCTL.AvalidOvVal.</p> <p>This bit is used in Host only.</p>
AvalidOvVal	5	rw	<p>A-Peripheral Session Valid Override Value</p> <p>This bit is used to set the override value for Avalid signal when GOTGCTL.AvalidOvEn is set.</p> <p>0_B Avalid value is 0_B when GOTGCTL.AvalidOvEn = 1</p> <p>1_B Avalid value is 1_B when GOTGCTL.AvalidOvEn = 1</p> <p>This bit is used in Host only.</p>
BvalidOvEn	6	rw	<p>B-Peripheral Session Valid Override Enable</p> <p>This bit is used to enable/disable the software to override the Bvalid signal using the GOTGCTL.BvalidOvVal.</p> <p>0_B Override is disabled and Bvalid signal from the PHY is used internally by the core.</p> <p>1_B Internally Bvalid received from the PHY is overridden with GOTGCTL.BvalidOvVal.</p> <p>This bit is used in Device only.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
BvalidOvVal	7	rw	<p>B-Peripheral Session Valid Override Value</p> <p>This bit is used to set the override value for Bvalid signal when GOTGCTL.BvalidOvEn is set.</p> <p>0_B Bvalid value is 0_B when GOTGCTL.BvalidOvEn = 1</p> <p>1_B Bvalid value is 1_B when GOTGCTL.BvalidOvEn = 1</p> <p>This bit is used in Device only.</p>
HstNegSs	8	rh	<p>Host Negotiation Success</p> <p>The core sets this bit when host negotiation is successful. The core clears this bit when the HNP Request (HNPREq) bit in this register is set.</p> <p>0_B Host negotiation failure</p> <p>1_B Host negotiation success</p> <p>This bit is used in Device only.</p>
HNPREq	9	rw	<p>HNP Request</p> <p>The application sets this bit to initiate an HNP request to the connected USB host. The application can clear this bit by writing a 0 when the Host Negotiation Success Status Change bit in the OTG Interrupt register (GOTGINT.HstNegSucStsChng) is set. The core clears this bit when the HstNegSucStsChng bit is cleared.</p> <p>0_B No HNP request</p> <p>1_B HNP request</p> <p>This bit is used in Device only.</p>
HstSetHNPEn	10	rw	<p>Host Set HNP Enable</p> <p>The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device.</p> <p>0_B Host Set HNP is not enabled</p> <p>1_B Host Set HNP is enabled</p> <p>This bit is used in Host only.</p>
DevHNPEn	11	rw	<p>Device HNP Enabled</p> <p>The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host.</p> <p>0_B HNP is not enabled in the application</p> <p>1_B HNP is enabled in the application</p> <p>This bit is used in Device only.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
ConIDSts	16	rh	<p>Connector ID Status</p> <p>Indicates the connector ID status on a connect event.</p> <p>0_B The USB core is in A-Device mode</p> <p>1_B The USB core is in B-Device mode</p>
DbncTime	17	rh	<p>Long/Short Debounce Time</p> <p>Indicates the debounce time of a detected connection.</p> <p>0_B Long debounce time, used for physical connections (100 ms + 2.5 μs)</p> <p>1_B Short debounce time, used for soft connections (2.5 μs)</p> <p>This bit is used in Host only.</p>
ASesVld	18	rh	<p>A-Session Valid</p> <p>Indicates the Host mode transceiver status.</p> <p>0_B A-session is not valid</p> <p>1_B A-session is valid</p> <p><i>Note: If the OTG features (such as SRP and HNP) are not enabled, the read reset value will be 1.</i></p> <p>This bit is used in Host only.</p>
BSesVld	19	rh	<p>B-Session Valid</p> <p>Indicates the Device mode transceiver status.</p> <p>0_B B-session is not valid.</p> <p>1_B B-session is valid.</p> <p>In OTG mode, this bit can be used to determine if the device is connected or disconnected.</p> <p><i>Note: If the OTG features (such as SRP and HNP) are not enabled, the read reset value will be 1.</i></p> <p>This bit is used in Device only.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
OTGVer	20	rw	OTG Version Indicates the OTG revision. 0_B OTG Version 1.3. In this version the core supports Data line pulsing and VBus pulsing for SRP. 1_B OTG Version 2.0. In this version the core supports only Data line pulsing for SRP. <i>Note: XMC4300 supports only OTG Version 1.3. Therefore, the OTGVer bit should always be written with 0.</i>
0	[15:12] , [31:21]	r	Reserved Read as 0; should be written with 0.

Interrupt Register (GOTGINT)

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

Note: All bits in this register are set only by hardware and cleared only by a software write of 1 to the bit.

GOTGINT

OTG Interrupt Register

(004_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												Dbn ceDo ne	ADe vTO UTC hg	HstN egDe t	0
												rwh	rwh	rwh	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						HstN egSu cSts Chn g	Ses Req Suc StsC hng	0					SesE ndD et	0	
						rwh	rwh						rwh	r	

Universal Serial Bus (USB)

Field	Bits	Type	Description
SesEndDet	2	rwh	Session End Detected The core sets this bit when the bvalid signal is deasserted. This bit is used in Device only.
SesReqSucStsChng	8	rwh	Session Request Success Status Change The core sets this bit on the success or failure of a session request. The application must read the Session Request Success bit in the OTG Control and Status register (GOTGCTL.SesReqScs) to check for success or failure.
HstNegSucStsChng	9	rwh	Host Negotiation Success Status Change The core sets this bit on the success or failure of a USB host negotiation request. The application must read the Host Negotiation Success bit of the OTG Control and Status register (GOTGCTL.HstNegScs) to check for success or failure.
HstNegDet	17	rwh	Host Negotiation Detected The core sets this bit when it detects a host negotiation request on the USB.
ADevTOU TChg	18	rwh	A-Device Timeout Change The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.
DbnceDone	19	rwh	Debounce Done The core sets this bit when the debounce is completed after the device connect. The application can start driving USB reset after seeing this interrupt. This bit is only valid when the HNP Capable or SRP Capable bit is set in the Core USB Configuration register (GUSBCFG.HNPCap or GUSBCFG.SRPCap, respectively). This bit is used in Host only.
0	[31:20], [16:10], [7:3], [1:0]	r	Reserved Read as 0; should be written with 0.

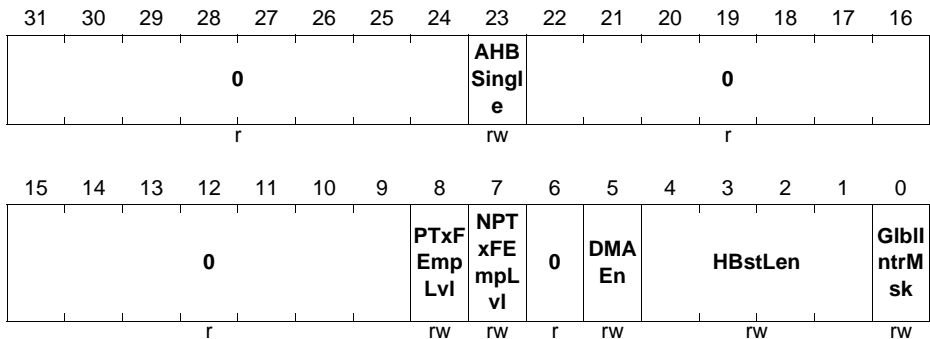
AHB Configuration Register (GAHBCFG)

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change

Universal Serial Bus (USB)

this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

GAHBCFG
AHB Configuration Register (008_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
GlibIntrMsk	0	rw	<p>Global Interrupt Mask</p> <p>The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit's setting, the interrupt status registers are updated by the core.</p> <p>0_B Mask the interrupt assertion to the application. 1_B Unmask the interrupt assertion to the application.</p>
HBstLen	[4:1]	rw	<p>Burst Length/Type</p> <p>This field is used in DMA mode to indicate the AHB Master burst type.</p> <p>0000_B Single 0001_B INCR 0011_B INCR4 0101_B INCR8 0111_B INCR16 Others: Reserved</p>
DMAEn	5	rw	<p>DMA Enable</p> <p>0_B Core operates in Slave mode 1_B Core operates in a DMA mode</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
NPTxFEmpLvl	7	rw	<p>Non-Periodic Tx FIFO Empty Level</p> <p>This bit indicates when IN endpoint Transmit FIFO empty interrupt (DIEPINTx.TxFEmp) is triggered.</p> <p>0_B DIEPINTx.TxFEmp interrupt indicates that the IN Endpoint Tx FIFO is half empty</p> <p>1_B DIEPINTx.TxFEmp interrupt indicates that the IN Endpoint Tx FIFO is completely empty</p> <p>This bit is used only in Device Slave mode.</p>
PTxFEmpLvl	8	rw	<p>Periodic Tx FIFO Empty Level</p> <p>Indicates when the Periodic Tx FIFO Empty Interrupt bit in the Core Interrupt register (GINTSTS.PTxFEmp) is triggered.</p> <p>0_B GINTSTS.PTxFEmp interrupt indicates that the Periodic Tx FIFO is half empty</p> <p>1_B GINTSTS.PTxFEmp interrupt indicates that the Periodic Tx FIFO is completely empty</p> <p>This bit is used only in Host Slave mode.</p>
AHBSingle	23	rw	<p>AHB Single Support</p> <p>This bit when programmed supports single transfers for the remaining data in a transfer when the core is operating in DMA mode.</p> <p>0_B The remaining data in a transfer is sent using INCR burst size. This is the default mode.</p> <p>1_B The remaining data in a transfer is sent using single burst size.</p>
0	[31:24], [22:9], 6	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

USB Configuration Register (GUSB_CFG)

This register can be used to configure the core after power-on or a changing to Host mode or Device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

Universal Serial Bus (USB)

GUSBCFG

USB Configuration Register

(00C_H)

Reset Value: 0000 1440_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTP	Forc eDev Mod e	Forc eHst Mod e	TxE nD el ay						0						Otg l 2CS el
rw	rw	rw	rw						r						rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0		USBTrdTim			HNP Cap	SRP Cap	0	PHY Sel		0			TOutCal	
	r		rw			rw	rw	r	r		r			rw	

Field	Bits	Type	Description
TOutCal	[2:0]	rw	<p>FS Timeout Calibration</p> <p>The number of PHY clocks that the application programs in this field is added to the full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.</p> <p>The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock is 0.25 bit times</p>
PHYSel	6	r	<p>USB 1.1 Full-Speed Serial Transceiver Select</p> <p>This bit is always read as 1 to indicate a full-speed transceiver is selected.</p> <p>0_B Reserved 1_B USB 1.1 full-speed serial transceiver</p>
SRPCap	8	rw	<p>SRP-Capable</p> <p>The application uses this bit to control the USB core SRP capabilities. If the core operates as a non-SRP-capable B-device, it cannot request the connected A-device (host) to activate VBUS and start a session.</p> <p>0_B SRP capability is not enabled. 1_B SRP capability is enabled.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
HNPCap	9	rw	<p>HNP-Capable</p> <p>The application uses this bit to control the USB core's HNP capabilities.</p> <p>0_B HNP capability is not enabled. 1_B HNP capability is enabled.</p>
USBTrdTim	[13:10]	rw	<p>USB Turnaround Time</p> <p>Sets the turnaround time in PHY clocks. Specifies the response time for a MAC request to the Packet FIFO Controller (PFC) to fetch data from the DFIFO (SPRAM).</p> <p><i>Note: USB turnaround time is critical for certification where long cables and 5-Hubs are used. See “Choosing the Value of GUSBCFG.USBTrdTim” on Page 16-83.</i></p> <p>This bit is used in Device Only.</p>
OtgI2CSel	16	rw	<p>UTMIFS Interface Select</p> <p>The application uses this bit to select the USB 1.1 Full-Speed interface.</p> <p>0_B UTMI USB 1.1 Full-Speed interface for OTG signals 1_B Reserved</p> <p>This bit should always be written with 0.</p>
TxEndDelay	28	rw	<p>Tx End Delay</p> <p>Writing a 1 to this bit enables the TxEndDelay timers in the core as per the section 4.1.5 on Opmode of the USB 2.0 Transceiver Macrocell Interface (UTMI) version 1.05.</p> <p>0_B Normal mode 1_B Introduce Tx end delay timers</p> <p>This bit is used in Device Only.</p>
ForceHstMode	29	rw	<p>Force Host Mode</p> <p>Writing a 1 to this bit forces the core to host mode irrespective of connected plug.</p> <p>0_B Normal Mode 1_B Force Host Mode</p> <p>After setting the force bit, the application must wait at least 25 ms before the change to take effect.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
ForceDev Mode	30	rw	Force Device Mode Writing a 1 to this bit forces the core to device mode irrespective of connected plug. 0 _B Normal Mode 1 _B Force Device Mode After setting the force bit, the application must wait at least 25 ms before the change to take effect.
CTP	31	rw	Corrupt Tx packet This bit is for debug purposes only. Never set this bit to 1.
0	[27:17] , [15:14] , 7, [5:3]	r	Reserved Read as 0; should be written with 0.

Reset Register (GRSTCTL)

The application uses this register to reset various hardware features inside the core.

GRSTCTL

Reset Register

(010_H)

Reset Value: 1000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHBI dle	DMA Req	0													
r	r	r													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		TxFNum						TxFF lsh	RxF Flsh	0	Frm Cntr Rst	0	CSft Rst		
r		rw						rwh	rwh	r	rwh	r	rwh		

Field	Bits	Type	Description
CSftRst	0	rwh	<p>Core Soft Reset</p> <p>Resets the hclk and phy_clock domains as follows:</p> <ul style="list-style-type: none"> • Clears the interrupts and all the CSR registers except the following register bits: <ul style="list-style-type: none"> – PCGCCTL.GateHclk – GUSBCFG.PHYSel – HCFG.FSLSPclkSel – DCFG.DevSpd – GGPIO • All module state machines (except the AHB Slave Unit) are reset to the IDLE state, and all the transmit FIFOs and the receive FIFO are flushed. • Any transactions on the AHB Master are terminated as soon as possible, after gracefully completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately. <p>The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit is cleared software must wait at least 3 PHY clocks before doing any access to the PHY domain (synchronization delay). Software must also must check that bit 31 of this register is 1 (AHB Master is IDLE) before starting any operation.</p> <p>Typically software reset is used during software development.</p>
FrmCntrRst	2	rwh	<p>Host Frame Counter Reset</p> <p>The application writes this bit to reset the frame number counter inside the core. When the frame counter is reset, the subsequent SOF sent out by the core has a frame number of 0.</p> <p>This bit is used in Host only.</p> <p>This bit is set only by software and cleared only by hardware.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
RxFFIsh	4	rwh	<p>RxFIFO Flush</p> <p>The application can flush the entire RxFIFO using this bit, but must first ensure that the core is not in the middle of a transaction.</p> <p>The application must only write to this bit after checking that the core is neither reading from the RxFIFO nor writing to the RxFIFO.</p> <p>The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.</p> <p>This bit is set only by software and cleared only by hardware.</p>
TxFFIsh	5	rwh	<p>TxFIFO Flush</p> <p>This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction. The application must write this bit only after checking that the core is neither writing to the TxFIFO nor reading from the TxFIFO. Verify using these registers:</p> <ul style="list-style-type: none"> • Read—NAK Effective Interrupt ensures the core is not reading from the FIFO • Write—GRSTCTL.AHBIdle ensures the core is not writing anything to the FIFO. <p>Flushing is normally recommended when FIFOs are re-configured. FIFO flushing is also recommended during device endpoint disable.</p> <p>The application must wait until the core clears this bit before performing any operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.</p> <p>This bit is set only by software and cleared only by hardware.</p>

Field	Bits	Type	Description
TxFNum	[10:6]	rw	<p>TxFIFO Number</p> <p>This is the FIFO number that must be flushed using the TxFIFO Flush bit. This field must not be changed until the core clears the TxFIFO Flush bit.</p> <p>00_H Non-periodic TxFIFO flush in Host mode or Tx FIFO 0 flush in device mode</p> <p>01_H Periodic TxFIFO flush in Host mode or Tx FIFO 1 flush in device mode</p> <p>02_H Tx FIFO 2 flush in device mode</p> <p>..._H ...</p> <p>0F_H Tx FIFO 15 flush in device mode</p> <p>10_H Flush all the transmit FIFOs in device or host mode.</p>
DMAReq	30	r	<p>DMA Request Signal</p> <p>Indicates that the DMA request is in progress. Used for debug.</p>
AHBIdle	31	r	<p>AHB Master Idle</p> <p>Indicates that the AHB Master State Machine is in the IDLE condition.</p>
0	[29:11] , 3, 1	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Interrupt Register (GINTSTS)

This register interrupts the application for system-level events in the current mode (Device mode or Host mode). It is shown in [Figure 16-71](#).

Some of the bits in this register are valid only in Host mode, while others are valid in Device mode only. This register also indicates the current mode.

Note: In the GINTSTS register, interrupt status bits with access type 'rwh' are set by hardware. To clear these bits, the application must write 1 into these bits.

The FIFO status interrupts are read only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

GINTSTS

Interrupt Register [HOSTMODE]

(014_H)

Reset Value: 1400 0020_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WkUpInt	SessReqInt	DiscOnnInt	ConlDStsChng	0	PTxFEmp	HChInt	PrtInt	0		incompIP			0		
rwh	rwh	rwh	rwh	r	rh	rh	rh	r		rwh			r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					0					1	RxFLvl	Sof	OTGInt	ModeMis	CurMod
					r					r	rh	rwh	rh	rwh	rh

Field	Bits	Type	Description
CurMod	0	rh	Current Mode of Operation Indicates the current mode. 0 _B Device mode 1 _B Host mode
ModeMis	1	rwh	Mode Mismatch Interrupt The core sets this bit when the application is trying to access: <ul style="list-style-type: none"> A Host mode register, when the core is operating in Device mode A Device mode register, when the core is operating in Host mode The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.
OTGInt	2	rh	OTG Interrupt The core sets this bit to indicate an OTG protocol event. The application must read the OTG Interrupt Status (GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the GOTGINT register to clear this bit.
Sof	3	rwh	Start of Frame In Host mode, the core sets this bit to indicate that an SOF is transmitted on the USB.

Universal Serial Bus (USB)

Field	Bits	Type	Description
RxFVLvl	4	rh	RxFIFO Non-Empty Indicates that there is at least one packet pending to be read from the RxFIFO.
incomplP	21	rwh	Incomplete Periodic Transfer In Host mode , the core sets this interrupt bit when there are incomplete periodic transactions still pending which are scheduled for the current frame.
PrtInt	24	rh	Host Port Interrupt The core sets this bit to indicate a change in port status of one of the USB core ports in Host mode. The application must read the Host Port Control and Status (HPRT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the Host Port Control and Status register to clear this bit.
HChInt	25	rh	Host Channels Interrupt The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in Host mode). The application must read the Host All Channels Interrupt (HAINT) register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding Host Channel-n Interrupt (HCINTx) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the HCINTx register to clear this bit.
PTxFEmp	26	rh	Periodic Tx FIFO Empty This interrupt is asserted when the Periodic Transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the Periodic Request Queue. The half or completely empty status is determined by the Periodic Tx FIFO Empty Level bit in the Core AHB Configuration register (GAHB_CFG.PTxFEmpLvl).
ConIDSts Chng	28	rwh	Connector ID Status Change This interrupt is asserted when there is a change in connector ID status.
DisconnInt	29	rwh	Disconnect Detected Interrupt This interrupt is asserted when a device disconnect is detected.

Universal Serial Bus (USB)

Field	Bits	Type	Description
SessReqInt	30	rwh	Session Request/New Session Detected Interrupt In Host mode, this interrupt is asserted when a session request is detected from the device. In Device mode, this interrupt is asserted when the Bvalid signal goes high.
WkUpInt	31	rwh	Resume/Remote Wakeup Detected Interrupt Wakeup Interrupt during Suspend state. <ul style="list-style-type: none"> • Device Mode - This interrupt is asserted only when Host Initiated Resume is detected on USB. • Host Mode - This interrupt is asserted only when Device Initiated Remote Wakeup is detected on USB.
1	5	r	Reserved Read as 1; should be written with 1.
0	27, [23:22], [20:6]	r	Reserved Read as 0; should be written with 0.

GINTSTS

Interrupt Register [DEVICEMODE] (014_H) **Reset Value: 1400 0020_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WkUpInt	SessReqInt	0	ConlDStsChng	0	1			0		incompISOOUT	incompISOIN	OEPInt	IEPInt		0
rwh	rwh	r	rwh	r	r			r		rwh	rwh	r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPF	ISOOutDrop	EnumDone	USB Rst	USB Susp	Eryl Susp	0		GOUTNakEff	GINNakEff	1	RxF Lvl	Sof	OTGI nt	Mod eMis	Cur Mod
rwh	rwh	rwh	rwh	rwh	rwh	r		rh	rh	r	rh	rwh	rh	rwh	rh

Universal Serial Bus (USB)

Field	Bits	Type	Description
CurMod	0	rh	<p>Current Mode of Operation Indicates the current mode.</p> <p>0_B Device mode 1_B Host mode</p>
ModeMis	1	rwh	<p>Mode Mismatch Interrupt The core sets this bit when the application is trying to access:</p> <ul style="list-style-type: none"> • A Host mode register, when the core is operating in Device mode • A Device mode register, when the core is operating in Host mode <p>The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.</p>
OTGInt	2	rh	<p>OTG Interrupt The core sets this bit to indicate an OTG protocol event. The application must read the OTG Interrupt Status (GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the GOTGINT register to clear this bit.</p>
Sof	3	rwh	<p>Start of Frame In Device mode, the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the Device Status register to get the current frame number.</p>
RxFLvl	4	rh	<p>RxFIFO Non-Empty Indicates that there is at least one packet pending to be read from the RxFIFO.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
GINNakEff	6	rh	<p>Global IN Non-Periodic NAK Effective</p> <p>Indicates that the Set Global Non-periodic IN NAK bit in the Device Control register (DCTL.SGNPInNak), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear Global Non-periodic IN NAK bit in the Device Control register (DCTL.CGNPInNak).</p> <p>This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.</p>
GOUTNakEff	7	rh	<p>Global OUT NAK Effective</p> <p>Indicates that the Set Global OUT NAK bit in the Device Control register (DCTL.SGOUTNak), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear Global OUT NAK bit in the Device Control register (DCTL.CGOUTNak).</p>
ErlySusp	10	rwh	<p>Early Suspend</p> <p>The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.</p>
USBSusp	11	rwh	<p>USB Suspend</p> <p>The core sets this bit to indicate that a suspend was detected on the USB. The core enters the Suspended state when there is no activity on the two USB data signals for an extended period of time.</p>
USBRst	12	rwh	<p>USB Reset</p> <p>The core sets this bit to indicate that a reset is detected on the USB.</p>
EnumDone	13	rwh	<p>Enumeration Done</p> <p>The core sets this bit to indicate that speed enumeration is complete. The application must read the Device Status (DSTS) register to obtain the enumerated speed.</p>
ISOOutDrop	14	rwh	<p>Isochronous OUT Packet Dropped Interrupt</p> <p>The core sets this bit when it fails to write an isochronous OUT packet into the RxFIFO because the RxFIFO does not have enough space to accommodate a maximum packet size packet for the isochronous OUT endpoint.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
EOPF	15	rwh	End of Periodic Frame Interrupt Indicates that the period specified in the Periodic Frame Interval field of the Device Configuration register (DCFG.PerFrInt) has been reached in the current frame.
IEPInt	18	r	IN Endpoints Interrupt The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in Device mode). The application must read the Device All Endpoints Interrupt (DAINT) register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding Device IN Endpoint-n Interrupt (DIEPINTx) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding DIEPINTx register to clear this bit.
OEPInt	19	r	OUT Endpoints Interrupt The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in Device mode). The application must read the Device All Endpoints Interrupt (DAINT) register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding Device OUT Endpoint-n Interrupt (DOEPINTx) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding DOEPINTx register to clear this bit.
incompISOIN	20	rwh	Incomplete Isochronous IN Transfer The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of Periodic Frame Interrupt (EOPF) bit in this register. <i>Note: This interrupt is not asserted in Scatter/Gather DMA mode.</i>

Universal Serial Bus (USB)

Field	Bits	Type	Description
incomplS OOUT	21	rwh	Incomplete Isochronous OUT Transfer In the Device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of Periodic Frame Interrupt (EOPF) bit in this register.
ConIDSts Chng	28	rwh	Connector ID Status Change This interrupt is asserted when there is a change in connector ID status.
SessReqI nt	30	rwh	Session Request/New Session Detected Interrupt In Host mode, this interrupt is asserted when a session request is detected from the device. In Device mode, this interrupt is asserted when the Bvalid signal goes high.
WkUpInt	31	rwh	Resume/Remote Wakeup Detected Interrupt Wakeup Interrupt during Suspend state. <ul style="list-style-type: none"> • Device Mode - This interrupt is asserted only when Host Initiated Resume is detected on USB. • Host Mode - This interrupt is asserted only when Device Initiated Remote Wakeup is detected on USB.
1	26, 5	r	Reserved Read as 1; should be written with 1.
0	29, 27, [25:22] , [17:16] , [9:8]	r	Reserved Read as 0; should be written with 0.

Interrupt Mask Register (GINTMSK)

This register works with the Interrupt Register (“[Interrupt Register \(GINTSTS\)](#)” on [Page 16-256](#)) to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the GINTSTS register bit corresponding to that interrupt is still set.

- Mask interrupt: 0_B
- Unmask interrupt: 1_B

Universal Serial Bus (USB)

GINTMSK

Interrupt Mask Register [HOSTMODE](018_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WkU plnt Msk	Sess Reql ntMsk	Disc onnl ntMsk	Conl DSts Chn gMsk	0	PTxF Emp Msk	HChl ntMsk	Prtln tMsk	0		inco mplP Msk			0		
rw	rw	rw	rw	r	rw	rw	rw	r		rw			r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					0						RxF LvIM sk	SofM sk	OTGI ntMsk	Mod eMis Msk	0
					r						rw	rw	rw	rw	r

Field	Bits	Type	Description
ModeMisMsk	1	rw	Mode Mismatch Interrupt Mask
OTGIntMsk	2	rw	OTG Interrupt Mask
SofMsk	3	rw	Start of Frame Mask
RxFLvIMsk	4	rw	Receive FIFO Non-Empty Mask
incomplPMsk	21	rw	Incomplete Periodic Transfer Mask
PrtIntMsk	24	rw	Host Port Interrupt Mask
HChIntMsk	25	rw	Host Channels Interrupt Mask
PTxFEmpMsk	26	rw	Periodic Tx FIFO Empty Mask
ConIDStsChngMsk	28	rw	Connector ID Status Change Mask
DisconnIntMsk	29	rw	Disconnect Detected Interrupt Mask
SessReqlntMsk	30	rw	Session Request/New Session Detected Interrupt Mask

Universal Serial Bus (USB)

Field	Bits	Type	Description
WkUpIntMask	31	rw	Resume/Remote Wakeup Detected Interrupt Mask
0	27, [23:22], [20:5], 0	r	Reserved Read as 0; should be written with 0.

GINTMSK

Interrupt Mask Register [DEVICEMODE] (018_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
WkUpIntMask	SessReqIntMask	0	ConlDStsChngMask			0				incompISOOUTMask	incompISOINMask	OEPIntMask	IEPIntMask		0	
rw	rw	r	rw			r				rw	rw	rw	rw		r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
EOPFMask	ISOOutDropMask	EnumDoneMask	USB RstMask	USB SuspMask	Early SuspMask		0	GOUTNakEffMask	GINNakEffMask	0	RxF LvlMask	SofMask	OTGIntMask	ModeMisMask	0	
rw	rw	rw	rw	rw	rw		r	rw	rw	r	rw	rw	rw	rw	rw	r

Field	Bits	Type	Description
ModeMisMask	1	rw	Mode Mismatch Interrupt Mask
OTGIntMask	2	rw	OTG Interrupt Mask
SofMask	3	rw	Start of Frame Mask
RxFLVIMask	4	rw	Receive FIFO Non-Empty Mask
GINNakEffMask	6	rw	Global Non-periodic IN NAK Effective Mask
GOUTNakEffMask	7	rw	Global OUT NAK Effective Mask
EarlySuspMask	10	rw	Early Suspend Mask

Universal Serial Bus (USB)

Field	Bits	Type	Description
USBSuspMsk	11	rw	USB Suspend Mask
USBRstMsk	12	rw	USB Reset Mask
EnumDoneMsk	13	rw	Enumeration Done Mask
ISOOutDropMsk	14	rw	Isochronous OUT Packet Dropped Interrupt Mask
EOPFMsk	15	rw	End of Periodic Frame Interrupt Mask Mode: Device only Reset: 0 _B
IEPIntMsk	18	rw	IN Endpoints Interrupt Mask
OEPIntMsk	19	rw	OUT Endpoints Interrupt Mask
incomplSOINMsk	20	rw	Incomplete Isochronous IN Transfer Mask
incomplISOOUTMsk	21	rw	Incomplete Isochronous OUT Transfer Mask
ConIDStsChngMsk	28	rw	Connector ID Status Change Mask
SessReqIntMsk	30	rw	Session Request/New Session Detected Interrupt Mask
WkUpIntMsk	31	rw	Resume/Remote Wakeup Detected Interrupt Mask
0	29, [27:22], [17:16], [9:8], 5, 0	r	Reserved Read as 0; should be written with 0.

Receive Status Debug Read/Status Read and Pop Registers (GRXSTSR/GRXSTSP)

A read to the Receive Status Debug Read register returns the contents of the top of the Receive FIFO. A read to the Receive Status Read and Pop register additionally pops the top data entry out of the Rx FIFO.

The receive status contents must be interpreted differently in Host and Device modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0000'0000_H. The application must only pop the Receive Status FIFO

Universal Serial Bus (USB)

when the Receive FIFO Non-Empty bit of the Core Interrupt register (GINTSTS.RxFLvl) is asserted.

Notes

1. Use of these fields vary based on whether the OTG core is functioning as a host or a device.
2. Do not read this register's reset value before configuring the core because the read value is "X".

Receive Status Debug Read/Status Read and Pop Registers in Host Mode

GRXSTSR

Receive Status Debug Read Register [HOSTMODE]

(01C_H)

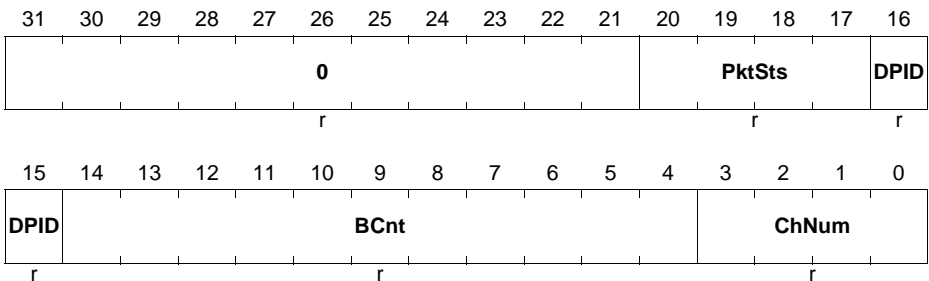
Reset Value: 0000 0000_H

GRXSTSP

Receive Status Read and Pop Register [HOSTMODE]

(020_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ChNum	[3:0]	r	Channel Number Indicates the channel number to which the current received packet belongs.
BCnt	[14:4]	r	Byte Count Indicates the byte count of the received IN data packet.
DPID	[16:15]	r	Data PID Indicates the Data PID of the received packet 00 _B DATA0 10 _B DATA1 01 _B DATA2 11 _B MDATA

Universal Serial Bus (USB)

Field	Bits	Type	Description
PktSts	[20:17]	r	Packet Status Indicates the status of the received packet 0010 _B IN data packet received 0011 _B IN transfer completed (triggers an interrupt) 0101 _B Data toggle error (triggers an interrupt) 0111 _B Channel halted (triggers an interrupt) Others: Reserved
0	[31:21]	r	Reserved Read as 0; should be written with 0.

Receive Status Debug Read/Status Read and Pop Registers in Device Mode (GRXSTSR/GRXSTSP)

GRXSTSR

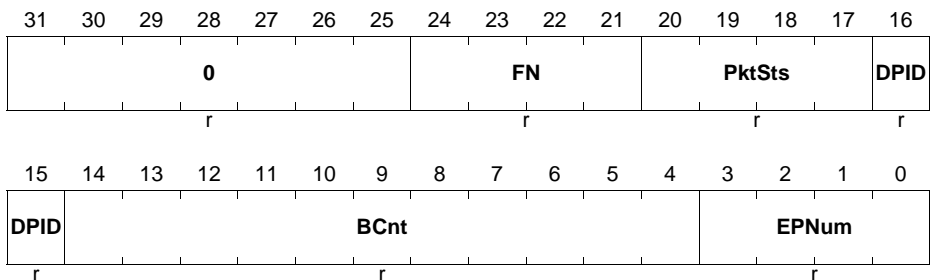
Receive Status Debug Read Register [DEVICEMODE]
(01C_H)

Reset Value: 0000 0000_H

GRXSTSP

Receive Status Read and Pop Register [DEVICEMODE]
(020_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
EPNum	[3:0]	r	Endpoint Number Indicates the endpoint number to which the current received packet belongs.
BCnt	[14:4]	r	Byte Count Indicates the byte count of the received data packet.

Universal Serial Bus (USB)

Field	Bits	Type	Description
DPID	[16:15]	r	Data PID Indicates the Data PID of the received OUT data packet 00 _B DATA0 10 _B DATA1 01 _B DATA2 11 _B MDATA
PktSts	[20:17]	r	Packet Status Indicates the status of the received packet 0001 _B Global OUT NAK (triggers an interrupt) 0010 _B OUT data packet received 0011 _B OUT transfer completed (triggers an interrupt) 0100 _B SETUP transaction completed (triggers an interrupt) 0110 _B SETUP data packet received Others: Reserved
FN	[24:21]	r	Frame Number This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.
0	[31:25]	r	Reserved Read as 0; should be written with 0.

Receive FIFO Size Register (GRXFSIZ)

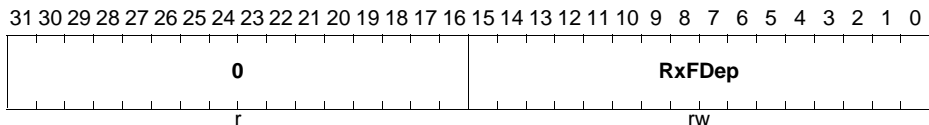
The application can program the RAM size that must be allocated to the Rx FIFO.

GRXFSIZ

Receive FIFO Size Register

(024_H)

Reset Value: 0000 011A_H



Field	Bits	Type	Description
RxFDep	[15:0]	rw	RxFIFO Depth This value is in terms of 32-bit words. <ul style="list-style-type: none"> • Minimum value is 16 • Maximum value is 282 Programmed values must not exceed the maximum value.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Non-Periodic Transmit FIFO Size Register (GNPTXFSIZ)

The application can program the RAM size and the memory start address for the Non-periodic Tx FIFO.

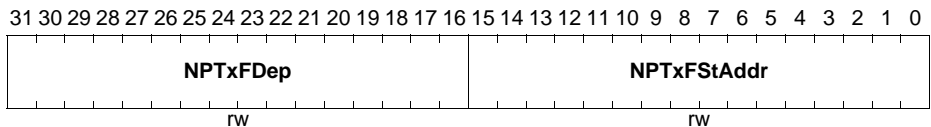
Note: The fields of this register change, depending on host or device mode.

GNPTXFSIZ

Non-Periodic Transmit FIFO Size Register [HOSTMODE]

(028_H)

Reset Value: 0010 011A_H



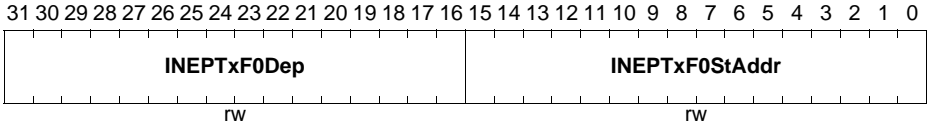
Field	Bits	Type	Description
NPTxFStAddr	[15:0]	rw	Non-periodic Transmit RAM Start Address This field contains the memory start address for Non-periodic Transmit FIFO RAM. Programmed values must not exceed the power-on value.
NPTxFDep	[31:16]	rw	Non-periodic Tx FIFO Depth This value is in terms of 32-bit words. <ul style="list-style-type: none"> • Minimum value is 16 • Maximum value is 16 Programmed values must not exceed the maximum value.

GNPTXFSIZ

Non-Periodic Transmit FIFO Size Register [DEVICEMODE]

(028_H)

Reset Value: 0010 011A_H



Field	Bits	Type	Description
INEPTxF0 StAddr	[15:0]	rw	IN Endpoint FIFO0 Transmit RAM Start Address This field contains the memory start address for IN Endpoint Transmit FIFO0. Programmed values must not exceed the power-on value.
INEPTxF0 Dep	[31:16]	rw	IN Endpoint Tx FIFO 0 Depth This value is in terms of 32-bit words. <ul style="list-style-type: none"> • Minimum value is 16 • Maximum value is 16 Programmed values must not exceed the maximum value.

Non-Periodic Transmit FIFO/Queue Status Register (GNPTXSTS)

This register is valid only in Host.

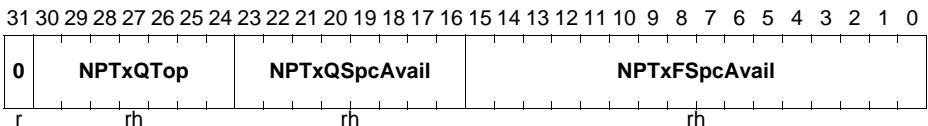
This read-only register contains the free space information for the Non-periodic Tx FIFO and the Non- periodic Transmit Request Queue.

GNPTXSTS

Non-Periodic Transmit FIFO/Queue Status Register(02C_H)

Reset Value:

0008 0010_H



Field	Bits	Type	Description
NPTxFSp cAvail	[15:0]	rh	<p>Non-periodic TxFIFO Space Avail</p> <p>Indicates the amount of free space available in the Non-periodic TxFIFO.</p> <p>Values are in terms of 32-bit words.</p> <p>0_H Non-periodic TxFIFO is full</p> <p>1_H 1 word available</p> <p>2_H 2 words available</p> <p>Others: Up to n words can be selected (0 < n < 16); selections greater than n are reserved</p>
NPTxQSp cAvail	[23:16]	rh	<p>Non-periodic Transmit Request Queue Space Available</p> <p>Indicates the amount of free space available in the Non-periodic Transmit RequestQueue. This queue holds both IN and OUT requests in Host mode.</p> <p>0_H Non-periodic Transmit Request Queue is full</p> <p>1_H 1 location available</p> <p>2_H 2 locations available</p> <p>Others: Up to n locations can be selected (0 < n < 8); selections greater than n are reserved</p>
NPTxQTo p	[30:24]	rh	<p>Top of the Non-periodic Transmit Request Queue</p> <p>Entry in the Non-periodic Tx Request Queue that is currently being processed by the MAC.</p> <ul style="list-style-type: none"> • Bits [30:27]: Channel/endpoint number • Bits [26:25]: <p>00_B IN/OUT token</p> <p>01_B Zero-length transmit packet (device IN/host OUT)</p> <p>10_B Reserved</p> <p>11_B Channel halt command</p> <ul style="list-style-type: none"> • Bit [24]: Terminate (last entry for selected channel/endpoint)
0	31	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

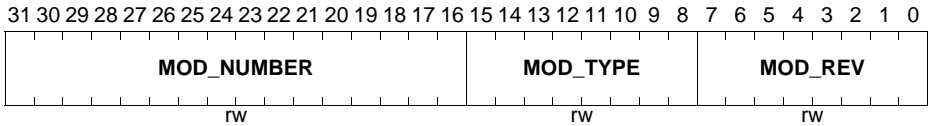
USB Module Identification Register (GUID)

This register contains the USB module version and revision and should not be overwritten by application.

GUID

USB Module Identification Register (03C_H)

Reset Value: 00AE C0XX_H



Field	Bits	Type	Description
MOD_REV	[7:0]	rw	Module Revision Indicates the revision number of the implementation. This information depends on the design step.
MOD_TYPE	[15:8]	rw	Module Type This internal marker is fixed to C0 _H .
MOD_NUMBER	[31:16]	rw	Module Number Indicates the module identification number.

Global DFIFO Software Config Register (GDFIFOCFG)

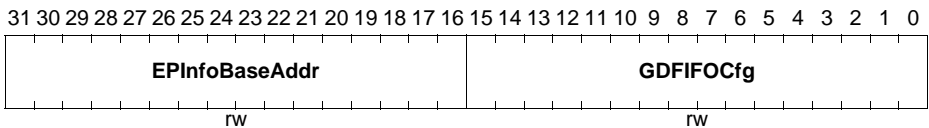
This register needs to be configured only when DMA mode is used. In slave (non-DMA) mode, it can be ignored.

Note: The reset value of the register does not represent the implemented FIFO RAM size.

GDFIFOCFG

Global DFIFO Software Config Register(05C_H)

Reset Value: 027A 02B2_H



Universal Serial Bus (USB)

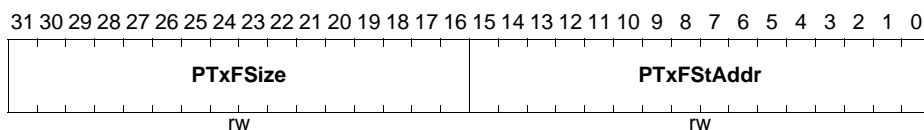
Field	Bits	Type	Description
GDFIFOCfg	[15:0]	rw	GDFIFOCfg This field is for dynamic programming of the DFIFO Size. This value takes effect only when the application programs a non zero value to this register. The value programmed must conform to the guidelines described in “ Data FIFO RAM Allocation ” on Page 16-223. The USB core does not have any corrective logic if the FIFO sizes are programmed incorrectly.
EPInfoBaseAddr	[31:16]	rw	EPInfoBaseAddr This field provides the start address of the RAM space allocated to store register information in DMA mode. See “ Data FIFO RAM Allocation ” on Page 16-223.

Host Periodic Transmit FIFO Size Register (HPTXFSIZ)

This register holds the size and the memory start address of the Periodic Tx FIFO.

HPTXFSIZ

Host Periodic Transmit FIFO Size Register(100_H) **Reset Value: 0100 012A_H**



Field	Bits	Type	Description
PTxFStAddr	[15:0]	rw	Host Periodic Tx FIFO Start Address The power-on reset value of this register is the sum of the Largest Rx Data FIFO Depth and Largest Non-periodic Tx Data FIFO Depth. Programmed values must not exceed the power-on value .
PTxFSize	[31:16]	rw	Host Periodic Tx FIFO Depth This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 256 Programmed values must not exceed the maximum value.

Device IN Endpoint Transmit FIFO Size Register (DIEPTXF_n)

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for instantiated IN endpoint FIFOs 1 to 6. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Note: The reset value of the register serves as a limit value and does not represent the implemented FIFO RAM size. The application must configure these registers in a way that the implemented FIFO RAM size is not exceeded.

DIEPTXF1

Device IN Endpoint 1 Transmit FIFO Size Register(104_H) Reset Value: 0100 012A_H

DIEPTXF2

Device IN Endpoint 2 Transmit FIFO Size Register(108_H) Reset Value: 0100 022A_H

DIEPTXF3

Device IN Endpoint 3 Transmit FIFO Size Register(10C_H) Reset Value: 0100 032A_H

DIEPTXF4

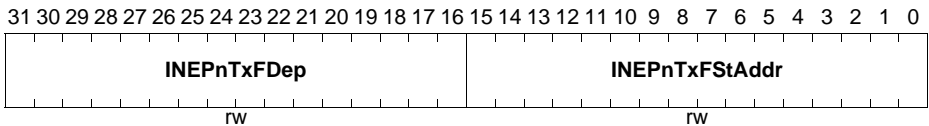
Device IN Endpoint 4 Transmit FIFO Size Register(110_H) Reset Value: 0100 042A_H

DIEPTXF5

Device IN Endpoint 5 Transmit FIFO Size Register(114_H) Reset Value: 0100 052A_H

DIEPTXF6

Device IN Endpoint 6 Transmit FIFO Size Register(118_H) Reset Value: 0100 062A_H



Field	Bits	Type	Description
INEPnTxFStAddr	[15:0]	rw	IN Endpoint FIFO_n Transmit RAM Start Address This field contains the memory start address for IN endpoint Transmit FIFO _n (1 < n ≤ 6). Programmed values must not exceed the reset value.
INEPnTxFDep	[31:16]	rw	IN Endpoint TxFIFO Depth This value is in terms of 32-bit words. <ul style="list-style-type: none"> • Minimum value is 16 • Maximum value is 256 Programmed values must not exceed the maximum value.

Host Mode Registers

These registers affect the operation of the core in the Host mode. Host mode registers must not be accessed in Device mode, as the results are undefined. Host Mode registers can be categorized as follows:

Host Configuration Register (HCFG)

This register configures the core after power-on. Do not make changes to this register after initializing the host.

HCFG

Host Configuration Register (400_H) **Reset Value: 0000 0200_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				PerSchedEna	FrListEn	DescDMA	0								
r				rw	rw	rw	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				1		0				FSLSSupp	FSLSPclkSel				
r				r		r				rw	rw				

Field	Bits	Type	Description
FSLSPclkSel	[1:0]	rw	FS PHY Clock Select 01 _B PHY clock is running at 48 MHz Others: Reserved
FSLSSupp	2	rw	FS-Only Support The application uses this bit to control the core's enumeration speed. Using this bit, the application can make the core enumerate as a FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming. 0 _B FS-only, connected device can supports also only FS. 1 _B FS-only, even if the connected device can support HS

Universal Serial Bus (USB)

Field	Bits	Type	Description
DescDMA	23	rw	<p>Enable Scatter/gather DMA in Host mode</p> <p>The application can set this bit during initialization to enable the Scatter/Gather DMA operation.</p> <p><i>Note: This bit must be modified only once after a reset. The following combinations are available for programming:</i></p> <ul style="list-style-type: none"> • GAHBCFG.DMAEn=0, HCFG.DescDMA=0 => Slave mode • GAHBCFG.DMAEn=0, HCFG.DescDMA=1 => Invalid • GAHBCFG.DMAEn=1, HCFG.DescDMA=0 => Buffered DMA mode • GAHBCFG.DMAEn=1, HCFG.DescDMA=1 => Scatter/Gather DMA mode <p>In non Scatter/Gather DMA mode, this bit is reserved.</p>
FrListEn	[25:24]	rw	<p>Frame List Entries</p> <p>This field is valid only in Scatter/Gather DMA mode. The value in the register specifies the number of entries in the Frame list.</p> <p>00_B 8 Entries 01_B 16 Entries 10_B 32 Entries 11_B 64 Entries</p> <p>In modes other than Scatter/Gather DMA mode, these bits are reserved.</p>
PerSchedEna	26	rw	<p>Enable Periodic Scheduling</p> <p>Applicable in Scatter/Gather DMA mode only. Enables periodic scheduling within the core. Initially, the bit is reset. The core will not process any periodic channels. As soon as this bit is set, the core will get ready to start scheduling periodic channels and sets HCFG.PerSchedStat. The setting of HCFG.PerSchedStat indicates the core has enabled periodic scheduling. Once HCFG.PerSchedEna is set, the application is not supposed to again reset the bit unless HCFG.PerSchedStat is set. As soon as this bit is reset, the core will get ready to stop scheduling periodic channels and resets HCFG.PerSchedStat.</p> <p>In non Scatter/Gather DMA mode, this bit is reserved.</p>

Universal Serial Bus (USB)

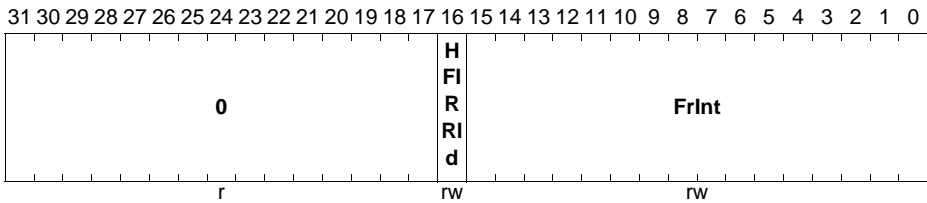
Field	Bits	Type	Description
1	9	r	Reserved Read as 1; should be written with 1.
0	[31:27] , [22:10] , [8:3]	r	Reserved Read as 0; should be written with 0.

Host Frame Interval Register (HFIR)

This register stores the frame interval information for the current speed to which the USB core has enumerated.

HFIR

Host Frame Interval Register (404_H) **Reset Value: 0000 EA60_H**



Universal Serial Bus (USB)

Field	Bits	Type	Description
FrInt	[15:0]	rw	<p>Frame Interval</p> <p>The value that the application programs to this field specifies the interval between two consecutive SOFs. This field contains the number of PHY clocks that constitute the required frame interval. The default value set in this field for a FS operation when the PHY clock frequency is 60 MHz. The application can write a value to this register only after the Port Enable bit of the Host Port Control and Status register (HPRT.PrtEnaPort) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS PHY Clock Select field of the Host Configuration register (HCFG.FSLSPckSel).</p> <p>Do not change the value of this field after the initial configuration.</p> <ul style="list-style-type: none"> 1 ms * (PHY clock frequency for FS)
HFIRIdCtrl	16	rw	<p>Reload Control</p> <p>This bit allows dynamic reloading of the HFIR register during runtime.</p> <p>0_B HFIR cannot be reloaded dynamically 1_B HFIR can be dynamically reloaded during runtime</p> <p>This bit needs to be programmed during the initial configuration and its value must not be changed during runtime.</p>
0	[31:17]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Host Frame Number/Frame Time Remaining Register (HFNUM)

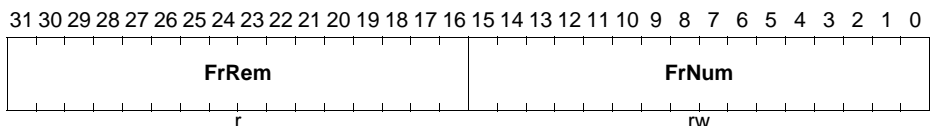
This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.

HFNUM

Host Frame Number/Frame Time Remaining Register

(408_H)

Reset Value: 0000 3FFF_H



Universal Serial Bus (USB)

Field	Bits	Type	Description
FrNum	[15:0]	rw	Frame Number This field increments when a new SOF is transmitted on the USB, and is reset to 0 _H when it reaches 3FFF _H .
FrRem	[31:16]	r	Frame Time Remaining Indicates the amount of time remaining in the current frame, in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame Interval register and a new SOF is transmitted on the USB.

Host Periodic Transmit FIFO/Queue Status Register (HPTXSTS)

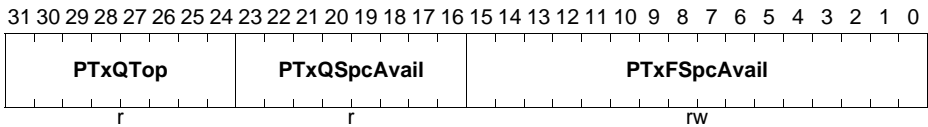
This read-only register contains the free space information for the Periodic Tx FIFO and the Periodic Transmit Request Queue.

HPTXSTS

**Host Periodic Transmit FIFO/
Queue Status Register**

(410_H)

Reset Value: 0008 0100_H



Field	Bits	Type	Description
PTxFSpC Avail	[15:0]	rw	<p>Periodic Transmit Data FIFO Space Available Indicates the number of free locations available to be written to in the Periodic Tx FIFO. Values are in terms of 32-bit words</p> <p>0_H Periodic Tx FIFO is full 1_H 1 word available 2_H 2 words available Others: Up to n words can be selected (0 < n < 256); selections greater than n are reserved</p>
PTxQSpC Avail	[23:16]	r	<p>Periodic Transmit Request Queue Space Available Indicates the number of free locations available to be written in the Periodic Transmit Request Queue. This queue holds both IN and OUT requests.</p> <p>0_H Periodic Transmit Request Queue is full 1_H 1 location available 2_H 2 locations available Others: Up to n locations can be selected (0 < n < 8); selections greater than n are reserved</p>
PTxQTop	[31:24]	r	<p>Top of the Periodic Transmit Request Queue This indicates the entry in the Periodic Tx Request Queue that is currently being processed by the MAC. This register is used for debugging.</p> <p>Bit [31]: Odd/Even frame 0_B send in even frame 1_B send in odd frame Bits [30:27]: Channel/endpoint number Bits [26:25]: Type 00_B IN/OUT 01_B Zero-length packet 10_B Reserved 11_B Disable channel command Bit [24]: Terminate (last entry for the selected channel/endpoint)</p>

Host All Channels Interrupt Register (HAINT)

When a significant event occurs on a channel, the Host All Channels Interrupt register interrupts the application using the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt). This is shown in [Figure 16-71 “Interrupt Hierarchy” on](#)

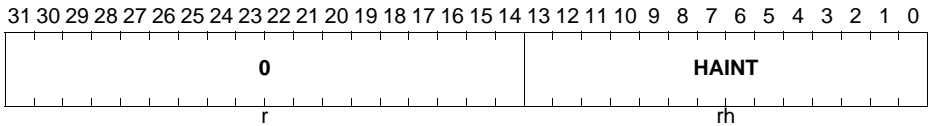
Universal Serial Bus (USB)

Page 16-232. There is one interrupt bit per channel, up to a maximum of 14 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Host Channel- n Interrupt register.

HAINT

Host All Channels Interrupt Register (414_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
HAINT	[13:0]	rw	Channel Interrupts One bit per channel: Bit 0 for Channel 0, bit 13 for Channel 13
0	[31:14]	r	Reserved Read as 0; should be written with 0.

Host All Channels Interrupt Mask Register (HAINTMSK)

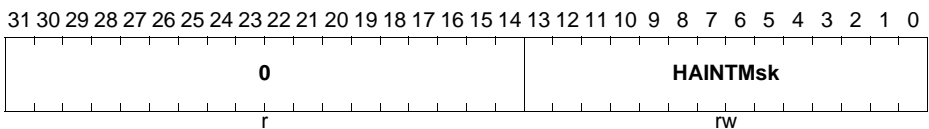
The Host All Channel Interrupt Mask register works with the Host All Channel Interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 14 bits.

- Mask interrupt: 0_B
- Unmask interrupt: 1_B

HAINTMSK

Host All Channels Interrupt Mask Register(418_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
HAINTMs k	[13:0]	rw	Channel Interrupt Mask One bit per channel: Bit 0 for channel 0, bit 13 for channel 13
0	[31:14]	r	Reserved Read as 0; should be written with 0.

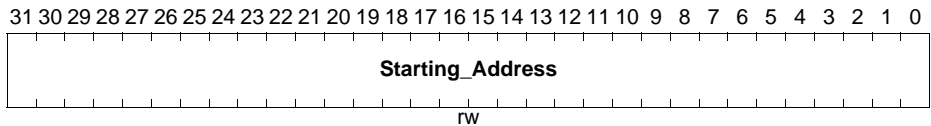
Host Frame List Base Address Register (HFLBADDR)

This register holds the starting address of the Frame list information. It is present only in case of Scatter/Gather DMA and used only for Isochronous and Interrupt Channels. The register is implemented in RAM.

HFLBADDR

Host Frame List Base Address Register(41C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
Starting Address	[31:0]	rw	Starting Address The starting address of the Frame list.

Host Port Control and Status Register (HPRT)

This register is available only in Host mode. Currently, the OTG Host supports only one port.

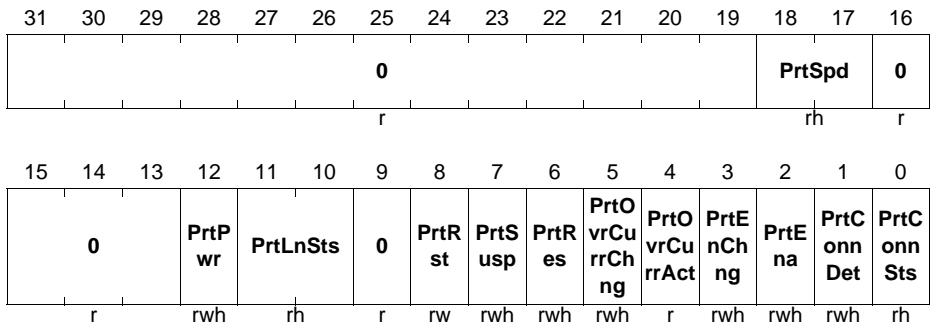
A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in [Figure 16-71 “Interrupt Hierarchy” on Page 16-232](#). The bits PrtOvrCurrChng, PrtEnChng and PrtConnDet in this register can trigger an interrupt to the application through the Host Port Interrupt bit of the Core Interrupt register (GINTSTS.PrtInt). On a Port Interrupt, the application must read this register and clear the bit that caused the interrupt. For these bits, the application must write a 1 to the bit to clear the interrupt.

Universal Serial Bus (USB)

HPRT

Host Port Control and Status Register(440_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PrtConnSts	0	rh	<p>Port Connect Status</p> <p>0_B No device is attached to the port. 1_B A device is attached to the port.</p>
PrtConnDet	1	rwh	<p>Port Connect Detected</p> <p>The core sets this bit when a device connection is detected to trigger an interrupt to the application using the Host Port Interrupt bit of the Core Interrupt register (GINTSTS.PrtInt).</p> <p>This bit is set only by hardware and cleared only by a software write of 1 to the bit.</p>
PrtEna	2	rwh	<p>Port Enable</p> <p>A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application.</p> <p>0_B Port disabled 1_B Port enabled</p>
PrtEnChng	3	rwh	<p>Port Enable/Disable Change</p> <p>The core sets this bit when the status of the Port Enable bit [2] of this register changes.</p> <p>This bit is set only by hardware and cleared only by a software write of 1 to the bit.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
PrtOvrCurAct	4	r	<p>Port Overcurrent Active</p> <p>Indicates the overcurrent condition of the port.</p> <p>0_B No overcurrent condition 1_B Overcurrent condition</p>
PrtOvrCurChng	5	rwh	<p>Port Overcurrent Change</p> <p>The core sets this bit when the status of the Port Overcurrent Active bit (bit 4) in this register changes. This bit is set only by hardware and cleared only by a software write of 1 to the bit.</p>
PrtRes	6	rwh	<p>Port Resume</p> <p>The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.</p> <p>If the core detects a USB remote wakeup sequence, as indicated by the Port Resume/Remote Wakeup Detected Interrupt bit of the Core Interrupt register (GINTSTS.WkUpInt), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.</p> <p>0_B No resume driven 1_B Resume driven</p> <p>This bit can be set and cleared by both hardware and software.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
PrtSusp	7	rwh	<p>Port Suspend</p> <p>The application sets this bit to put this port in Suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the Port Clock Stop bit, which asserts the suspend input pin of the PHY. The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wakeup signal is detected or the application sets the Port Reset bit or Port Resume bit in this register or the Resume/Remote WakeupDetected Interrupt bit or Disconnect Detected Interrupt bit in the Core Interrupt register (GINTSTS.WkUpInt or GINTSTS.DisconnInt, respectively).</p> <p>0_B Port not in Suspend mode 1_B Port in Suspend mode</p> <p>This bit is set only by software and cleared only by hardware.</p>
PrtRst	8	rw	<p>Port Reset</p> <p>When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.</p> <p>0_B Port not in reset 1_B Port in reset</p> <p>To start a reset on the port, the application must leave this bit set for at least the minimum duration mentioned below, as specified in the USB 2.0 specification, Section 7.1.7.5. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.</p> <ul style="list-style-type: none"> • Full speed: 10 ms
PrtLnSts	[11:10]	rh	<p>Port Line Status</p> <p>Indicates the current logic level USB data lines</p> <p>Bit [10]: Logic level of D+ Bit [11]: Logic level of D-</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
PrtPwr	12	rwh	Port Power The application uses this field to control power to this port, and the core can clear this bit on an over current condition. 0 _B Power off 1 _B Power on This bit is set only by software and can be cleared by hardware or a software write of 0 to the bit.
PrtSpd	[18:17]	rh	Port Speed Indicates the speed of the device attached to this port. 01 _B Full speed Other values are reserved.
0	[31:19] , [16:13] , 9	r	Reserved Read as 0; should be written with 0.

Host Channel-n Characteristics Register (HCCHARx)

HCCHARx (x=0-13)

Host Channel-x Characteristics Register(500_H + x*20_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ChE na	ChDi s	Odd Frm	DevAddr						MC_EC	EPTyep		0			
rwh	rwh	rw	rw						rw	rw		r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPDi r		EPNum				MPS									
rw		rw				rw									

Universal Serial Bus (USB)

Field	Bits	Type	Description
MPS	[10:0]	rw	Maximum Packet Size Indicates the maximum packet size of the associated endpoint.
EPNum	[14:11]	rw	Endpoint Number Indicates the endpoint number on the device serving as the data source or sink.
EPDir	15	rw	Endpoint Direction Indicates whether the transaction is IN or OUT. 0 _B OUT 1 _B IN
EPTYPE	[19:18]	rw	Endpoint Type Indicates the transfer type selected. 00 _B Control 01 _B Isochronous 10 _B Bulk 11 _B Interrupt
MC_EC	[21:20]	rw	Multi Count / Error Count This field indicates to the host the number of transactions that must be executed per frame for this periodic endpoint. For non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. 00 _B Reserved. This field yields undefined results. 01 _B 1 transaction 10 _B 2 transactions to be issued for this endpoint per frame 11 _B 3 transactions to be issued for this endpoint per frame
DevAddr	[28:22]	rw	Device Address This field selects the specific device serving as the data source or sink.

Universal Serial Bus (USB)

Field	Bits	Type	Description
OddFrm	29	rw	<p>Odd Frame</p> <p>This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd frame. This field is applicable for only periodic (isochronous and interrupt) transactions.</p> <p>0_B Even frame 1_B Odd frame</p> <p>This field is not applicable for Scatter/Gather DMA mode and need not be programmed by the application and is ignored by the core.</p>
ChDis	30	rwh	<p>Channel Disable</p> <p>The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled.</p> <p>This bit can be set by software and can be cleared by both hardware and software.</p>
ChEna	31	rwh	<p>Channel Enable</p> <p>0_B Scatter/Gather mode enabled: Indicates that the descriptor structure is not yet ready. Scatter/Gather mode disabled: Channel disabled</p> <p>1_B Scatter/Gather mode enabled: Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor. Scatter/Gather mode disabled: Channel enabled</p> <p>This bit is set only by software and cleared only by hardware.</p>
0	[17:16]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Host Channel-n Interrupt Register (HCINTx)

This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in [Figure 16-71 “Interrupt Hierarchy” on Page 16-232](#). The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

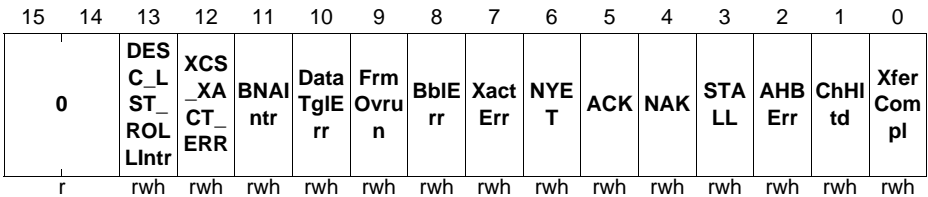
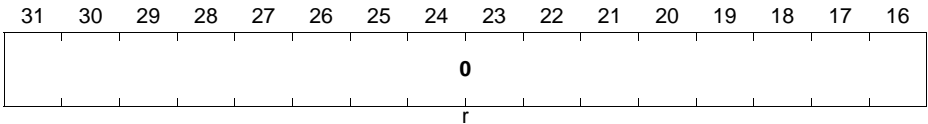
Universal Serial Bus (USB)

Note: All bits of the access type 'rwh' in this register are set only by hardware and cleared only by a software write of 1 to the bits.

HCINTx (x=0-13)

Host Channel-x Interrupt Register(508_H + x*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
XferCompl	0	rwh	<p>Transfer Completed</p> <p>For Scatter/Gather DMA mode, it indicates that current descriptor processing got completed with IOC bit set in its descriptor.</p> <p>In non Scatter/Gather DMA mode, it indicates that Transfer completed normally without any errors.</p>
ChHltd	1	rwh	<p>Channel Halted</p> <p>In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer.</p> <p>In Scatter/Gather DMA mode, this indicates that transfer completed due to any of the following</p> <ul style="list-style-type: none"> • EOL being set in descriptor • AHB error • Excessive transaction errors • In response to disable request by the application • Babble • Stall • Buffer Not Available (BNA)

Universal Serial Bus (USB)

Field	Bits	Type	Description
AHBErr	2	rwh	AHB Error This is generated only in DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address.
STALL	3	rwh	STALL Response Received Interrupt In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
NAK	4	rwh	NAK Response Received Interrupt In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
ACK	5	rwh	ACK Response Received/Transmitted Interrupt In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
NYET	6	rwh	NYET Response Received Interrupt In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
XactErr	7	rwh	Transaction Error Indicates one of the following errors occurred on the USB. <ul style="list-style-type: none"> • CRC check failure • Timeout • Bit stuff error • False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
BblErr	8	rwh	Babble Error In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
FrmOvrnun	9	rwh	Frame Overrun In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core
DataTglErr	10	rwh	Data Toggle Error In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.

Universal Serial Bus (USB)

Field	Bits	Type	Description
BNAIntr	11	rwh	BNA (Buffer Not Available) Interrupt This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. For non Scatter/Gather DMA mode, this bit is reserved.
XCS_XACT_ERR	12	rwh	Excessive Transaction Error This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels. For non Scatter/Gather DMA mode, this bit is reserved.
DESC_LS_T_ROLLIntr	13	rwh	Descriptor rollover interrupt This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. For non Scatter/Gather DMA mode, this bit is reserved.
0	[31:14]	r	Reserved Read as 0; should be written with 0.

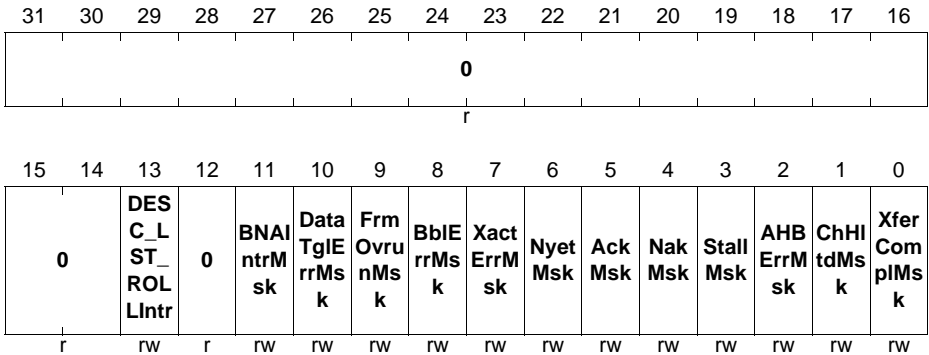
Host Channel-n Interrupt Mask Register (HCINTMSKx)

This register reflects the mask for each channel status described in register HCINTx.

- Mask interrupt: 0_B
- Unmask interrupt: 1_B

HCINTMSKx (x=0-13)

Host Channel-x Interrupt Mask Register(50C_H + x*20_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
XferCompIMsk	0	rw	Transfer Completed Mask
ChHltdMsk	1	rw	Channel Halted Mask
AHBErrMsk	2	rw	AHB Error Mask
StallMsk	3	rw	STALL Response Received Interrupt Mask This bit is not applicable in Scatter/Gather DMA mode.
NakMsk	4	rw	NAK Response Received Interrupt Mask This bit is not applicable in Scatter/Gather DMA mode.
AckMsk	5	rw	ACK Response Received/Transmitted Interrupt Mask This bit is not applicable in Scatter/Gather DMA mode.
NyetMsk	6	rw	NYET Response Received Interrupt Mask This bit is not applicable in Scatter/Gather DMA mode.
XactErrMsk	7	rw	Transaction Error Mask This bit is not applicable in Scatter/Gather DMA mode
BblErrMsk	8	rw	Babble Error Mask This bit is not applicable in Scatter/Gather DMA mode.
FrmOvrnMsk	9	rw	Frame Overrun Mask This bit is not applicable in Scatter/Gather DMA mode.

Universal Serial Bus (USB)

Field	Bits	Type	Description
DataToggleErrorMsk	10	rw	Data Toggle Error Mask This bit is not applicable in Scatter/Gather DMA mode.
BNAIntrMsk	11	rw	BNA (Buffer Not Available) Interrupt mask register This bit is valid only when Scatter/Gather DMA mode is enabled. In non Scatter/Gather DMA mode, this bit is reserved
DESC_LST_ROLLIntrMsk	13	rw	Descriptor rollover interrupt Mask register This bit is valid only when Scatter/Gather DMA mode is enabled. In non Scatter/Gather DMA mode, this bit is reserved.
0	[31:14], 12	r	Reserved Read as 0; should be written with 0.

Host Channel-n Transfer Size Register (HCTSIZx)

The HCTSIZx register description depends on the selected DMA mode.

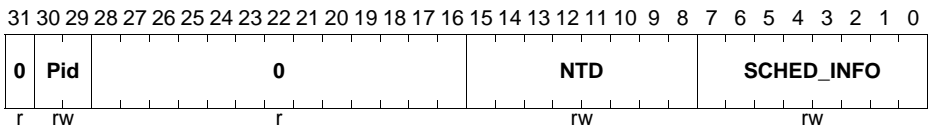
In **Scatter/Gather mode**, HCTSIZx is defined as follows:

HCTSIZx (x=0-13)

Host Channel-x Transfer Size Register [SCATGATHER]

$$(510_H + x*20_H)$$

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SCHED_I NFO	[7:0]	rw	Schedule information This field should be written with 1111'1111 _B for a FS Host.
NTD	[15:8]	rw	Number of Transfer Descriptors (Non Isochronous) This value is in terms of number of descriptors. Maximum number of descriptor that can be present in the list is 64. The values can be from 0 to 63. 0 _D : 1 descriptor ... _D : ... 63 _D : 64 descriptors This field indicates the total number of descriptors present in that list. The core will wrap around after servicing NTD number of descriptors for that list. (Isochronous) This field indicates the number of descriptors present in that list frame The possible values are 1 _D : 2 descriptors 3 _D : 4 descriptors 7 _D : 8 descriptors 15 _D : 16 descriptors 31 _D : 32 descriptors 63 _D : 64 descriptors
Pid	[30:29]	rw	PID The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer. 00 _B DATA0 01 _B DATA2 10 _B DATA1 11 _B MDATA (non-control)
0	31, [28:16]	r	Reserved Read as 0; should be written with 0.

Host Channel-n Transfer Size Register (HCTSIZx)

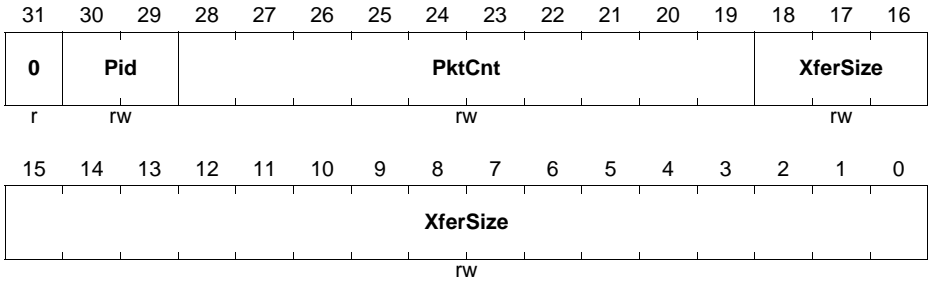
In **Buffer DMA Mode**, HCTSIZx is defined as follows:

HCTSIZx (x=0-13)

Host Channel-x Transfer Size Register [BUFFERMODE]

(510_H + x*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
XferSize	[18:0]	rw	<p>Transfer Size</p> <p>For an OUT, this field is the number of data bytes the host sends during the transfer.</p> <p>For an IN, this field is the buffer size that the application has reserved for the transfer.</p> <p>The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).</p>
PktCnt	[28:19]	rw	<p>Packet Count</p> <p>This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).</p> <p>The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
Pid	[30:29]	rw	PID The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer. 00 _B DATA0 01 _B DATA2 10 _B DATA1 11 _B MDATA (non-control)/SETUP (control)
0	31	r	Reserved Read as 0; should be written with 0.

Host Channel-n DMA Address Register (HCDMAx)

This register is used by the OTG host in DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned. The HCDMAx register description depends on the selected DMA mode.

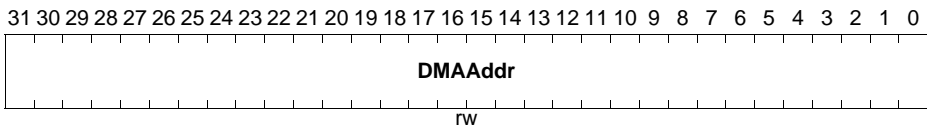
In **Buffer DMA Mode**, HCDMAx is defined as follows:

HCDMAx (x=0-13)

Host Channel-x DMA Address Register [BUFFERMODE]

$$(514_H + x*20_H)$$

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DMAAddr	[31:0]	rw	DMA Address This field holds the start address in the external memory from which the data for the endpoint must be fetched or to which it must be stored. This register is incremented on every AHB transaction.

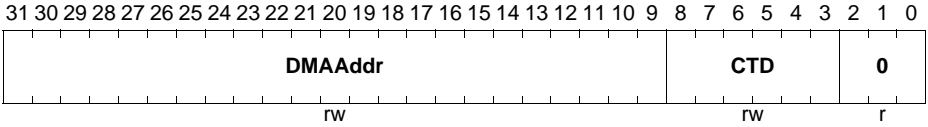
In **Scatter/Gather DMA Mode**, HCDMAx is defined as follows:

HCDMA_x (x=0-13)

Host Channel-x DMA Address Register [SCATGATHER]

(514_H + x*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CTD	[8:3]	rw	<p>Current Transfer Desc:</p> <p>Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0_D 1 descriptor ..._D ... 63_D 64 descriptors This field indicates the current descriptor processed in the list. This field is updated both by application and the core. For example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8 bytes*5=) 40 to DMAAddr.</p> <p>Isochronous: For isochronous transfers, the bits are [N-1:3]. CTD for isochronous is based on the current frame value. Need to be set to zero by application.</p>
DMAAddr	[31:9]	rw	<p>DMA Address</p> <p>Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value.</p> <p>Isochronous: For isochronous transfers, the bits are [31:N]. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as listed below: nTD=1 => N=4 nTD=3 => N=5 nTD=7 => N=6 nTD=15 => N=7 nTD=31 => N=8 nTD=63 => N=9</p> <p><i>Note: For Scatter/Gather DMA mode, this address is the start of the page address where the descriptor list is located.</i></p>
0	[2:0]	r	<p>Reserved Read as 0; should be written with 0.</p>

Host Channel-n DMA Buffer Address Register (HCDMABx)

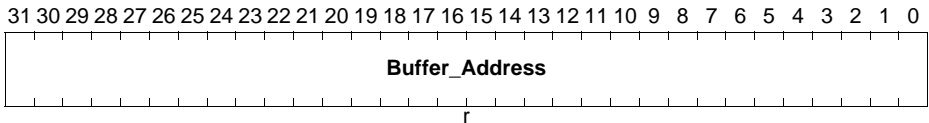
This register is present only in case of Scatter/Gather DMA. It is implemented in RAM. This register holds the current buffer address.

HCDMABx (x=0-13)

Host Channel-x DMA Buffer Address Register(51C_H + x*20_H)

Reset Value:

0000 0000_H



Field	Bits	Type	Description
Buffer_Ad dress	[31:0]	r	Buffer Address Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. Reset: "X" if not programmed as the register is in SPRAM

Device Mode Registers

These registers are visible only in Device mode and must not be accessed in Host mode, as the results are unknown. Some of them affect all the endpoints uniformly, while others affect only a specific endpoint. Device Mode registers fall into two categories:

Device Logical IN Endpoint-Specific Registers

One set of endpoint registers is instantiated per logical endpoint. A logical endpoint is unidirectional: it can be either IN or OUT. To represent a bidirectional endpoint, two logical endpoints are required, one for the IN direction and the other for the OUT direction. This is also true for control endpoints.

The registers and register fields described in this section can pertain to IN or OUT endpoints, or both, or specific endpoint types as noted.

Device Configuration Register (DCFG)

This register configures the core in Device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

Universal Serial Bus (USB)

DCFG

Device Configuration Register (800_H) **Reset Value: 0820 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				1	0	PerSchInt vI	Desc DMA	0	1	0				0	
r				r	r	rw	rw	r	r	r				r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		PerFrInt		DevAddr						0	NZSt sOU THShk	DevSpd			
r		rw		rw						r	rw	rw			

Field	Bits	Type	Description
DevSpd	[1:0]	rw	<p>Device Speed</p> <p>Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected. See “Device Initialization” on Page 16-72 for details.</p> <p>00_B Reserved 01_B Reserved 10_B Reserved 11_B Full speed (USB 1.1 transceiver clock is 48 MHz)</p>
NZStsOU THShk	2	rw	<p>Non-Zero-Length Status OUT Handshake</p> <p>The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's Status stage.</p> <p>1_B Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application. 0_B Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the Device Endpoint Control register.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
DevAddr	[10:4]	rw	<p>Device Address</p> <p>The application must program this field after every SetAddress control command.</p>
PerFrInt	[12:11]	rw	<p>Periodic Frame Interval</p> <p>Indicates the time within a frame at which the application must be notified using the End Of Periodic Frame Interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.</p> <p>00_B 80% of the frame interval 01_B 85% 10_B 90% 11_B 95%</p>
DescDMA	23	rw	<p>Enable Scatter/Gather DMA in Device mode.</p> <p>The application can set this bit during initialization to enable the Scatter/Gather DMA operation.</p> <p><i>Note: This bit must be modified only once after a reset.</i></p> <p>The following combinations are available for programming:</p> <ul style="list-style-type: none"> • GAHBCFG.DMAEn=0,DCFG.DescDMA=0 => Slave mode • GAHBCFG.DMAEn=0,DCFG.DescDMA=1 => Invalid • GAHBCFG.DMAEn=1 ,DCFG.DescDMA=0 => Buffered DMA mode • GAHBCFG.DMAEn=1 ,DCFG.DescDMA=1 => Scatter/Gather DMA mode

Universal Serial Bus (USB)

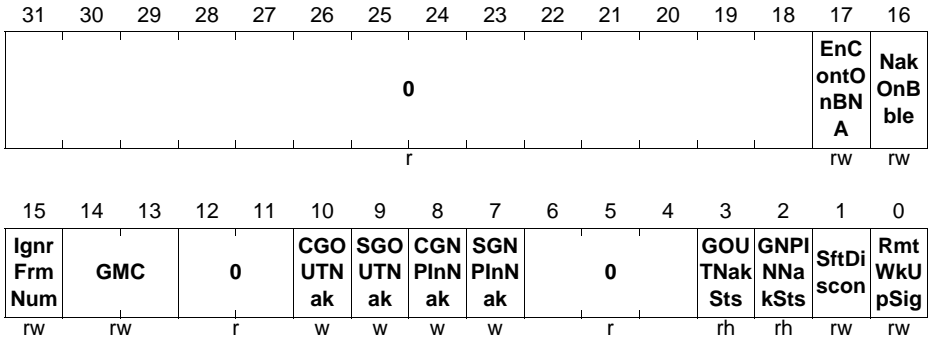
Field	Bits	Type	Description
PerSchIntvl	[25:24]	rw	<p>Periodic Scheduling Interval PerSchIntvl must be programmed only for Scatter/Gather DMA mode.</p> <p>Description: This field specifies the amount of time the Internal DMA engine must allocate for fetching periodic IN endpoint data. Based on the number of periodic endpoints, this value must be specified as 25, 50 or 75% of frame.</p> <ul style="list-style-type: none"> When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data. When no periodic endpoints are active, then the internal DMA engine services non-periodic endpoints, ignoring this field. After the specified time within a frame, the DMA switches to fetching for non-periodic endpoints. <p>00_B 25% of frame. 01_B 50% of frame. 10_B 75% of frame. 11_B Reserved.</p>
1	27, 21	r	<p>Reserved Read as 1; should be written with 1.</p>
0	[31:28], 26, 22, [20:18], [17:13], 3	r	<p>Reserved Read as 0; should be written with 0.</p>

Universal Serial Bus (USB)

Device Control Register (DCTL)

DCTL

Device Control Register (804_H) Reset Value: 0000 0002_H



Field	Bits	Type	Description
RmtWkUp Sig	0	rw	<p>Remote Wakeup Signaling</p> <p>When the application sets this bit, the core initiates remote signaling to wake the USB host. The application must set this bit to instruct the core to exit the Suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1 to 15 ms after setting it.</p>
SftDiscon	1	rw	<p>Soft Disconnect</p> <p>The application uses this bit to signal the USB core to do a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit. The minimum duration for which the core must keep this bit set is specified in Table 16-23.</p> <p>0_B Normal operation. When this bit is cleared after a soft disconnect, the core drives a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.</p> <p>1_B The core drives a device disconnect event to the USB host.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
GNPINNa kSts	2	rh	<p>Global Non-periodic IN NAK Status</p> <p>0_B A handshake is sent out based on the data availability in the transmit FIFO.</p> <p>1_B A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.</p>
GOUTNa kSts	3	rh	<p>Global OUT NAK Status</p> <p>0_B A handshake is sent based on the FIFO Status and the NAK and STALL bit settings.</p> <p>1_B No data is written to the Rx FIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.</p>
SGNPInN ak	7	w	<p>Set Global Non-periodic IN NAK</p> <p>A write to this field sets the Global Non-periodic IN NAK. The application uses this bit to send a NAK handshake on all non-periodic IN endpoints. The core can also set this bit when a timeout condition is detected on a non-periodic endpoint in shared FIFO operation. The application must set this bit only after making sure that the Global IN NAK Effective bit in the Core Interrupt Register (GINTSTS.GINNakEff) is cleared.</p>
CGNPInN ak	8	w	<p>Clear Global Non-periodic IN NAK</p> <p>A write to this field clears the Global Non-periodic IN NAK.</p>
SGOUTNa k	9	w	<p>Set Global OUT NAK</p> <p>A write to this field sets the Global OUT NAK. The application uses this bit to send a NAK handshake on all OUT endpoints. The application must set the this bit only after making sure that the Global OUT NAK Effective bit in the Core Interrupt Register (GINTSTS.GOUTNakEff) is cleared.</p>
CGOUTN ak	10	w	<p>Clear Global OUT NAK</p> <p>A write to this field clears the Global OUT NAK.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
GMC	[14:13]	rw	<p>Global Multi Count</p> <p>GMC must be programmed only once after initialization. Applicable only for Scatter/Gather DMA mode. This indicates the number of packets to be serviced for that end point before moving to the next end point. It is only for non-periodic end points.</p> <p>00_B Invalid. 01_B 1 packet. 10_B 2 packets. 11_B 3 packets.</p> <p>When Scatter/Gather DMA mode is disabled, this field is reserved. and reads 00_B.</p>
IgnrFrmNum	15	rw	<p>Ignore frame number for isochronous endpoints in case of Scatter/Gather DMA</p> <p><i>Note: When this bit is enabled, there must be only one packet per descriptor.</i></p> <p>In Scatter/Gather DMA mode, when this bit is enabled, the packets are not flushed when an ISOC IN token is received for an elapsed frame.</p> <p>When Scatter/Gather DMA mode is disabled, this field is used by the application to enable periodic transfer interrupt. The application can program periodic endpoint transfers for multiple frames.</p> <p>0_B Scatter/Gather enabled: The core transmits the packets only in the frame number in which they are intended to be transmitted. Scatter/Gather disabled: Periodic transfer interrupt feature is disabled; the application must program transfers for periodic endpoints every frame</p> <p>1_B Scatter/Gather enabled: The core ignores the frame number, sending packets immediately as the packets are ready. Scatter/Gather disabled: Periodic transfer interrupt feature is enabled; the application can program transfers for multiple frames for periodic endpoints.</p> <p>In non-Scatter/Gather DMA mode, the application receives transfer complete interrupt after transfers for multiple frames are completed.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
NakOnBble	16	rw	Set NAK automatically on babble The core sets NAK automatically for the endpoint on which babble is received.
EnContOnBNA	17	rw	Enable continue on BNA This bit enables the core to continue on BNA for Bulk OUT endpoints. With this feature enabled, when a Bulk OUT endpoint receives a BNA interrupt the core starts processing the descriptor that caused the BNA interrupt after the endpoint re-enables the endpoint. 0 _B After receiving BNA interrupt, the core disables the endpoint. When the endpoint is re-enabled by the application, the core starts processing from the DOEPDMA descriptor. 1 _B After receiving BNA interrupt, the core disables the endpoint. When the endpoint is re-enabled by the application, the core starts processing from the descriptor that received the BNA interrupt.
0	[31:18] , [12:11] , [6:4]	r	Reserved Read as 0; should be written with 0.

Table 16-23 lists the minimum duration under various conditions for which the Soft Disconnect (SftDiscon) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add some extra delay to the specified minimum duration.

Table 16-23 Minimum Duration for Soft Disconnect

Operating Speed	Device State	Minimum Duration
Full speed	Suspended	1 ms + 2.5 μ s
Full speed	Idle	2.5 μ s
Full speed	Not Idle or Suspended (Performing transactions)	2.5 μ s

Device Status Register (DSTS)

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from Device All Interrupts (DAINT) register.

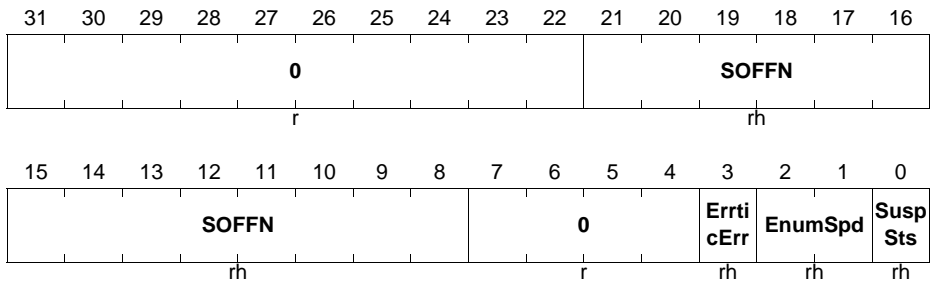
Universal Serial Bus (USB)

DSTS

Device Status Register

(808_H)

Reset Value: 0000 0002_H



Field	Bits	Type	Description
SuspSts	0	rh	<p>Suspend Status</p> <p>In Device mode, this bit is set as long as a Suspend condition is detected on the USB. The core enters the Suspended state when there is no activity on the two USB data signals for an extended period of time. The core comes out of the suspend:</p> <ul style="list-style-type: none"> • When there is any activity on the two USB data signals • When the application writes to the Remote Wakeup Signaling bit in the Device Control register (DCTL.RmtWkUpSig)
EnumSpd	[2:1]	rh	<p>Enumerated Speed</p> <p>Indicates the speed at which the USB core has come up after speed detection through a chirp sequence.</p> <p>00_B Reserved 01_B Reserved 10_B Reserved 11_B Full speed (PHY clock is running at 48 MHz)</p>
ErrticErr	3	rh	<p>Erratic Error</p> <p>The core sets this bit to report any erratic errors. Due to erratic errors, the USB core goes into Suspended state and an interrupt is generated to the application with Early Suspend bit of the Core Interrupt register (GINTSTS.ErlySusp). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.</p>

Field	Bits	Type	Description
SOFFN	[21:8]	rh	Frame Number of the Received SOF When the core is operating at full speed, this field contains a frame number.
0	[31:22], [7:4]	r	Reserved Read as 0; should be written with 0.

Device IN Endpoint Common Interrupt Mask Register (DIEPMSK)

This register works with each of the Device IN Endpoint Interrupt (DIEPINTx) registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the DIEPINTx register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

- Mask interrupt: 0_B
- Unmask interrupt: 1_B

DIEPMSK

Device IN Endpoint Common Interrupt Mask Register

(810_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16										
0																									
r																									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
0		NAK Msk		0		BNAI nIntr Msk		Txfi oUn drnM sk		0		INEP Nak EffM sk		0		INTk nTX FEm pMs k		Time OUT Msk		AHB ErrM sk		EPDi sbld Msk		Xfer Com pIMs k	
r		rw		r		rw		rw		r		rw		r		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
XferCompIMsk	0	rw	Transfer Completed Interrupt Mask
EPDisbldMsk	1	rw	Endpoint Disabled Interrupt Mask
AHBErrMsk	2	rw	AHB Error Mask
TimeOUTMsk	3	rw	Timeout Condition Mask (Non-isochronous endpoints)

Universal Serial Bus (USB)

Field	Bits	Type	Description
INTknTXFEmpMsk	4	rw	IN Token Received When TxFIFO Empty Mask
INEPNakEffMsk	6	rw	IN Endpoint NAK Effective Mask
TxfifoUndrnMsk	8	rw	Fifo Underrun Mask
BNAIntrMsk	9	rw	BNA Interrupt Mask
NAKMsk	13	rw	NAK interrupt Mask
0	[31:14] , [12:10] , 7, 5	r	Reserved Read as 0; should be written with 0.

Device OUT Endpoint Common Interrupt Mask Register (DOEPMSK)

This register works with each of the Device OUT Endpoint Interrupt (DOEPINTx) registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the DOEPINTx register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

- Mask interrupt: 0_B
- Unmask interrupt: 1_B

DOEPMSK

Device OUT Endpoint Common

Interrupt Mask Register

(814_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	NYE TMs k	NAK Msk	Bble ErrM sk	0	Bna Outl ntrM sk	OutP ktErr Msk	0	Back 2Bac kSE Tup	0	OUT TknE Pdis Msk	SetU PMs k	AHB ErrM sk	EPDi sbld Msk	Xfer Com pIMs k	
r	rw	rw	rw	r	rw	rw	r	rw	r	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
XferComplMsk	0	rw	Transfer Completed Interrupt Mask
EPDisbldMsk	1	rw	Endpoint Disabled Interrupt Mask

Universal Serial Bus (USB)

Field	Bits	Type	Description
AHBErrMsk	2	rw	AHB Error
SetUPMsk	3	rw	SETUP Phase Done Mask Applies to control endpoints only.
OUTTknEPdisMsk	4	rw	OUT Token Received when Endpoint Disabled Mask Applies to control OUT endpoints only.
Back2BackSETup	6	rw	Back-to-Back SETUP Packets Received Mask Applies to control OUT endpoints only.
OutPktErrMsk	8	rw	OUT Packet Error Mask
BnaOutIntrMsk	9	rw	BNA interrupt Mask
BbleErrMsk	12	rw	Babble Interrupt Mask
NAKMsk	13	rw	NAK Interrupt Mask
NYETMsk	14	rw	NYET Interrupt Mask
0	[31:15] , [11:10] , 7, 5	r	Reserved Read as 0; should be written with 0.

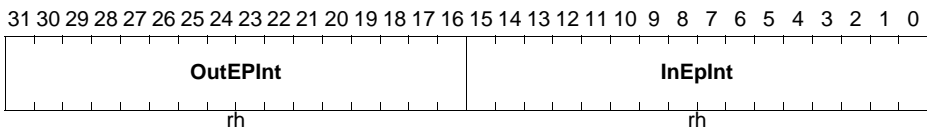
Device All Endpoints Interrupt Register (DAINT)

When a significant event occurs on an endpoint, a Device All Endpoints Interrupt register interrupts the application using the Device OUT Endpoints Interrupt bit or Device IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively). This is shown in [Figure 16-71 “Interrupt Hierarchy” on Page 16-232](#). There is one interrupt bit per endpoint, up to a maximum of 7 bits for OUT endpoints and 7 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Device Endpoint-n Interrupt register (DIEPINTx/DOEPINTx).

DAINT

Device All Endpoints Interrupt Register(818_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
InEplnt	[15:0]	rh	IN Endpoint Interrupt Bits One bit per IN Endpoint: Bit 0 for IN endpoint 0, bit 6 for endpoint 6. Bits [15:7] are not used.
OutEPInt	[31:16]	rh	OUT Endpoint Interrupt Bits One bit per OUT endpoint: Bit 16 for OUT endpoint 0, bit 22 for OUT endpoint 6. Bits [31:23] are not used.

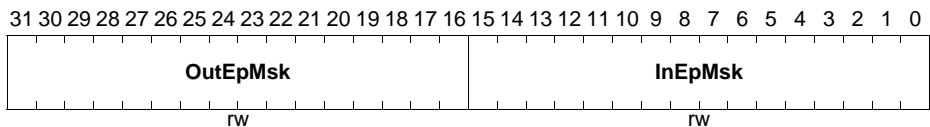
Device All Endpoints Interrupt Mask Register (DAINTMSK)

The Device Endpoint Interrupt Mask register works with the Device Endpoint Interrupt register to interrupt the application when an event occurs on a device endpoint. However, the Device All Endpoints Interrupt (DAINT) register bit corresponding to that interrupt is still set.

- Mask Interrupt: 0_B
- Unmask Interrupt: 1_B

DAINTMSK

Device All Endpoints Interrupt Mask Register(81C_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
InEpMsk	[15:0]	rw	IN EP Interrupt Mask Bits One bit per IN Endpoint: Bit 0 for IN EP 0, bit 6 for IN EP 6. Bits [15:7] are not used.
OutEpMsk	[31:16]	rw	OUT EP Interrupt Mask Bits One per OUT Endpoint: Bit 16 for OUT EP 0, bit 22 for OUT EP 6. Bits [31:23] are not used.

Device VBUS Discharge Time Register (DVBUSDIS)

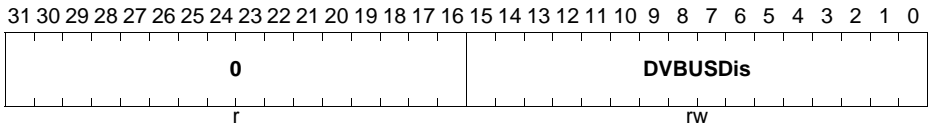
This register specifies the VBUS discharge time after VBUS pulsing during SRP.

Universal Serial Bus (USB)

DVBUSDIS

Device VBUS Discharge Time Register(828_H)

Reset Value: 0000 17D7_H



Field	Bits	Type	Description
DVBUSDis	[15:0]	rw	Device Vbus Discharge Time Specifies the Vbus discharge time after Vbus pulsing during SRP. This value equals: Vbus discharge time in PHY clocks / 1,024 The reset value is based on PHY operating at 60 MHz. Depending on the Vbus load, this value might need adjustment.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

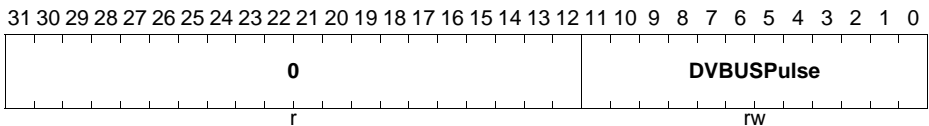
Device VBUS Pulsing Time Register (DVBUSPULSE)

This register specifies the VBUS pulsing time during SRP.

DVBUSPULSE

Device VBUS Pulsing Time Register (82C_H)

Reset Value: 0000 05B8_H



Field	Bits	Type	Description
DVBUSPulse	[11:0]	rw	Device Vbus Pulsing Time Specifies the Vbus pulsing time during SRP. This value equals: Vbus pulsing time in PHY clocks / 1,024 The reset value is based on PHY operating at 60 MHz.
0	[31:12]	r	Reserved Read as 0; should be written with 0.

Device IN Endpoint FIFO Empty Interrupt Mask Register (DIEPEMPMSK)

This register is used to control the IN endpoint FIFO empty interrupt generation (DIEPINTx.TxfEmp).

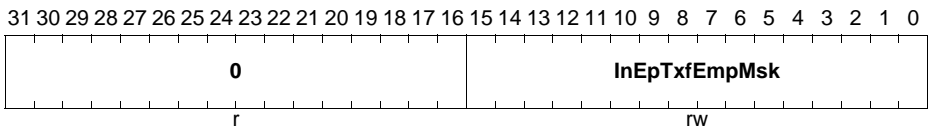
- Mask interrupt: 0_B
- Unmask interrupt: 1_B

DIEPEMPMSK

Device IN Endpoint FIFO Empty Interrupt Mask Register

(834_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
InEpTxfEmpMsk	[15:0]	rw	IN EP Tx FIFO Empty Interrupt Mask Bits These bits acts as mask bits for DIEPINTx. TxfEmp interrupt One bit per IN Endpoint: <ul style="list-style-type: none"> • Bit 0 for IN endpoint 0 • ... • Bit 6 for endpoint 6 Bits [15:7] are not used.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Device Control IN Endpoint 0 Control Register (DIEPCTL0)

This section describes the Control IN Endpoint 0 Control register. Non-zero control endpoints use registers for endpoints 1-6.

Universal Serial Bus (USB)

DIEPCTL0

Device Control IN Endpoint 0 Control Register(900_H) **Reset Value: 0000 8000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPE na	EPDi s	0	SNA K	CNA K	TxFNum			Stall	0	EPT ype	NAK Sts	0			
rwh	rwh	r	w	w	rw			rwh	r	r	rh	r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USB ActE P	0										MPS				
r	r										rw				

Field	Bits	Type	Description
MPS	[1:0]	rw	Maximum Packet Size Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. 00 _B 64 bytes 01 _B 32 bytes 10 _B 16 bytes 11 _B 8 bytes
USBActE P	15	r	USB Active Endpoint This bit is always set to 1, indicating that control endpoint 0 is always active in all configurations and interfaces.
NAKSts	17	rh	NAK Status Indicates the following: 0 _B The core is transmitting non-NAK handshakes based on the FIFO status 1 _B The core is transmitting NAK handshakes on this endpoint. When this bit is set, either by the application or core, the core stops transmitting data, even if there is data available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.
EPT ype	[19:18]	r	Endpoint Type Hardcoded to 00 _B for control.

Universal Serial Bus (USB)

Field	Bits	Type	Description
Stall	21	rwh	STALL Handshake The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. This bit is set only by software and cleared only by hardware.
TxFNum	[25:22]	rw	TxFIFO Number <ul style="list-style-type: none"> This value is set to the FIFO number that is assigned to IN Endpoint 0.
CNAK	26	w	Clear NAK A write to this bit clears the NAK bit for the endpoint.
SNAK	27	w	Set NAK A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for an endpoint after a SETUP packet is received on that endpoint.
EPDis	30	rwh	Endpoint Disable The application sets this bit to stop transmitting data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled Interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. This bit is set only by software and cleared only by hardware.

Universal Serial Bus (USB)

Field	Bits	Type	Description
EPEna	31	rwh	Endpoint Enable <ul style="list-style-type: none"> When Scatter/Gather DMA mode is enabled, for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. When Scatter/Gather DMA mode is disabled—such as in buffer-pointer based DMA mode—this bit indicates that data is ready to be transmitted on the endpoint. The core clears this bit before setting the following interrupts on this endpoint: <ul style="list-style-type: none"> Endpoint Disabled Transfer Completed This bit is set only by software and cleared only by hardware.
0	[29:28], 20, 16, [14:2]	r	Reserved Read as 0; should be written with 0.

Device Control OUT Endpoint 0 Control Register (DOEPTL0)

This section describes the Control OUT Endpoint 0 Control register. Non-zero control endpoints use registers for endpoints 1-6.

DOEPTL0

Device Control OUT Endpoint 0 Control Register(B00_H) Reset Value: 0000 8000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPE na	EPDi s	0	SNA K	CNA K			0			Stall	Snp	EPT ype	NAK Sts	0	
rwh	r	r	w	w			r			rwh	rw	r	rh	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USB ActE P							0								MPS
r							r								r

Universal Serial Bus (USB)

Field	Bits	Type	Description
MPS	[1:0]	r	<p>Maximum Packet Size</p> <p>The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN Endpoint 0.</p> <p>00_B 64 bytes 01_B 32 bytes 10_B 16 bytes 11_B 8 bytes</p>
USBActEP	15	r	<p>USB Active Endpoint</p> <p>This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.</p>
NAKSts	17	rh	<p>NAK Status</p> <p>Indicates the following:</p> <p>0_B The core is transmitting non-NAK handshakes based on the FIFO status. 1_B The core is transmitting NAK handshakes on this endpoint.</p> <p>When either the application or the core sets this bit, the core stops receiving data, even if there is space in the RxFIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>
EPTYPE	[19:18]	r	<p>Endpoint Type</p> <p>Hardcoded to 00 for control.</p>
SnP	20	rw	<p>Snoop Mode</p> <p>This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>
Stall	21	rwh	<p>STALL Handshake</p> <p>The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p> <p>This bit is set only by software and cleared only by hardware.</p>
CNAK	26	w	<p>Clear NAK</p> <p>A write to this bit clears the NAK bit for the endpoint.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
SNAK	27	w	<p>Set NAK</p> <p>A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set bit on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.</p>
EPDis	30	r	<p>Endpoint Disable</p> <p>The application cannot disable control OUT endpoint 0.</p>
EPEna	31	rwh	<p>Endpoint Enable</p> <p>When Scatter/Gather DMA mode is enabled, for OUT endpoints this bit indicates that the descriptor structure and data buffer to receive data is setup.</p> <ul style="list-style-type: none"> When Scatter/Gather DMA mode is disabled—(such as for buffer-pointer based DMA mode)—this bit indicates that the application has allocated the memory to start receiving data from the USB. <p>The core clears this bit before setting any of the following interrupts on this endpoint:</p> <ul style="list-style-type: none"> SETUP Phase Done Endpoint Disabled Transfer Completed <p><i>Note: In DMA mode, this bit must be set for the core to transfer SETUP data packets into memory.</i></p> <p>This bit is set only by software and cleared only by hardware.</p>
0	[29:28] , [25:22] , 16, [14:2]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Device Endpoint-n Control Register (DIEPCTLx/DOEPCTLx)

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Note: The fields of the DIEPCTLx/DOEPCTLx register change, depending on interrupt/bulk or isochronous/control endpoint.

Universal Serial Bus (USB)

DIEPCTLx (x=1-6)

Device Endpoint-x Control Register [INTBULK]

(900_H + x*20_H)

Reset Value: 0000 0000_H

DOEPCTLx (x=1-6)

Device Endpoint-x Control Register [INTBULK]

(B00_H + x*20_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPE na	EPDi s	SetD 1PID	SetD 0PID	SNA K	CNA K	TxFNum			Stall	Snp	EPT ype	NAK Sts	DPID		
rwh	rwh	w	w	w	w	rw			rw	rw	rw	rh	rh		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USB ActE P	0				MPS										
rwh	r				rw										

Field	Bits	Type	Description
MPS	[10:0]	rw	Maximum Packet Size Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.
USBActEP	15	rwh	USB Active Endpoint Applies to IN and OUT endpoints. Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. This bit is set only by software and can be cleared by hardware or a software write of 0 to the bit.

Universal Serial Bus (USB)

Field	Bits	Type	Description
DPID	16	rh	<p>Endpoint Data PID</p> <p>Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <p>0_B DATA0 1_B DATA1</p> <p>This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p>
NAKSts	17	rh	<p>NAK Status</p> <p>Applies to IN and OUT endpoints. Indicates the following:</p> <p>0_B The core is transmitting non-NAK handshakes based on the FIFO status. 1_B The core is transmitting NAK handshakes on this endpoint.</p> <p>When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.</p> <p>Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
EPTYPE	[19:18]	rw	<p>Endpoint Type Applies to IN and OUT endpoints. This is the transfer type supported by this logical endpoint.</p> <p>00_B Control 01_B Isochronous 10_B Bulk 11_B Interrupt</p>
Snp	20	rw	<p>Snoop Mode Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>
Stall	21	rw	<p>STALL Handshake Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.</p>
TxFNum	[25:22]	rw	<p>TxFIFO Number These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.</p>
CNAK	26	w	<p>Clear NAK Applies to IN and OUT endpoints. A write to this bit clears the NAK bit for the endpoint.</p>
SNAK	27	w	<p>Set NAK Applies to IN and OUT endpoints. A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
SetD0PID	28	w	<p>Set DATA0 PID</p> <p>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p>
SetD1PID	29	w	<p>29 Set DATA1 PID</p> <p>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode.</p>
EPDis	30	rwh	<p>Endpoint Disable</p> <p>Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. This bit is set only by software and cleared only by hardware.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
EPEna	31	rwh	<p>Endpoint Enable</p> <p>Applies to IN and OUT endpoints.</p> <ul style="list-style-type: none"> • When Scatter/Gather DMA mode is enabled, • For IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. • For OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. • When Scatter/Gather DMA mode is enabled—such as for buffer-pointer based DMA mode: <ul style="list-style-type: none"> – For IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. – For OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. – The core clears this bit before setting any of the following interrupts on this endpoint: <ul style="list-style-type: none"> • SETUP Phase Done • Endpoint Disabled • Transfer Completed <p><i>Note: For control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory.</i></p> <p>This bit is set only by software and cleared only by hardware.</p>
0	[14:11]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Universal Serial Bus (USB)

DIEPCTLx (x=1-6)

Device Endpoint-x Control Register [ISOCONT]

(900_H + x*20_H)

Reset Value: 0000 0000_H

DOEPCTLx (x=1-6)

Device Endpoint-x Control Register [ISOCONT]

(B00_H + x*20_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPE na	EPDi s	SetO ddFr	SetE venFr	SNA K	CNA K	TxFNum			Stall	Snp	EPTType		NAK Sts	EO_ FrNu m	
rwh	rwh	w	w	w	w	rw			rwh	rw	rw		rh	rh	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USB ActE P	0			MPS											
rwh	r			rw											

Field	Bits	Type	Description
MPS	[10:0]	rw	Maximum Packet Size Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.
USBActEP	15	rwh	USB Active Endpoint Applies to IN and OUT endpoints. Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. This bit is set only by software and can be cleared by hardware or a software write of 0 to the bit.

Universal Serial Bus (USB)

Field	Bits	Type	Description
EO_FrNum	16	rh	<p>Even/Odd Frame</p> <p>Applies to isochronous IN and OUT endpoints only. In non-Scatter/Gather DMA mode, the bit Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <p>0_B Even frame 1_B Odd rame</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.</p>
NAKSts	17	rh	<p>NAK Status</p> <p>Applies to IN and OUT endpoints. Indicates the following:</p> <p>0_B The core is transmitting non-NAK handshakes based on the FIFO status. 1_B The core is transmitting NAK handshakes on this endpoint.</p> <p>When either the application or the core sets this bit: The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.</p> <p>For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO.</p> <p>For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO.</p> <p>Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
EPTYPE	[19:18]	rw	<p>Endpoint Type Applies to IN and OUT endpoints. This is the transfer type supported by this logical endpoint.</p> <p>00_B Control 01_B Isochronous 10_B Bulk 11_B Interrupt</p>
Snp	20	rw	<p>Snoop Mode Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.</p>
Stall	21	rwh	<p>STALL Handshake Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. This bit is set only by software and cleared only by hardware.</p>
TxFNum	[25:22]	rw	<p>TxFIFO Number These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.</p>
CNAK	26	w	<p>Clear NAK Applies to IN and OUT endpoints. A write to this bit clears the NAK bit for the endpoint.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
SNAK	27	w	<p>Set NAK</p> <p>Applies to IN and OUT endpoints. A write to this bit sets the NAK bit for the endpoint.</p> <p>Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.</p>
SetEvenFr	28	w	<p>In non-Scatter/Gather DMA mode: Set Even frame</p> <p>Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd frame (EO_FrNum) field to even frame.</p> <p>When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.</p>
SetOddFr	29	w	<p>Set Odd frame</p> <p>Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd frame (EO_FrNum) field to odd frame.</p> <p>This field is not applicable for Scatter/Gather DMA mode.</p>
EPDis	30	rwh	<p>Endpoint Disable</p> <p>Applies to IN and OUT endpoints.</p> <p>The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint.</p> <p>This bit is set only by software and cleared only by hardware.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
EPEna	31	rwh	<p>Endpoint Enable</p> <p>Applies to IN and OUT endpoints.</p> <ul style="list-style-type: none"> • When Scatter/Gather DMA mode is enabled, • For IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. • For OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. • When Scatter/Gather DMA mode is enabled—such as for buffer-pointer based DMA mode: <ul style="list-style-type: none"> – For IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. – For OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. – The core clears this bit before setting any of the following interrupts on this endpoint: <ul style="list-style-type: none"> • SETUP Phase Done • Endpoint Disabled • Transfer Completed <p><i>Note: For control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory.</i></p> <p>This bit is set only by software and cleared only by hardware.</p>
0	[14:11]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Device Endpoint-n Interrupt Register (DIEPINTx/DOEPINTx)

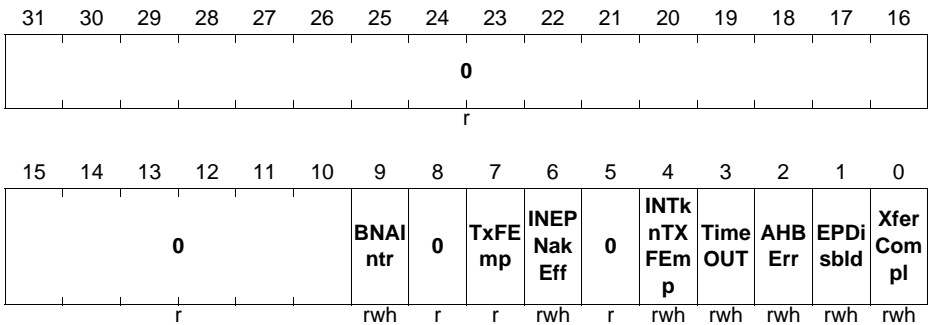
This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in [Figure 16-71 “Interrupt Hierarchy” on Page 16-232](#). The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Note: In the DIEPINTx/DOEPINTx registers, status bits with access type ‘rwh’ are set by hardware. To clear these bits, the application must write 1 into these bits.

Universal Serial Bus (USB)

DIEPINT_x (x=0-6)

Device Endpoint-x Interrupt Register (908_H + x*20_H) **Reset Value: 0000 0080_H**



Field	Bits	Type	Description
XferCompl	0	rwh	<p>Transfer Completed Interrupt Applies to IN and OUT endpoints.</p> <ul style="list-style-type: none"> • When Scatter/Gather DMA mode is enabled • For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. • For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is set. • When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.
EPDisblid	1	rwh	<p>Endpoint Disabled Interrupt Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.</p>
AHBErr	2	rwh	<p>AHB Error Applies to IN and OUT endpoints. This is generated only in DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.</p>

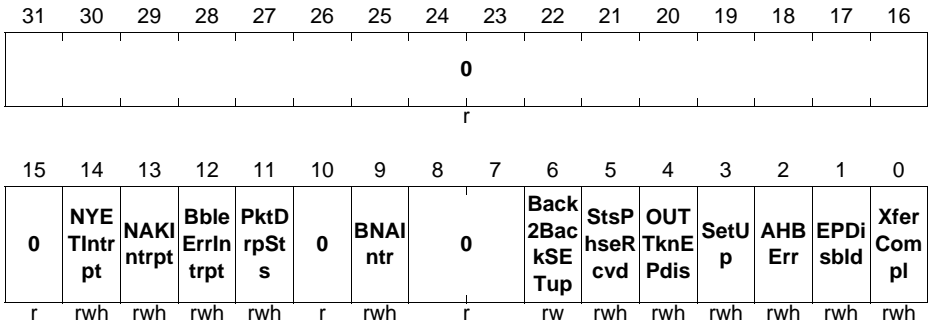
Universal Serial Bus (USB)

Field	Bits	Type	Description
TimeOUT	3	rwh	<p>Timeout Condition</p> <ul style="list-style-type: none"> Applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is not asserted. <p>Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.</p>
INTknTXF Emp	4	rwh	<p>IN Token Received When Tx FIFO is Empty</p> <p>Indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.</p>
INEPNakeff	6	rwh	<p>IN Endpoint NAK Effective</p> <p>Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLx.CNAK. This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. This bit is applicable only when the endpoint is enabled.</p>
TxFEmp	7	r	<p>Transmit FIFO Empty</p> <p>This bit is valid only for IN Endpoints. This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the Tx FIFO Empty Level bit in the Core AHB Configuration register (GAHB_CFG.NPTxFEmpLvl).</p>
BNAIntr	9	rwh	<p>BNA (Buffer Not Available) Interrupt</p> <p>The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done. This bit is valid only when Scatter/Gather DMA mode is enabled.</p>
0	[31:10], 8, 5	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Universal Serial Bus (USB)

DOEPINTx (x=0-6)

Device Endpoint-x Interrupt Register (B08_H + x*20_H) **Reset Value: 0000 0080_H**



Field	Bits	Type	Description
XferCompl	0	rwh	<p>Transfer Completed Interrupt Applies to IN and OUT endpoints.</p> <ul style="list-style-type: none"> When Scatter/Gather DMA mode is enabled For IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. For OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.
EPDisblid	1	rwh	<p>Endpoint Disabled Interrupt Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.</p>
AHBErr	2	rwh	<p>AHB Error Applies to IN and OUT endpoints. This is generated only in DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
SetUp	3	rwh	<p>SETUP Phase Done</p> <p>Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.</p>
OUTTknE Pdis	4	rwh	<p>OUT Token Received When Endpoint Disabled</p> <p>Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.</p>
StsPhseR cvd	5	rwh	<p>Status Phase Received For Control Write</p> <p>This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in case of Scatter Gather DMA mode.</p>
Back2Back kSETup	6	rw	<p>Back-to-Back SETUP Packets Received</p> <p>Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. For information about handling this interrupt, see “Handling More Than Three Back-to-Back SETUP Packets” on Page 16-97.</p>
BNAIntr	9	rwh	<p>BNA (Buffer Not Available) Interrupt</p> <p>The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done. This bit is valid only when Scatter/Gather DMA mode is enabled.</p>

Universal Serial Bus (USB)

Field	Bits	Type	Description
PktDrpSts	11	rwh	Packet Dropped Status This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. This bit is valid in non Scatter/Gather DMA mode when periodic transfer interrupt feature is selected.
BbleErrInt rpt	12	rwh	BbleErr (Babble Error) interrupt The core generates this interrupt when babble is received for the endpoint.
NAKIntrpt	13	rwh	NAK interrupt The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the TXFIFO.
NYETIntrpt	14	rwh	NYET interrupt The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.
0	[31:15], 10, [8:7]	r	Reserved Read as 0; should be written with 0.

Device Endpoint 0 Transfer Size Register (DIEPTSIZ0/DOEPTSIZ0)

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using Endpoint Enable bit of the Device Control Endpoint 0 Control registers (DIEPCTL0.EPEna/DOEPCCTL0.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Non-zero endpoints use the registers for endpoints 1-6.

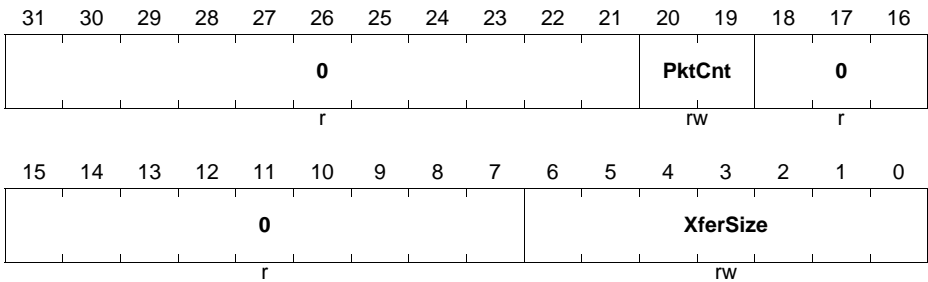
When Scatter/Gather DMA mode is enabled, this register must not be programmed by the application. If the application reads this register when Scatter/Gather DMA mode is enabled, the core returns all zeros.

Universal Serial Bus (USB)

DIEPTSIZE0

Device IN Endpoint 0 Transfer Size Register(910_H)

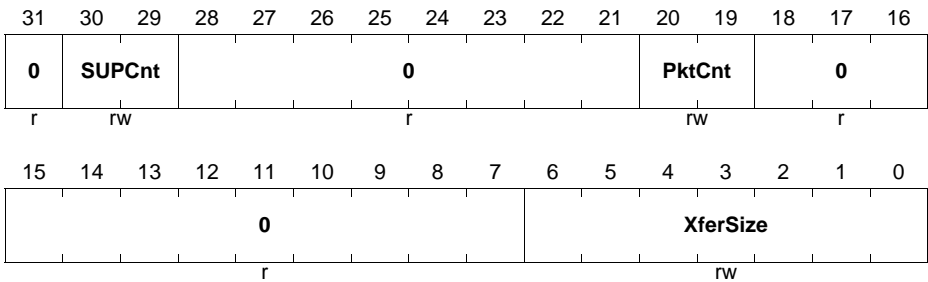
Reset Value: 0000 0000_H



Field	Bits	Type	Description
XferSize	[6:0]	rw	<p>Transfer Size</p> <p>Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.</p> <p>The core decrements this field every time a packet from the external memory is written to the TxFIFO.</p>
PktCnt	[20:19]	rw	<p>Packet Count</p> <p>Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.</p>
0	[31:21] , [18:7]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

DOEPTSIZE

Device OUT Endpoint 0 Transfer Size Register(B10_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
XferSize	[6:0]	rw	Transfer Size Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.
PktCnt	[20:19]	rw	Packet Count This field is decremented to zero after a packet is written into the RxFIFO.
SUPCnt	[30:29]	rw	SETUP Packet Count This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 01 _B 1 packet 10 _B 2 packets 11 _B 3 packets
0	31, [28:21], [18:7]	r	Reserved Read as 0; should be written with 0.

Device Endpoint-n Transfer Size Register (DIEPTSIZx/DOEPTSIZx)

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLx.EPEna/DOEPCTLx.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

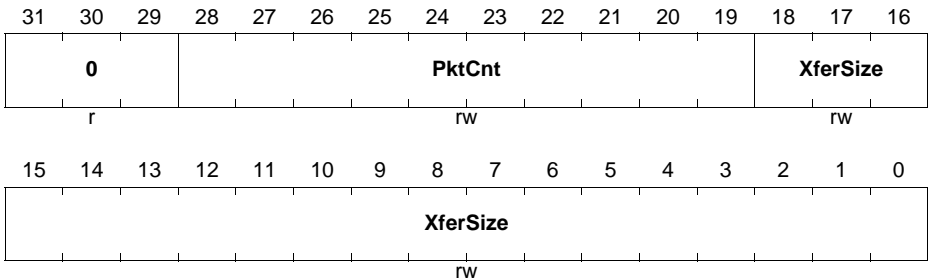
Universal Serial Bus (USB)

This register is used only for endpoints other than Endpoint 0.

Note: When Scatter/Gather DMA mode is enabled, this register must not be programmed by the application. If the application reads this register when Scatter/Gather DMA mode is enabled, the core returns all zeros

DIEPTSIZx (x=1-6)

Device Endpoint-x Transfer Size Register(910_H + x*20_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
XferSize	[18:0]	rw	<p>Transfer Size</p> <p>This field contains the transfer size in bytes for the current endpoint.</p> <p>The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.</p> <ul style="list-style-type: none"> IN Endpoints: The core decrements this field every time a packet from the external memory is written to the TxFIFO.
PktCnt	[28:19]	rw	<p>Packet Count</p> <p>Indicates the total number of USB packets that constitute the Transfer Size amount of data for this endpoint.</p> <ul style="list-style-type: none"> IN Endpoints: This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO..
0	[31:29]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

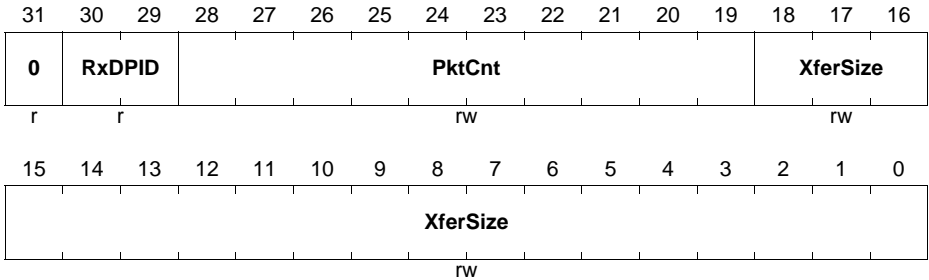
Note: The fields of the DOEPTSIZx register change, depending on isochronous or control OUT endpoint.

DOEPTSIZx (x=1-6)

Device Endpoint-x Transfer Size Register [ISO]

(B10_H + x*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
XferSize	[18:0]	rw	<p>Transfer Size</p> <p>This field contains the transfer size in bytes for the current endpoint.</p> <p>The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.</p> <ul style="list-style-type: none"> OUT Endpoints: The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.
PktCnt	[28:19]	rw	<p>Packet Count</p> <p>Indicates the total number of USB packets that constitute the Transfer Size amount of data for this endpoint.</p> <ul style="list-style-type: none"> OUT Endpoints: This field is decremented every time a packet (maximum size or short packet) is written to the Rx FIFO.

Universal Serial Bus (USB)

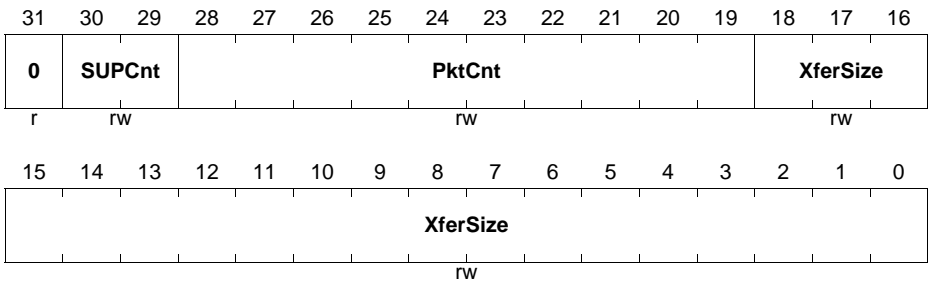
Field	Bits	Type	Description
RxDPID	[30:29]	r	Received Data PID Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint. 00 _B DATA0 01 _B DATA2 10 _B DATA1 11 _B MDATA
0	31	r	Reserved Read as 0; should be written with 0.

DOEPTSIZE_x (x=1-6)

Device Endpoint-x Transfer Size Register [CONT]

(B10_H + x*20_H)

Reset Value: 0000 0000_H



Universal Serial Bus (USB)

Field	Bits	Type	Description
XferSize	[18:0]	rw	<p>Transfer Size</p> <p>This field contains the transfer size in bytes for the current endpoint.</p> <p>The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.</p> <ul style="list-style-type: none"> OUT Endpoints: The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.
PktCnt	[28:19]	rw	<p>Packet Count</p> <p>Indicates the total number of USB packets that constitute the Transfer Size amount of data for this endpoint.</p> <ul style="list-style-type: none"> OUT Endpoints: This field is decremented every time a packet (maximum size or short packet) is written to the RxFIFO.
SUPCnt	[30:29]	rw	<p>SETUP Packet Count</p> <p>Applies to control OUT Endpoints only.</p> <p>This field specifies the number of back-to-back SETUP data packets the endpoint can receive.</p> <p>01_B 1 packet 10_B 2 packets 11_B 3 packets</p>
0	31	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Device Endpoint-n DMA Address Register (DIEPDMAx/DOEPDMAx)

These registers are implemented in RAM.

DIEPDMA_x (x=0-6)

Device Endpoint-x DMA Address Register(914_H + x*20_H)

Reset Value:

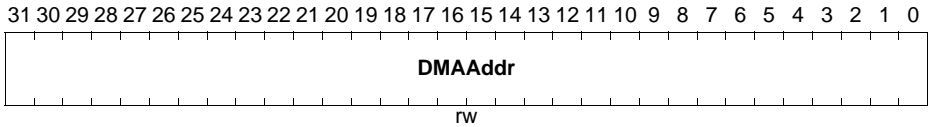
XXXX XXXX_H

DOEPDMA_x (x=0-6)

Device Endpoint-x DMA Address Register(B14_H + x*20_H)

Reset Value:

XXXX XXXX_H



Field	Bits	Type	Description
DMAAddr	[31:0]	rw	<p>DMA Address</p> <p>Holds the start address of the external memory for storing or fetching endpoint data.</p> <p><i>Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten.</i></p> <p>This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.</p> <ul style="list-style-type: none"> When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.

Device Endpoint-n DMA Buffer Address Register (DIEPDMAB_x/DOEPDMAB_x)

These fields are present only in case of Scatter/Gather DMA. These registers are implemented in RAM.

DIEPDMAB_x (x=0-6)

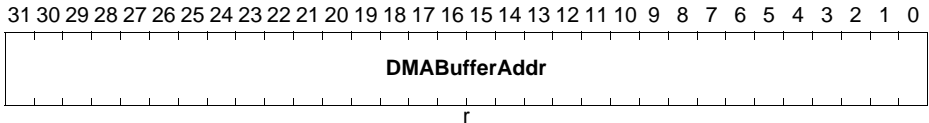
Device Endpoint-x DMA Buffer Address Register(91C_H + x*20_H) **Reset Value:**

XXXX XXXX_H

DOEPDMAB_x (x=0-6)

Device Endpoint-x DMA Buffer Address Register(B1C_H + x*20_H) **Reset Value:**

XXXX XXXX_H



Field	Bits	Type	Description
DMABufferAddr	[31:0]	r	DMA Buffer Address Holds the current buffer address. This register is updated as and when the data transfer for the corresponding endpoint is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

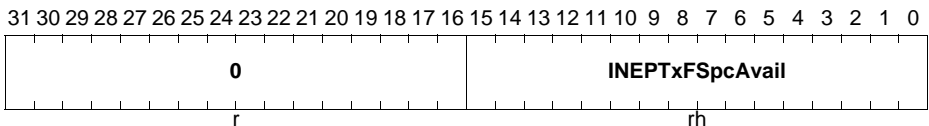
Device IN Endpoint Transmit FIFO Status Register (DTXFST_x)

This read-only register contains the free space information for the Device IN endpoint Tx FIFO.

DTXFST_x (x=0-6)

Device IN Endpoint Transmit FIFO Status Register(918_H + x*20_H) **Reset Value:**

0000 0000_H



Field	Bits	Type	Description
INEPTxFSpcAvail	[15:0]	rh	IN Endpoint TxFIFO Space Avail Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 0 _H Endpoint TxFIFO is full 1 _H 1 word available 2 _H 2 words available Others: Up to n words can be selected (0 < n < 256); selections greater than n are reserved
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Power and Clock Gating Registers

There is a single register for power and clock gating. It is available in both Host and Device modes.

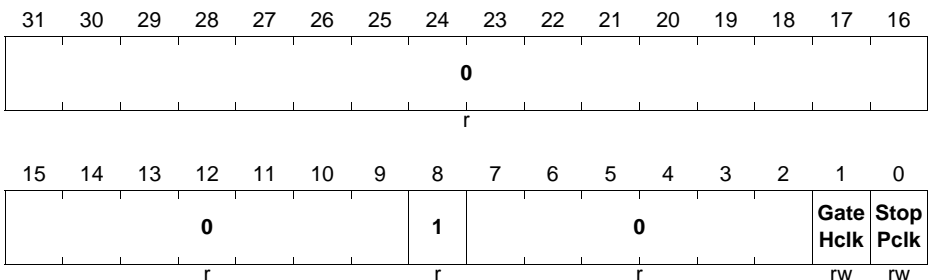
Power and Clock Gating Control Register (PCGCCTL)

This register is available in Host and Device modes. The application can use this register to control the core's clock gating features.

PCGCCTL

Power and Clock Gating Control Register(E00_H)

Reset Value: 0000 0100_H



Universal Serial Bus (USB)

Field	Bits	Type	Description
StopPclk	0	rw	Stop Pclk The application sets this bit to stop the PHY clock (phy_clk) when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.
GateHclk	1	rw	Gate Hclk The application sets this bit to gate hclk to modules other than the AHB Slave and Master and wakeup logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.
1	8	r	Reserved Read as 1; should be written with 1.
0	[31:9], [7:2]	r	Reserved Read as 0; should be written with 0.

16.20 Interconnects

The interconnects section describes the connectivity of the module.

Table 16-24 Pin Connections

Input/Output	I/O	Connected To	Description
USB0.ID	I	P0.9	ID pad signal
USB0.D+	I/O	USB_DP	Data + signal
USB0.D-	I/O	USB_DM	Data - signal
USB0.VBUS	I/O	VBUS	VBUS signal
USB0.DRIVEVBUS	O	P0.1 P3.2	Drive VBUS signal



17 Universal Serial Interface Channel (USIC)

The **Universal Serial Interface Channel** module (USIC) is a flexible interface module covering several serial communication protocols. A USIC module contains two independent communication channels named USIC_x_CH0 and USIC_x_CH1, with *x* being the number of the USIC module (e.g. channel 0 of USIC module 0 is referenced as USIC0_CH0). The user can program during run-time which protocol will be handled by each communication channel and which pins are used.

References

The following documents are referenced for further information

[20] IIC Bus Specification (Philips Semiconductors v2.1)

[21] IIS Bus Specification (Philips Semiconductors June 5 1996 revision)

Table 17-1 Abbreviations

CTQ	Time Quanta Counter
DSU	Data Shift Unit
f_{PERIPH}	USIC module clock frequency
f_{PIN}	Input frequency to baud rate generator
MCLK	Master Clock
PPP	Protocol Pre-Processor
RSR	Receive Shift Register
SCLK	Shift Clock
TSR	Transmit Shift Register

17.1 Overview

This section gives an overview about the feature set of the USIC.

17.1.1 Features

Each USIC channel can be individually configured to match the application needs, e.g. the protocol can be selected or changed during run time without the need for a reset. The following protocols are supported:

- **UART** (ASC, asynchronous serial channel)
 - Module capability: receiver/transmitter with max. baud rate $f_{\text{PERIPH}} / 4$
 - Wide baud rate range down to single-digit baud rates
 - Number of data bits per data frame: 1 to 63
 - MSB or LSB first

Universal Serial Interface Channel (USIC)

- **LIN** Support by hardware (Local Interconnect Network)
 - Data transfers based on ASC protocol
 - Baud rate detection possible by built-in capture event of baud rate generator
 - Checksum generation under software control for higher flexibility
- **SSC/SPI** (synchronous serial channel with or without slave select lines)
 - Standard, Dual and Quad SPI format supported
 - Module capability: maximum baud rate $f_{\text{PERIPH}} / 2$, limited by loop delay
 - Number of data bits per data frame 1 to 63, more with explicit stop condition
 - Parity bit generation supported
 - MSB or LSB first
- **IIC** (Inter-IC Bus)
 - Application baud rate 100 kbit/s to 400 kbit/s
 - 7-bit and 10-bit addressing supported
 - Full master and slave device capability
- **IIS** (infotainment audio bus)
 - Module capability: maximum baud rate $f_{\text{PERIPH}} / 2$

Note: The real baud rates that can be achieved in a real application depend on the operating frequency of the device, timing parameters as described in the Data Sheet, signal delays on the PCB and timings of the peer device.

In addition to the flexible choice of the communication protocol, the USIC structure has been designed to reduce the system load (CPU load) allowing efficient data handling. The following aspects have been considered:

- **Data buffer capability**
The standard buffer capability includes a double word buffer for receive data and a single word buffer for transmit data. This allows longer CPU reaction times (e.g. interrupt latency).
- **Additional FIFO buffer capability**
In addition to the standard buffer capability, the received data and the data to be transmitted can be buffered in a FIFO buffer structure. The size of the receive and the transmit FIFO buffer can be programmed independently. Depending on the application needs, a total buffer capability of 64 data words can be assigned to the receive and transmit FIFO buffers of a USIC module (the two channels of the USIC module share the 64 data word buffer).
In addition to the FIFO buffer, a bypass mechanism allows the introduction of high-priority data without flushing the FIFO buffer.
- **Transmit control information**
For each data word to be transmitted, a 5-bit transmit control information has been added to automatically control some transmission parameters, such as word length, frame length, or the slave select control for the SPI protocol. The transmit control information is generated automatically by analyzing the address where the user software has written the data word to be transmitted (32 input locations = $2^5 = 5$ bit transmit control information).

Universal Serial Interface Channel (USIC)

This feature allows individual handling of each data word, e.g. the transmit control information associated to the data words stored in a transmit FIFO can automatically modify the slave select outputs to select different communication targets (slave devices) without CPU load. Alternatively, it can be used to control the frame length.

- **Flexible frame length control**

The number of bits to be transferred within a data frame is independent of the data word length and can be handled in two different ways. The first option allows automatic generation of frames up to 63 bits with a known length. The second option supports longer frames (even unlimited length) or frames with a dynamically controlled length.

- **Interrupt capability**

The events of each USIC channel can be individually routed to one of 6 service request outputs SR[5:0] available for each USIC module, depending on the application needs. Furthermore, specific start and end of frame indications are supported in addition to protocol-specific events.

- **Flexible interface routing**

Each USIC channel offers the choice between several possible input and output pins connections for the communications signals. This allows a flexible assignment of USIC signals to pins that can be changed without resetting the device.

- **Input conditioning**

Each input signal is handled by a programmable input conditioning stage with programmable filtering and synchronization capability.

- **Baud rate generation**

Each USIC channel contains its own baud rate generator. The baud rate generation can be based either on the internal module clock or on an external frequency input. This structure allows data transfers with a frequency that can not be generated internally, e.g. to synchronize several communication partners.

- **Transfer trigger capability**

In master mode, data transfers can be triggered by events generated outside the USIC module, e.g. by an input pin or a timer unit (transmit data validation). This feature allows time base related data transmission.

- **Debugger support**

The USIC offers specific addresses to read out received data without interaction with the FIFO buffer mechanism. This feature allows debugger accesses without the risk of a corrupted receive data sequence.

To reach a desired baud rate, two criteria have to be respected, the module capability and the application environment. The module capability is defined with respect to the module's input clock frequency, being the base for the module operation. Although the module's capability being much higher (depending on the module clock and the number of module clock cycles needed to represent a data bit), the reachable baud rate is generally limited by the application environment. In most cases, the application

Universal Serial Interface Channel (USIC)

environment limits the maximum reachable baud rate due to driver delays, signal propagation times, or due to EMI reasons.

Note: Depending on the selected additional functions (such as digital filters, input synchronization stages, sample point adjustment, data structure, etc.), the maximum reachable baud rate can be limited. Please also take care about additional delays, such as (internal or external) propagation delays and driver delays (e.g. for collision detection in ASC mode, for IIC, etc.).

Universal Serial Interface Channel (USIC)

A block diagram of the USIC module/channel structure is shown in **Figure 17-1**.

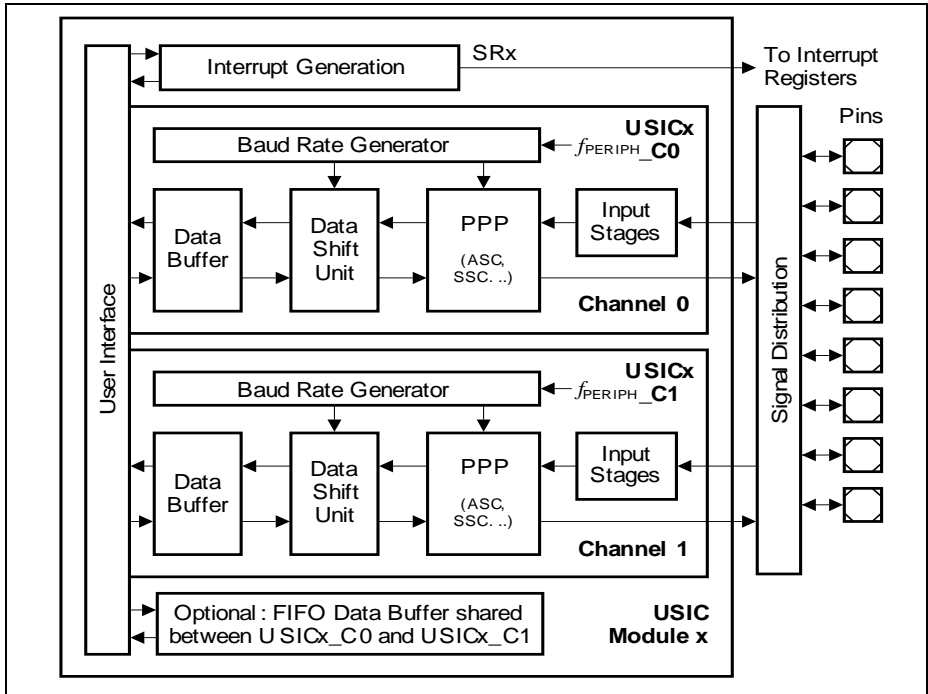


Figure 17-1 USIC Module/Channel Structure

17.2 Operating the USIC

This section describes how to operate the USIC communication channel.

17.2.1 USIC Structure Overview

This section introduces the USIC structure.

17.2.1.1 Channel Structure

The USIC module contains two independent communication channels, with a structure as shown in [Figure 17-1](#).

The data shift unit and the data buffering of each channel support full-duplex data transfers. The protocol-specific actions are handled by the protocol pre-processors (PPP). In order to simplify data handling, an additional FIFO data buffer is optionally available for each USIC module to store transmit and receive data for each channel.

Due to the independent channel control and baud rate generation, the communication protocol, baud rate and the data format can be independently programmed for each communication channel.

17.2.1.2 Input Stages

For each protocol, the number of input signals used depends on the selected protocol. Each input signal is handled by an input stage (called DX_n, where n=0-5) for signal conditioning, such as input selection, polarity control, or a digital input filter. They can be classified according to their meaning for the protocols, see [Table 17-2](#).

The inputs marked as “optional” are not needed for the standard function of a protocol and may be used for enhancements. The descriptions of protocol-specific items are given in the related protocol chapters. For the external frequency input, please refer to the baud rate generator section, and for the transmit data validation, to the data handling section.

Universal Serial Interface Channel (USIC)
Table 17-2 Input Signals for Different Protocols

Selected Protocol	Shift Data Input(s) (handled by DX0, DX3, DX4 and DX5)¹⁾	Shift Clock Input (handled by DX1)	Shift Control Input (handled by DX2)
ASC, LIN	RXD	optional: external frequency input or TXD collision detection	optional: transmit data validation
Standard SSC, SPI (Master)	DIN0 (MRST, MISO)	optional: external frequency input or delay compensation	optional: transmit data validation or delay compensation
Standard SSC, SPI (Slave)	DIN0 (MTRSR, MOSI)	SCLKIN	SELIN
Dual- SSC, SPI (Master)	DIN[1:0] (MRST[1:0], MISO[1:0])	optional: external frequency input or delay compensation	optional: transmit data validation or delay compensation
Dual- SSC, SPI (Slave)	DIN[1:0] (MTRSR[1:0], MOSI[1:0])	SCLKIN	SELIN
Quad- SSC, SPI (Master)	DIN[3:0] (MRST[3:0], MISO[3:0])	optional: external frequency input or delay compensation	optional: transmit data validation or delay compensation
Quad- SSC, SPI (Slave)	DIN[3:0] (MTRSR[3:0], MOSI[3:0])	SCLKIN	SELIN
IIC	SDA	SCL	optional: transmit data validation
IIS (Master)	DIN0	optional: external frequency input or delay compensation	optional: transmit data validation or delay compensation
IIS (Slave)	DIN0	SCLKIN	WAIN

1) ASC, IIC, IIS and standard SSC protocols use only DX0 as the shift data input.

Universal Serial Interface Channel (USIC)

Note: To allow a certain flexibility in assigning required USIC input functions to port pins of the device, each input stage can select the desired input location among several possibilities.

The available USIC signals and their port locations are listed in the interconnects section, see [Page 17-226](#).

17.2.1.3 Output Signals

For each protocol, up to 14 protocol-related output signals are available. The number of actually used outputs depends on the selected protocol. They can be classified according to their meaning for the protocols, see [Table 17-3](#).

The outputs marked as “optional” are not needed for the standard function of a protocol and may be used for enhancements. The descriptions of protocol-specific items are given in the related protocol chapters. The MCLKOUT output signal has a stable frequency relation to the shift clock output (the frequency of MCLKOUT can be higher than for SCLKOUT) for synchronization purposes of a slave device to a master device. If the baud rate generator is not needed for a specific protocol (e.g. in SSC slave mode), the SCLKOUT and MCLKOUT signals can be used as clock outputs with 50% duty cycle with a frequency that can be independent from the communication baud rate.

Table 17-3 Output Signals for Different Protocols

Selected Protocol	Shift Data Output(s) DOUT[3:0]¹⁾	Shift Clock Output SCLKOUT	Shift Control Outputs SELO[7:0]	Master Clock Output MCLKOUT
ASC, LIN	TXD	not used	not used	optional: master time base
Standard SSC, SPI (Master)	DOUT0 (MSTR, MOSI)	master shift clock	slave select, chip select	optional: master time base
Standard SSC, SPI (Slave)	DOUT0 (MRST, MISO)	optional: independent clock output	not used	optional: independent clock output
Dual-SSC, SPI (Master)	DOUT[1:0] (MSTR[1:0], MOSI[1:0])	master shift clock	slave select, chip select	optional: master time base
Dual-SSC, SPI (Slave)	DOUT[1:0] (MRST[1:0], MISO[1:0])	optional: independent clock output	not used	optional: independent clock output

Universal Serial Interface Channel (USIC)

Table 17-3 Output Signals for Different Protocols (cont'd)

Selected Protocol	Shift Data Output(s) DOUT[3:0]¹⁾	Shift Clock Output SCLKOUT	Shift Control Outputs SELO[7:0]	Master Clock Output MCLKOUT
Quad-SSC, SPI (Master)	DOUT[3:0] (MSTR[3:0], MOSI[3:0])	master shift clock	slave select, chip select	optional: master time base
Quad-SSC, SPI (Slave)	DOUT[3:0] (MRST[3:0], MISO[3:0])	optional: independent clock output	not used	optional: independent clock output
IIC	SDA	SCL	not used	optional: master time base
IIS (master)	DOUT0	master shift clock	WA	optional: master time base
IIS (slave)	DOUT0	optional: independent clock output	not used	optional: independent clock output

1) ASC, IIC, IIS and standard SSC protocols use only DOUT0 as the shift data output.

Note: To allow a certain flexibility in assigning required USIC output functions to port pins of the device, most output signals are made available on several port pins. The port control itself defines pin-by-pin which signal is used as output signal for a port pin (see port chapter). The available USIC signals and their port locations are listed in the interconnects section, see [Page 17-226](#).

17.2.1.4 Baud Rate Generator

Each USIC Channel contains a baud rate generator structured as shown in [Figure 17-2](#). It is based on coupled divider stages, providing the frequencies needed for the different protocols. It contains:

- A fractional divider to generate the input frequency $f_{PIN} = f_{FD}$ for baud rate generation based on the internal system frequency f_{PERIPH} .
- The DX1 input to generate the input frequency $f_{PIN} = f_{DX1}$ for baud rate generation based on an external signal.
- Two protocol-related counters: the divider mode counter to provide the master clock signal MCLK, the shift clock signal SCLK, and other protocol-related signals; and the capture mode timer for time interval measurement, e.g. baud rate detection.
- A time quanta counter associated to the protocol pre-processor defining protocol-specific timings, such shift control signals or bit timings, based on the input frequency f_{CTQIN} .

Universal Serial Interface Channel (USIC)

- The output signals MCLKOUT and SCLKOUT of the protocol-related divider that can be made available on pins. In order to adapt to different applications, some output characteristics of these signals can be configured. For device-specific details about availability of USIC signals on pins please refer to the interconnects section.

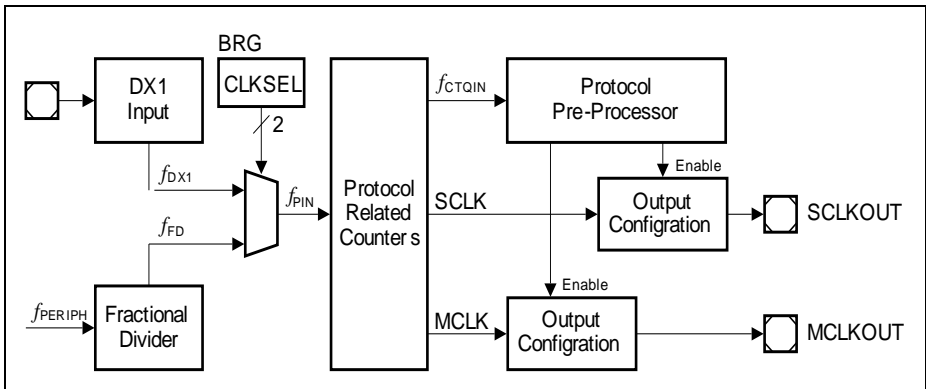


Figure 17-2 Baud Rate Generator

17.2.1.5 Channel Events and Interrupts

The notification of the user about events occurring during data traffic and data handling is based on:

- Data transfer events related to the transmission or reception of a data word, independent of the selected protocol.
- Protocol-specific events depending on the selected protocol.
- Data buffer events related to data handling by the optional FIFO data buffers.

17.2.1.6 Data Shifting and Handling

The data handling of the USIC module is based on an independent data shift unit (DSU) and a buffer structure that is similar for the supported protocols. The data shift and buffer registers are 16-bit wide (maximum data word length), but several data words can be concatenated to achieve longer data frames. The DSU inputs are the shift data (handled by input stage DX0, DX3, DX4 and DX5), the shift clock (handled by the input stage DX1), and the shift control (handled by the input stage DX2). The signal DOUT[3:0] represents the shift data outputs.

Universal Serial Interface Channel (USIC)

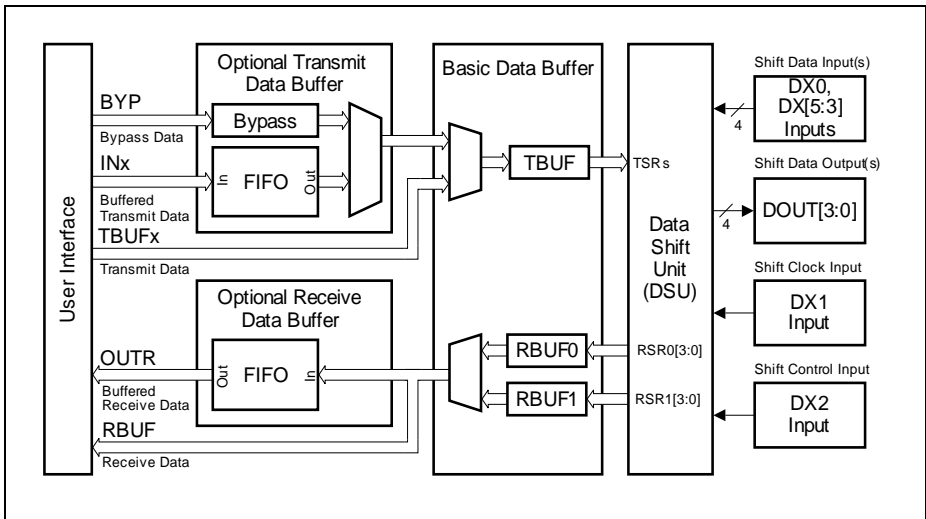


Figure 17-3 Principle of Data Buffering

The principle of data handling comprises:

- A transmitter with transmit shift registers (TSR and TSR[3:0]) in the DSU and a transmit data buffer (TBUF). A data validation scheme allows triggering and gating of data transfers by external events under certain conditions.
- A receiver with two alternating sets of receive shift registers (RSR0[3:0] and RSR1[3:0]) in the DSU and a double receive buffer structure (RBUF0, RBUF1). The alternating receive shift registers support the reception of data streams and data frames longer than one data word.
- A user interface to handle data, interrupts, and status and control information.

Basic Data Buffer Structure

The read access to received data and the write access of data to be transmitted can be handled by a basic data buffer structure.

The received data stored in the receiver buffers RBUF0/RBUF1 can be read directly from these registers. In this case, the user has to take care about the reception sequence to read these registers in the correct order. To simplify the use of the receive buffer structure, register RBUF has been introduced. A read action from this register delivers the data word received first (oldest data) to respect the reception sequence. With a read access from at least the low byte of RBUF, the data is automatically declared to be no longer new and the next received data word becomes visible in RBUF and can be read out next.

Universal Serial Interface Channel (USIC)

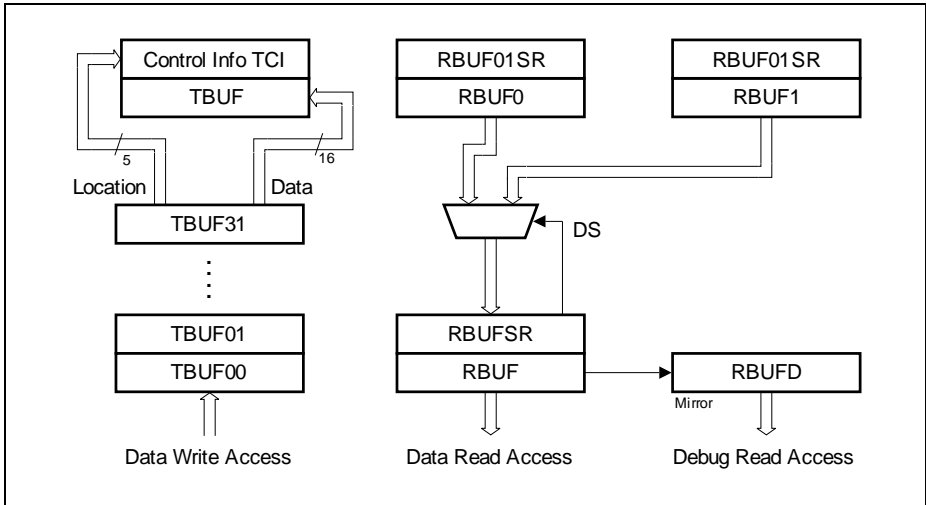


Figure 17-4 Data Access Structure without additional Data Buffer

It is recommended to read the received data words by accesses to RBUF and to avoid handling of RBUF0 and RBUF1. The USIC module also supports the use of debug accesses to receive data words. Debugger read accesses should not disturb the receive data sequence and, as a consequence, should not target RBUF. Therefore, register RBUFD has been introduced. It contains the same value as RBUF, but a read access from RBUFD does not change the status of the data (same data can be read several times). In addition to the received data, some additional status information about each received data word is available in the receiver buffer status register RBUF01SR (related to data in RBUF0 and RBUF1) and RBUFSR (related to data in RBUF).

Transmit data can be loaded to TBUF by software by writing to the transmit buffer input locations TBUF x ($x = 00-31$), consisting of 32 consecutive addresses. The data written to one of these input locations is stored in the transmit buffer TBUF. Additionally, the address of the written location is evaluated and can be used for additional control purposes. This 5-bit wide information (named **Transmit Control Information TCI**) can be used for different purposes in different protocols.

FIFO Buffer Structure

To allow easier data setup and handling, an additional data buffering mechanism can be optionally supported. The data buffer is based on the first-in-first-out principle (FIFO) that ensures that the sequence of transferred data words is respected.

If a FIFO buffer structure is used, the data handling scheme (data with associated control information) is similar to the one without FIFO. The additional FIFO buffer can be

Universal Serial Interface Channel (USIC)

independently enabled/disabled for transmission and reception (e.g. if data FIFO buffers are available for a specific USIC channel, it is possible to configure the transmit data path without and the receive data path with FIFO buffering).

The transmit FIFO buffer is addressed by using 32 consecutive address locations for INx instead of TBUFx (x=00-31) regardless of the FIFO depth. The 32 addresses are used to store the 5-bit TCI (together with the written data) associated with each FIFO entry.

The receive FIFO can be read out at two independent addresses, OUTR and OUTDR instead of RBUF and RBUFD. A read from the OUTR location triggers the next data packet to be available for the next read (general FIFO mechanism). In order to allow non-intrusive debugging (without risk of data loss), a second address location (OUTDR) has been introduced. A read at this location delivers the same value as OUTR, but without modifying the FIFO contents.

The transmit FIFO also has the capability to bypass the data stream and to load bypass data to TBUF. This can be used to generate high-priority messages or to send an emergency message if the transmit FIFO runs empty. The transmission control of the FIFO buffer can also use the transfer trigger and transfer gating scheme of the transmission logic for data validation (e.g. to trigger data transfers by events).

Note: The available size of a FIFO data buffer for a USIC channel depends on the specific device. Please refer to the implementation chapter for details about available FIFO buffer capability.

Universal Serial Interface Channel (USIC)

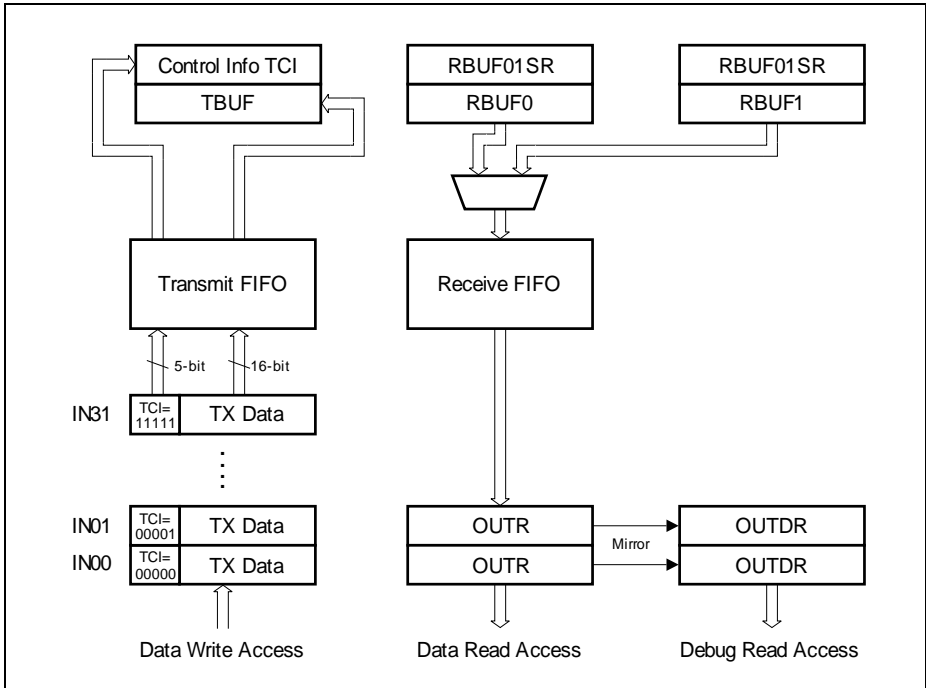


Figure 17-5 Data Access Structure with FIFO

17.2.2 Operating the USIC Communication Channel

This section describes how to operate a USIC communication channel, including protocol control and status, mode control and interrupt handling. The following aspects have to be taken into account:

- Enable the USIC module for operation and configure the behavior for the different device operation modes (see [Page 17-16](#)).
- Configure the pinning (refer to description in the corresponding protocol section).
- Configure the data structure (shift direction, word length, frame length, polarity, etc.).
- Configure the data buffer structure of the optional FIFO buffer area. A FIFO buffer can only be enabled if the related bit in register CCFG is set.
- Select a protocol by CCR.MODE. A protocol can only be selected if the related bit in register CCFG is set.

17.2.2.1 Protocol Control and Status

The protocol-related control and status information are located in the protocol control register PCR and in the protocol status register PSR. These registers are shared between the available protocols. As a consequence, the meaning of the bit positions in these registers is different within the protocols.

Use of PCR Bits

The signification of the bits in register PCR is indicated by the protocol-related alias names for the different protocols.

- PCR for the ASC protocol (see [Page 17-66](#))
- PCR for the SSC protocol (see [Page 17-98](#))
- PCR for the IIC protocol (see [Page 17-129](#))
- PCR for the IIS protocol (see [Page 17-147](#))

Use of PSR Flags

The signification of the flags in register PSR is indicated by the protocol-related alias names for the different protocols.

- PSR flags for the ASC protocol (see [Page 17-69](#))
- PSR flags for the SSC protocol (see [Page 17-102](#))
- PSR flags for the IIC protocol (see [Page 17-132](#))
- PSR flags for the IIS protocol (see [Page 17-150](#))

17.2.2.2 Mode Control

The mode control concept for system control tasks, such as suspend request for debugging, allows to program the module behavior under different device operating conditions. The behavior of a communication channel can be programmed for each of the device operating modes (normal operation, suspend mode). Therefore, each communication channel has an associated kernel state configuration register KSCFG defining its behavior in the following operating modes:

- **Normal operation:**
This operating mode is the default operating mode when no suspend request is pending. The module clock is not switched off and the USIC registers can be read or written. The channel behavior is defined by KSCFG.NOMCFG.
- **Suspend mode:**
This operating mode is requested when a suspend request is pending in the device. The module clock is not switched off and the USIC registers can be read or written. The channel behavior is defined by KSCFG.SUMCFG.

The four kernel modes defined by the register KSCFG are shown in [Table 17-4](#).

Table 17-4 USIC Communication Channel Behavior

Kernel Mode	Channel Behavior	KSCFG. NOMCFG
Run mode 0	Channel operation as specified, no impact on data transfer	00 _B
Run mode 1		01 _B
Stop mode 0	Explicit stop condition as described in the protocol chapters	10 _B
Stop mode 1		11 _B

Generally, bit field KSCFG.NOMCFG should be configured for run mode 0 as default setting for standard operation. If a communication channel should not react to a suspend request (and to continue its operation as in normal mode), bit field KSCFG.SUMCFG has to be configured with the same value as KSCFG.NOMCFG. If the communication channel should show a different behavior and stop operation when a specific stop condition is reached, the code for stop mode 0 or stop mode 1 have to be written to KSCFG.SUMCFG.

The stop conditions are defined for the selected protocol (see mode control description in the protocol section).

Note: The stop mode selection strongly depends on the application needs and it is very unlikely that different stop modes are required in parallel in the same application. As a result, only one stop mode type (either 0 or 1) should be used in the bit fields in register KSCFG. Do not mix stop mode 0 and stop mode 1 and avoid transitions

Universal Serial Interface Channel (USIC)

from stop mode 0 to stop mode 1 (or vice versa) for the same communication channel.

17.2.2.3 General Channel Events and Interrupts

The general event and interrupt structure is shown in **Figure 17-6**. If a defined condition is met, an event is detected and an event indication flag becomes automatically set. The flag stays set until it is cleared by software. If enabled, an interrupt can be generated if an event is detected. The actual status of the event indication flag has no influence on the interrupt generation. As a consequence, the event indication flag does not need to be cleared to generate further interrupts.

Additionally, the service request output SRx of the USIC channel that becomes activated in case of an event condition can be selected by an interrupt node pointer. This structure allows to assign events to interrupts, e.g. depending on the application, several events can share the same interrupt routine (several events activate the same SRx output) or can be handled individually (only one event activates one SRx output).

The SRx outputs are connected to interrupt control registers to handle the CPU reaction to the service requests. This assignment is described in the implementation section on **Page 17-153**.

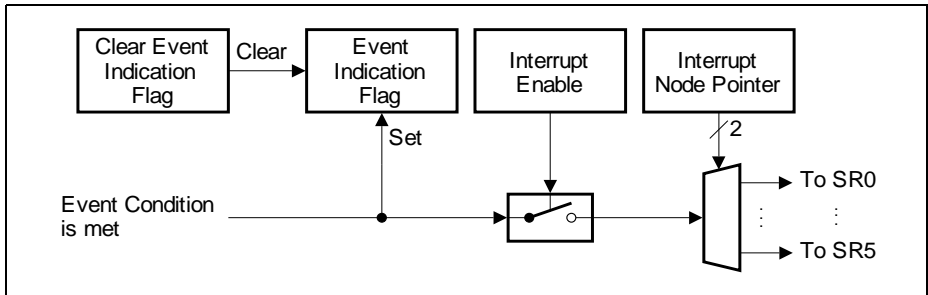


Figure 17-6 General Event and Interrupt Structure

17.2.2.4 Data Transfer Events and Interrupts

The data transfer events are based on the transmission or reception of a data word. The related indication flags are located in register PSR. All events can be individually enabled for interrupt generation.

- **Receive event to indicate that a data word has been received:**
If a new received word becomes available in the receive buffer RBUF0 or RBUF1, either a receive event or an alternative receive event occurs.
The receive event occurs if bit RBUF_{SR}.PERR = 0. It is indicated by flag PSR.RIF and, if enabled, leads to receive interrupt.
- **Receiver start event to indicate that a data word reception has started:**
When the receive clock edge that shifts in the first bit of a new data word is detected and reception is enabled, a receiver start event occurs. It is indicated by flag PSR.RSIF and, if enabled, leads to transmit buffer interrupt.
In full duplex mode, this event follows half a shift clock cycle after the transmit buffer event and indicates when the shift control settings are internally “frozen” for the current data word reception and a new setting can be programmed.
In SSC and IIS mode, the transmit data valid flag TCSR.TDV is cleared in single shot mode with the receiver start event.
- **Alternative receive event to indicate that a specific data word has been received:**
If a new received word becomes available in the receive buffer RBUF0 or RBUF1, either a receive event or an alternative receive event occurs.
The alternative receive event occurs if bit RBUF_{SR}.PERR = 1. It is indicated by flag PSR.AIF and, if enabled, leads to alternative receive interrupt.
Depending on the selected protocol, bit RBUF_{SR}.PERR is set to indicate a parity error in ASC mode, the reception of the first byte of a new frame in IIC mode, and the WA information about right/left channel in IIS mode. In SSC mode, it is used as indication if the received word is the first data word, and is set if first and reset if not.
- **Transmit shift event to indicate that a data word has been transmitted:**
A transmit shift event occurs with the last shift clock edge of a data word. It is indicated by flag PSR.TSIF and, if enabled, leads to transmit shift interrupt.
- **Transmit buffer event to indicate that a data word transmission has been started:**
When a data word from the transmit buffer TBUF has been loaded to the shift register and a new data word can be written to TBUF, a transmit buffer event occurs. This happens with the transmit clock edge that shifts out the first bit of a new data word and transmission is enabled. It is indicated by flag PSR.TBIF and, if enabled, leads to transmit buffer interrupt.
This event also indicates when the shift control settings (word length, shift direction, etc.) are internally “frozen” for the current data word transmission.
In ASC and IIC mode, the transmit data valid flag TCSR.TDV is cleared in single shot mode with the transmit buffer event.
- **Data lost event to indicate a loss of the oldest received data word:**
If the data word available in register RBUF (oldest data word from RBUF0 or RBUF1)

Universal Serial Interface Channel (USIC)

has not been read out before it becomes overwritten with new incoming data, this event occurs. It is indicated by flag PSR.DLIF and, if enabled, leads to a protocol interrupt.

Table 17-5 shows the registers, bits and bit fields indicating the data transfer events and controlling the interrupts of a USIC channel.

Table 17-5 Data Transfer Events and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Standard receive event	PSR.RIF	PSCR.CRIF	CCR.RIEN	INPR.RINP
Receive start event	PSR.RSIF	PSCR.CRSIF	CCR.RSIEN	INPR.TBINP
Alternative receive event	PSR.AIF	PSCR.CAIF	CCR.AIEN	INPR.AINP
Transmit shift event	PSR.TSIF	PSCR.CTSIF	CCR.TSIEN	INPR.TSINP
Transmit buffer event	PSR.TBIF	PSCR.CTBIF	CCR.TBIEN	INPR.TBINP
Data lost event	PSR.DLIF	PSCR.CDLIF	CCR.DLIEN	INPR.PINP

Figure 17-7 shows the two transmit events and interrupts.

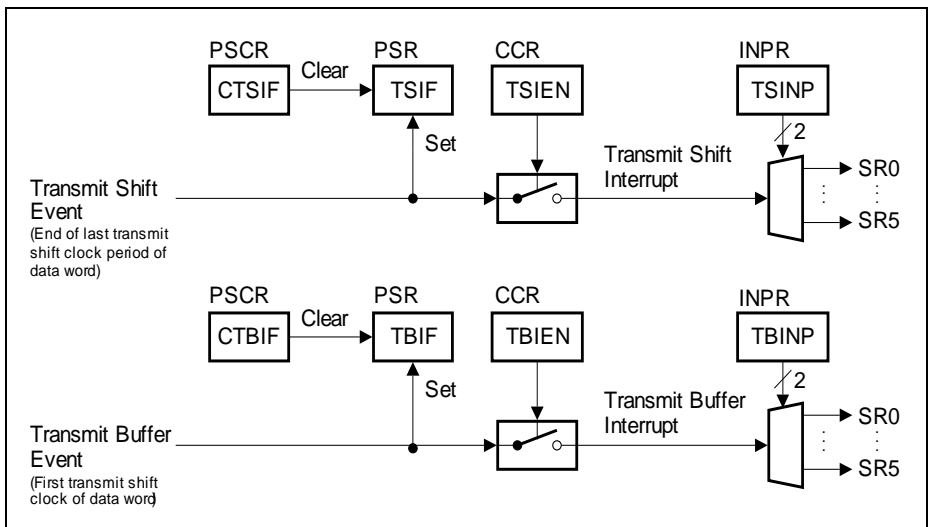


Figure 17-7 Transmit Events and Interrupts

Universal Serial Interface Channel (USIC)

Figure 17-8 shows the receive events and interrupts.

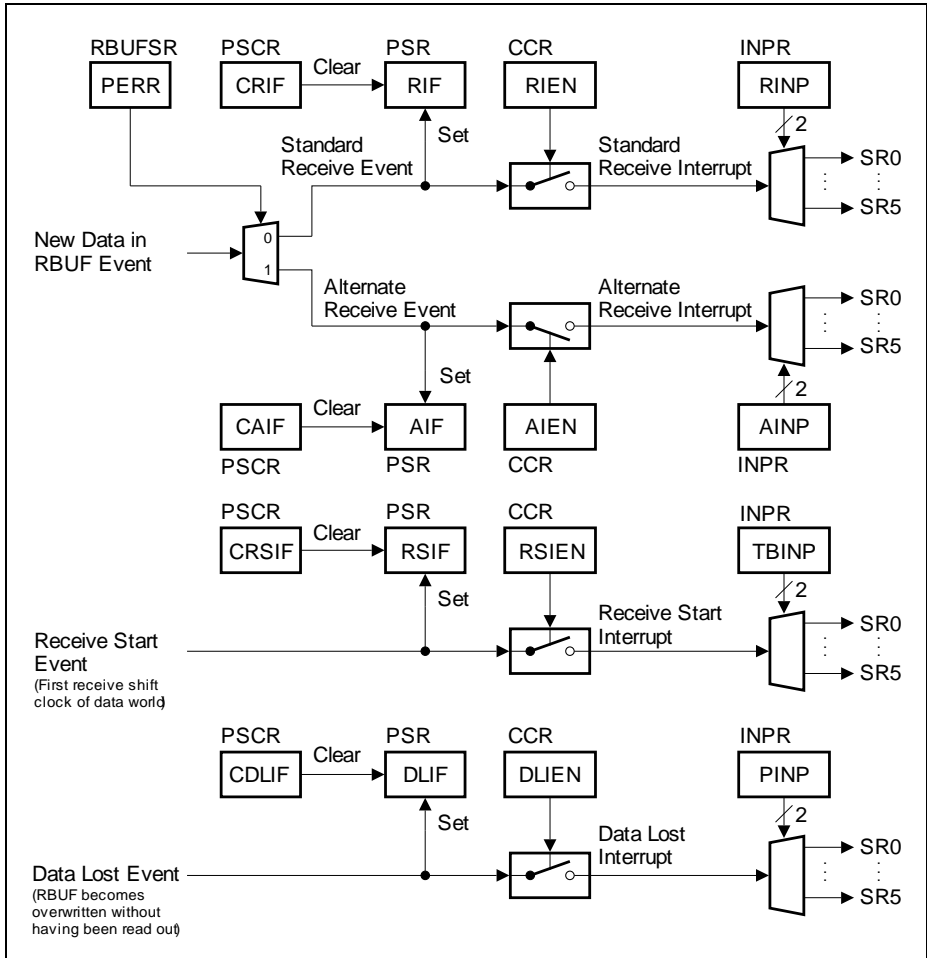


Figure 17-8 Receive Events and Interrupts

17.2.2.5 Baud Rate Generator Event and Interrupt

The baud rate generator event is based on the capture mode timer reaching its maximum value. It is indicated by flag PSR.BRGIF and, if enabled, leads to a protocol interrupt.

Universal Serial Interface Channel (USIC)

Table 17-6 shows the registers, bits and bit fields indicating the baud rate generator event and controlling the interrupt of a USIC channel.

Table 17-6 Baud Rate Generator Event and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Baud rate generator event	PSR. BRGIF	PSCR. CBRGIF	CCR. BRGIEN	INPR.PINP

Figure 17-9 shows the baud rate generator event and interrupt.

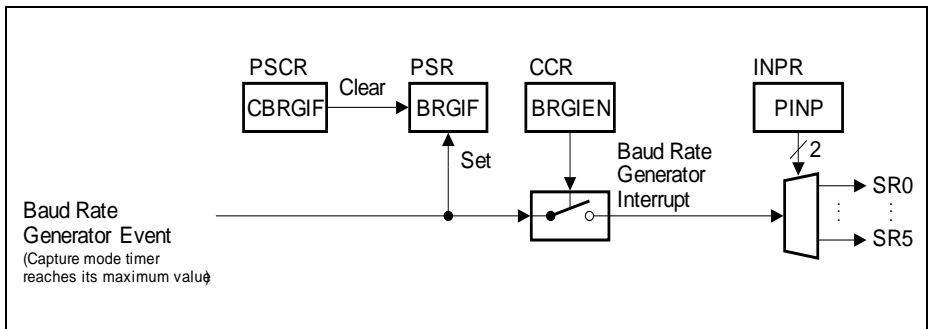


Figure 17-9 Baud Rate Generator Event and Interrupt

17.2.2.6 Protocol-specific Events and Interrupts

These events are related to protocol-specific actions that are described in the corresponding protocol chapters. The related indication flags are located in register PSR. All events can be individually enabled for the generation of the common protocol interrupt.

- Protocol-specific events in ASC mode:
Synchronization break, data collision on the transmit line, receiver noise, format error in stop bits, receiver frame finished, transmitter frame finished
- Protocol-specific events in SSC mode:
MSLS event (start-end of frame in master mode), DX2T event (start/end of frame in slave mode), both based on slave select signals, parity error
- Protocol-specific events in IIC mode:
Wrong transmit code (error in frame sequence), start condition received, repeated start condition received, stop condition received, non-acknowledge received, arbitration lost, slave read request, other general errors
- Protocol-specific events in IIS mode:
DX2T event (change on WA line), WA falling edge or rising edge detected, WA generation finished

Table 17-7 Protocol-specific Events and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Protocol-specific events in ASC mode	PSR.ST[8:2]	PSCR.CST[8:2]	PCR.CTR[7:3]	INPR.PINP
Protocol-specific events in SSC mode	PSR.ST[3:2]	PSCR.CST[3:2]	PCR.CTR[15:14]	INPR.PINP
Protocol-specific events in IIC mode	PSR.ST[8:1]	PSCR.CST[8:1]	PCR.CTR[24:18]	INPR.PINP
Protocol-specific events in IIS mode	PSR.ST[6:3]	PSCR.CST[6:3]	PCR.CTR[6:4], PCR.CTR[15]	INPR.PINP

17.2.3 Operating the Input Stages

All input stages offer the same feature set. They are used for all protocols, because the signal conditioning can be adapted in a very flexible way and the digital filters can be switched on and off separately.

17.2.3.1 General Input Structure

There are generally two types of input stages, one for the data input stages DX0, DX[5:3] and the other for non-data input stages DX[2:1], as shown in [Figure 17-10](#) and [Figure 17-11](#). The difference is that for the data input stages, the input signal can be additionally selected from the port signal HWINn if hardware port control is enabled through CCR.HPCEN bit. All other enable/disable functions and selections are controlled independently for each input stage by bits in the registers DXnCR.

The desired input signal can be selected among the input lines DXnA to DXnG and a permanent 1-level by programming bit field DSEL (for the data input stages, hardware port control must be disabled for DSEL to take effect). Please refer to the interconnects section ([Section 17.12](#)) for the device-specific input signal assignment. Bit DPOL allows a polarity inversion of the selected input signal to adapt the input signal polarity to the internal polarity of the data shift unit and the protocol state machine. For some protocols, the input signals can be directly forwarded to the data shift unit for the data transfers (DSEN = 0, INSW = 1) without any further signal conditioning. In this case, the data path does not contain any delay due to synchronization or filtering.

In the case of noise on the input signals, there is the possibility to synchronize the input signal (signal DXnS is synchronized to f_{PERIPH}) and additionally to enable a digital noise filter in the signal path. The synchronized input signal (and optionally filtered if DFEN = 1) is taken into account by DSEN = 1. Please note that the synchronization leads to a delay in the signal path of 2-3 times the period of f_{PERIPH} .

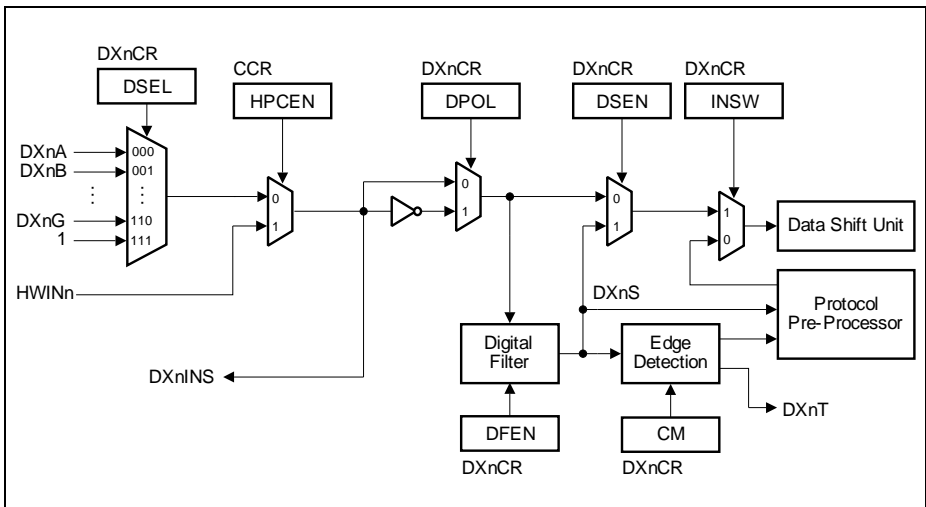


Figure 17-10 Input Conditioning for DX0 and DX[5:3]

Universal Serial Interface Channel (USIC)

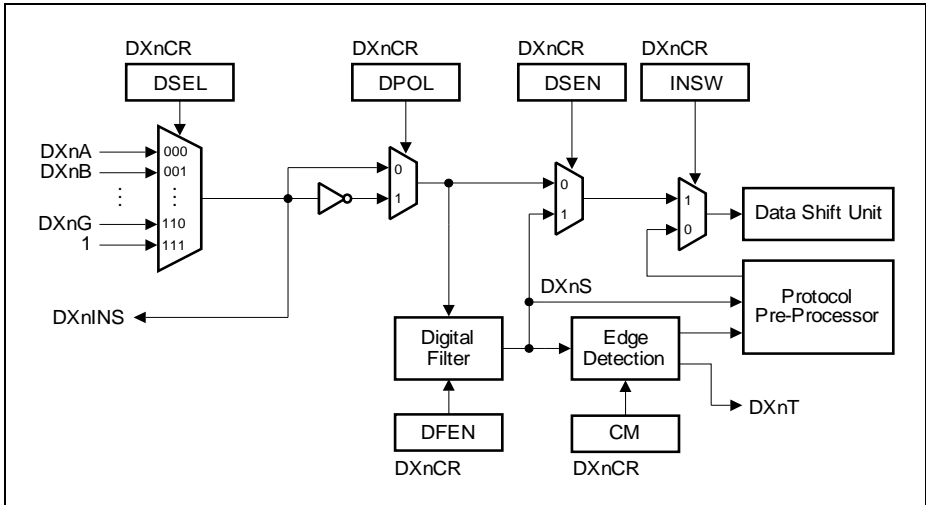


Figure 17-11 Input Conditioning for DX[2:1]

If the input signals are handled by a protocol pre-processor, the data shift unit is directly connected to the protocol pre-processor by $INSW = 0$. The protocol pre-processor is connected to the synchronized input signal $DXnS$ and, depending on the selected protocol, also evaluates the edges.

To support delay compensation in SSC and IIS protocols, the $DX1$ input stage additionally allows the receive shift clock to be controlled independently from the transmit shift clock through the bit $DCEN$. When $DCEN = 0$, the shift clock source is selected by $INSW$ and is the same for both receive and transmit. When $DCEN = 1$, the receive shift clock is derived from the selected input line as shown in [Figure 17-12](#).

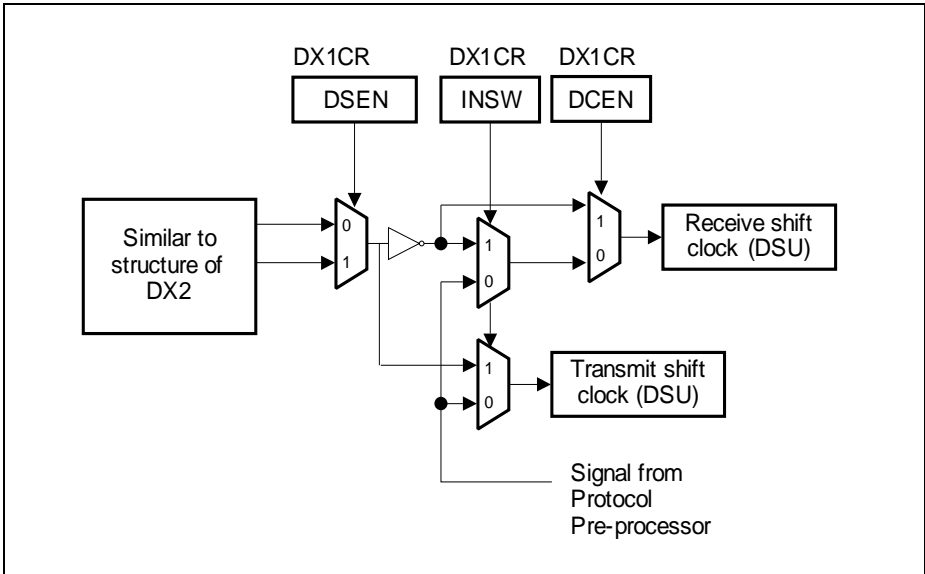


Figure 17-12 Delay Compensation Enable in DX1

17.2.3.2 Digital Filter

The digital filter can be enabled to reduce noise on the input signals. Before being filtered, the input signal becomes synchronized to f_{PERIPH} . If the filter is disabled, signal DXnS corresponds to the synchronized input signal. If the filter is enabled, pulses shorter than one filter sampling period are suppressed in signal DXnS. After an edge of the synchronized input signal, signal DXnS changes to the new value if two consecutive samples of the new value have been detected.

In order to adapt the filter sampling period to different applications, it can be programmed. The first possibility is the system frequency f_{PERIPH} . Longer pulses can be suppressed if the fractional divider output frequency f_{FD} is selected. This frequency is programmable in a wide range and can also be used to determine the baud rate of the data transfers.

In addition to the synchronization delay of 2-3 periods of f_{PERIPH} , an enabled filter adds a delay of up to two filter sampling periods between the selected input and signal DXnS.

17.2.3.3 Edge Detection

The synchronized (and optionally filtered) signal DXnS can be used as input to the data shift unit and is also an input to the selected protocol pre-processor. If the protocol pre-processor does not use the DXnS signal for protocol-specific handling, DXnS can be

Universal Serial Interface Channel (USIC)

used for other tasks, e.g. to control data transmissions in master mode (a data word can be tagged valid for transmission, see chapter about data buffering).

A programmable edge detection indicates that the desired event has occurred by activating the trigger signal DXnT (introducing a delay of one period of f_{PERIPH} before a reaction to this event can take place).

17.2.3.4 Selected Input Monitoring

The selected input signal of each input stage has been made available with the signals DXnINS. These signals can be used in the system to trigger other actions, e.g. to generate interrupts.

17.2.3.5 Loop Back Mode

The USIC transmitter output signals can be connected to the corresponding receiver inputs of the same communication channel in loop back mode. Therefore, the input “G” of the input stages that are needed for the selected protocol have to be selected. In this case, drivers for ASC, SSC, and IIS can be evaluated on-chip without the connections to port pins. Data transferred by the transmitter can be received by the receiver as if it would have been sent by another communication partner.

17.2.4 Operating the Baud Rate Generator

The following blocks can be configured to operate the baud rate generator, see also [Figure 17-2](#).

17.2.4.1 Fractional Divider

The fractional divider generates its output frequency f_{FD} by either dividing the input frequency f_{PERIPH} by an integer factor n or by multiplication of $n/1024$. It has two operating modes:

- Normal divider mode ($\text{FDR.DM} = 01_{\text{B}}$):
In this mode, the output frequency f_{FD} is derived from the input clock f_{PERIPH} by an integer division by a value between 1 and 1024. The division is based on a counter FDR.RESULT that is incremented by 1 with f_{PERIPH} . After reaching the value 3FF_{H} , the counter is loaded with FDR.STEP and then continues counting. In order to achieve $f_{\text{FD}} = f_{\text{PERIPH}}$, the value of STEP has to be programmed with 3FF_{H} .
The output frequency in normal divider mode is defined by the equation:

(17.1)

$$f_{\text{FD}} = f_{\text{PERIPH}} \times \frac{1}{n} \quad \text{with } n = 1024 - \text{STEP}$$

Universal Serial Interface Channel (USIC)

- Fractional divider mode (FDR.DM = 10_B):
In this mode, the output frequency f_{FD} is derived from the input clock f_{PERIPH} by a fractional multiplication of $n/1024$ for a value of n between 0 and 1023. In general, the fractional divider mode allows to program the average output clock frequency with a finer granularity than in normal divider mode. Please note that in fractional divider mode f_{FD} can have a maximum period jitter of one f_{PERIPH} period. This jitter is not accumulated over several cycles.
The frequency f_{FD} is generated by an addition of FDR.STEP to FDR.RESULT with f_{PERIPH} . The frequency f_{FD} is based on the overflow of the addition result over $3FF_H$. The output frequency in fractional divider mode is defined by the equation:

(17.2)

$$f_{FD} = f_{PERIPH} \times \frac{n}{1024} \quad \text{with } n = \text{STEP}$$

The output frequency f_{FD} of the fractional divider is selected for baud rate generation by BRG.CLKSEL = 00_B.

17.2.4.2 External Frequency Input

The baud rate can be generated referring to an external frequency input (instead of to f_{PERIPH}) if in the selected protocol the input stage DX1 is not needed (DX1CTR.INSW = 0). In this case, an external frequency input signal at the DX1 input stage can be synchronized and sampled with the system frequency f_{PERIPH} . It can be optionally filtered by the digital filter in the input stage. This feature allows data transfers with frequencies that can not be generated by the device itself, e.g. for specific audio frequencies.

If BRG.CLKSEL = 10_B, the trigger signal DX1T determines f_{DX1} . In this mode, either the rising edge, the falling edge, or both edges of the input signal can be used for baud rate generation, depending on the configuration of the DX1T trigger event by bit field DX1CTR.CM. The signal MCLK toggles with each trigger event of DX1T.

If BRG.CLKSEL = 11_B, the rising edges of the input signal can be used for baud rate generation. The signal MCLK represents the synchronized input signal DX1S.

Both, the high time and the low time of external input signal must each have a length of minimum 2 periods of f_{PERIPH} to be used for baud rate generation.

17.2.4.3 Divider Mode Counter

The divider mode counter is used for an integer division delivering the output frequency f_{PDIV} . Additionally, two divider stages with a fixed division by 2 provide the output signals MCLK and SCLK with 50% duty cycle. If the fractional divider mode is used, the maximum fractional jitter of 1 period of f_{PERIPH} can also appear in these signals. The output frequencies of this divider is controlled by register BRG.

Universal Serial Interface Channel (USIC)

In order to define a frequency ratio between the master clock MCLK and the shift clock SCLK, the divider stage for MCLK is located in front of the divider by PDIV+1, whereas the divider stage for SCLK is located at the output of this divider.

$$f_{MCLK} = \frac{f_{PIN}}{2} \tag{17.3}$$

$$f_{SCLK} = \frac{f_{PDIV}}{2} \tag{17.4}$$

In the case that the master clock is used as reference for external devices (e.g. for IIS components) and a fixed phase relation to SCLK and other timing signals is required, it is recommended to use the MCLK signal as input for the PDIV divider. If the MCLK signal is not used or a fixed phase relation is not necessary, the faster frequency f_{PIN} can be selected as input frequency.

$$f_{PDIV} = f_{PIN} \times \frac{1}{PDIV + 1} \quad \text{if } PPEN = 0 \tag{17.5}$$

$$f_{PDIV} = f_{MCLK} \times \frac{1}{PDIV + 1} \quad \text{if } PPEN = 1$$

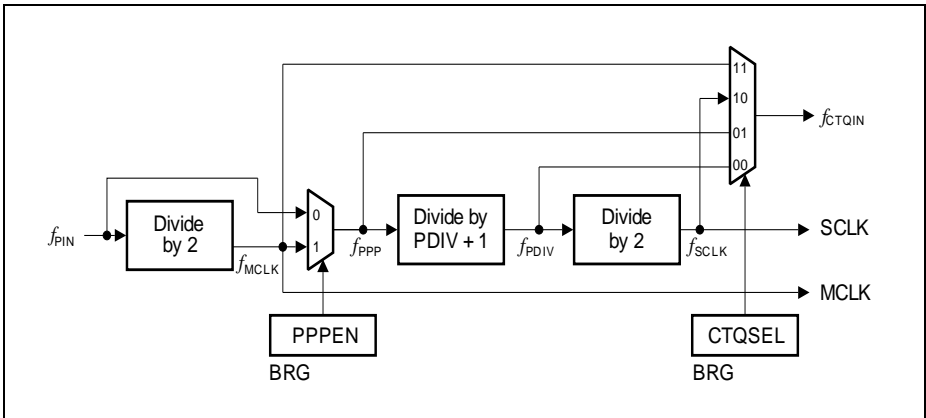


Figure 17-13 Divider Mode Counter

17.2.4.4 Capture Mode Timer

The capture mode timer is used for time interval measurement and is enabled by BRG.TMEN = 1. The timer works independently from the divider mode counter. Therefore, any serial data reception or transmission can continue while the timer is performing timing measurements. The timer counts f_{PPP} periods and stops counting

Universal Serial Interface Channel (USIC)

when it reaches its maximum value. Additionally, a baud rate generator interrupt event is generated (bit PSR.BRGIF becomes set).

If an event is indicated by DX0T or DX1T, the actual timer value is captured into bit field CMTR.CTV and the timer restarts from 0. Additionally, a transmit shift interrupt event is generated (bit PSR.TSIF becomes set).

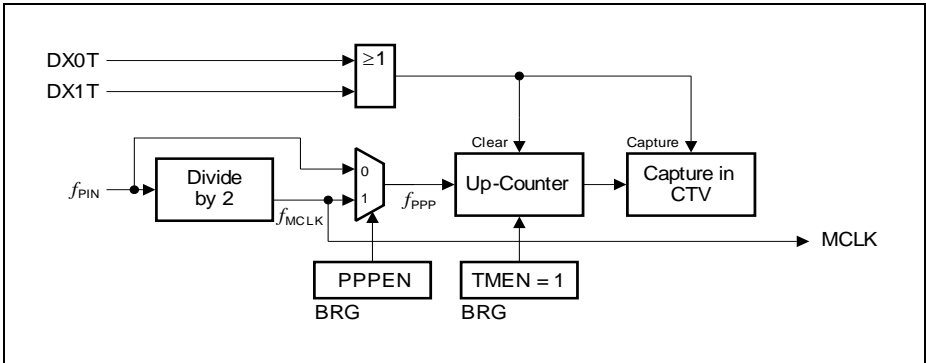


Figure 17-14 Protocol-Related Counter (Capture Mode)

The capture mode timer can be used to measure the baud rate in slave mode before starting or during data transfers, e.g. to measure the time between two edges of a data signal (by DX0T) or of a shift clock signal (by DX1T). The conditions to activate the DXnT trigger signals can be configured in each input stage.

17.2.4.5 Time Quanta Counter

The time quanta counter CTQ associated to the protocol pre-processor allows to generate time intervals for protocol-specific purposes. The length of a time quantum t_q is given by the selected input frequency f_{CTQIN} and the programmed pre-divider value. The meaning of the time quanta depend on the selected protocol, please refer to the corresponding chapters for more protocol-specific information.

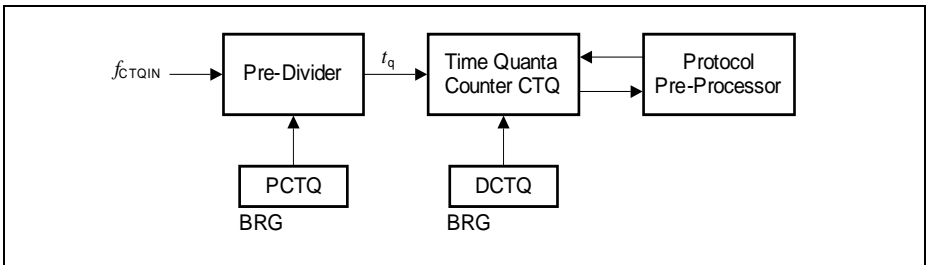


Figure 17-15 Time Quanta Counter

17.2.4.6 Master and Shift Clock Output Configuration

The master clock output signal MCLKOUT available at the corresponding output pin can be configured in polarity. The MCLK signal can be generated for each protocol in order to provide a kind of higher frequency time base compared to the shift clock.

The configuration mechanism of the master clock output signal MCLKOUT ensures that no shortened pulses can occur. Each MCLK period consists of two phases, an active phase, followed by a passive phase. The polarity of the MCLKOUT signal during the active phase is defined by the inverted level of bit BRG.MCLKCFG, evaluated at the start of the active phase. The polarity of the MCLKOUT signal during the passive phase is defined by bit BRG.MCLKCFG, evaluated at the start of the passive phase. If bit BRG.MCLKCFG is programmed with another value, the change is taken into account with the next change between the phases. This mechanism ensures that no shorter pulses than the length of a phase occur at the MCLKOUT output. In the example shown in [Figure 17-16](#), the value of BRG.MCLKCFG is changed from 0 to 1 during the passive phase of MCLK period 2.

The generation of the MCLKOUT signal is enabled/disabled by the protocol pre-processor, based on bit PCR.MCLK. After this bit has become set, signal MCLKOUT is generated with the next active phase of the MCLK period. If PCR.MCLK = 0 (MCLKOUT generation disabled), the level for the passive phase is also applied for active phase.

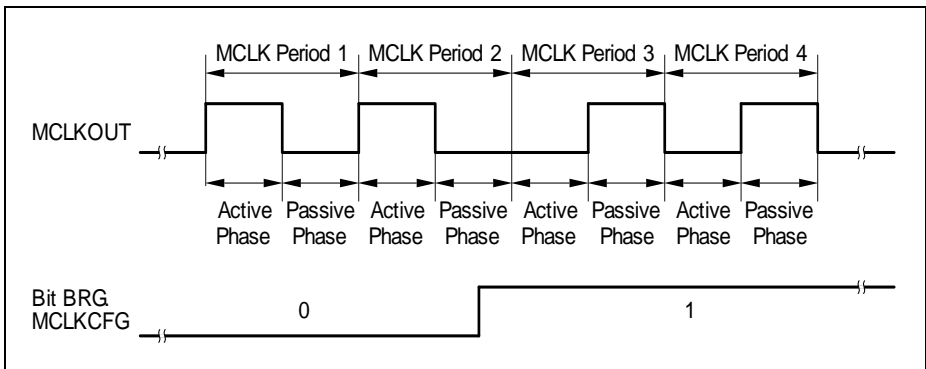


Figure 17-16 Master Clock Output Configuration

The shift clock output signal SCLKOUT available at the corresponding output pin can be configured in polarity and additionally, a delay of one period of f_{PDIV} (= half SCLK period) can be introduced. The delay allows to adapt the order of the shift clock edges to the application requirements. If the delay is used, it has to be taken into account for the calculation of the signal propagation times and loop delays.

The mechanism for the polarity control of the SCLKOUT signal is similar to the one for MCLKOUT, but based on bit field BRG.SCLKCFG. The generation of the SCLKOUT

Universal Serial Interface Channel (USIC)

signal is enabled/disabled by the protocol pre-processor. Depending on the selected protocol, the protocol pre-processor can control the generation of the SCLKOUT signal independently of the divider chain, e.g. for protocols without the need of a shift clock available at a pin, the SCLKOUT generation is disabled.

17.2.5 Operating the Transmit Data Path

The transmit data path is based on 16-bit wide transmit shift registers (TSR and TSR[3:0]) and a transmit buffer TBUF. The data transfer parameters like data word length, data frame length, or the shift direction are controlled commonly for transmission and reception by the shift control register SCTR. The transmit control and status register TCSR controls the transmit data handling and monitors the transmit status.

A change of the value of the data shift output signal DOUTx only happens at the corresponding edge of the shift clock input signal. The level of the last data bit of a data word/frame is held constant at DOUTx until the next data word begins with the next corresponding edge of the shift clock.

17.2.5.1 Transmit Buffering

The transmit shift registers can not be directly accessed by software, because they are automatically updated with the value stored in the transmit buffer TBUF if a currently transmitted data word is finished and new data is valid for transmission. Data words can be loaded directly into TBUF by writing to one of the transmit buffer input locations TBUFx (see [Page 17-33](#)) or, optionally, by a FIFO buffer stage (see [Page 17-39](#)).

Universal Serial Interface Channel (USIC)

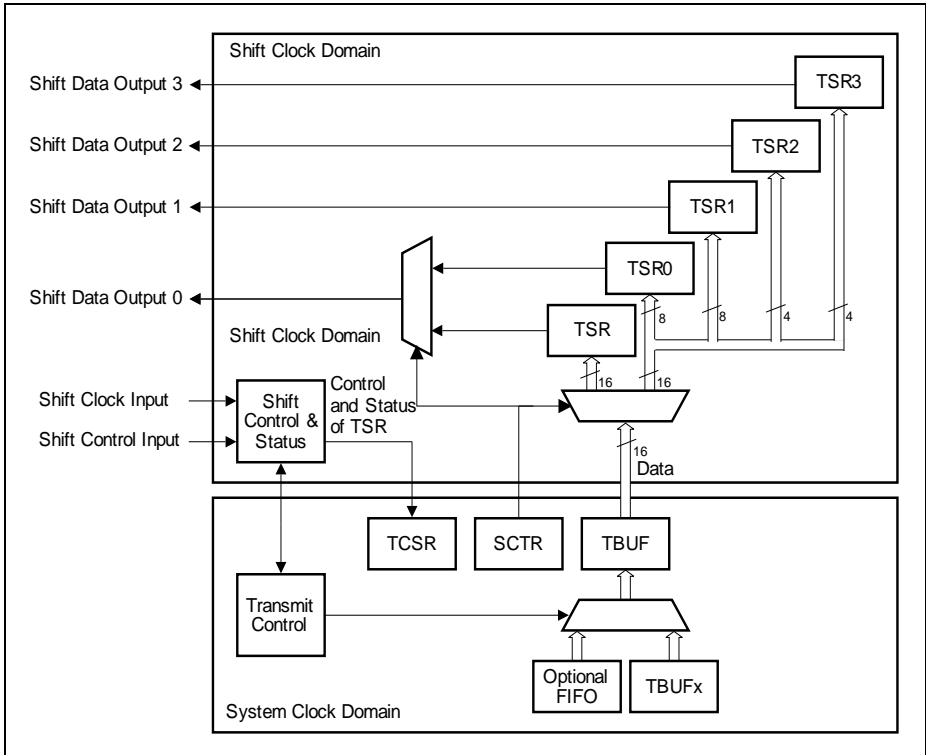


Figure 17-17 Transmit Data Path

17.2.5.2 Transmit Data Shift Mode

The transmit shift data can be selected to be shifted out one, two or four bits at a time through the corresponding number of output lines. This option allows the USIC to support protocols such as the Dual- and Quad-SSC. The selection is done through the bit field DSM in the shift control register SCTR.

Note: The bit field SCTR.DSM controls the data shift mode for both the transmit and receive paths to allow the transmission and reception of data through one to four data lines.

For the shift mode with two or four parallel data outputs, the data word and frame length must be in multiples of two or four respectively. The number of data shifts required to output a specific data word or data frame length is thus reduced by the factor of the number of parallel data output lines. For example, to transmit a 16-bit data word through four output lines, only four shifts are required.

Universal Serial Interface Channel (USIC)

Depending on the shift mode, different transmit shift registers with different bit composition are used as shown in **Table 17-8**. Note that the 'n' in the table denotes the shift number less one, i.e. for the first data shift $n = 0$, the second data shift $n = 1$ and continues until the total number of shifts less one is reached.

For all transmit shift registers, whether the first bit shifted out is the MSB or LSB depends on the setting of SCTR.SDIR.

Table 17-8 Transmit Shift Register Composition

Transmit Shift Registers	Single Data Output (SCTR.DSM = 00_B)	Two Data Outputs (SCTR.DSM = 10_B)	Four Data Outputs (SCTR.DSM = 11_B)
TSR	All data bits	Not used	Not used
TSR0	Not used	Bit $n*2$	Bit $n*4$
TSR1	Not used	Bit $n*2 + 1$	Bit $n*4 + 1$
TSR2	Not used	Not used	Bit $n*4 + 2$
TSR3	Not used	Not used	Bit $n*4 + 3$

17.2.5.3 Transmit Control Information

The transmit control information TCI is a 5-bit value derived from the address x of the written TBUFx or INx input location. For example, writing to TBUF31 generates a TCI of 11111_B.

The TCI can be used as an additional control parameter for data transfers to dynamically change the data word length, the data frame length, or other protocol-specific functions (for more details about this topic, please refer to the corresponding protocol chapters). The way how the TCI is used in different applications can be programmed by the bits WLEMD, FLEMD, SELMD, WAMD and HPCMD in register TCSR. Please note that not all possible settings lead to useful system behavior.

- **Word length control:**
If TCSR.WLEMD = 1, bit field SCTR.WLE is updated with TCI[3:0] if a transmit buffer input location TBUFx is written. This function can be used in all protocols to dynamically change the data word length between 1 and 16 data bits per data word. Additionally, bit TCSR.EOF is updated with TCI[4]. This function can be used in SSC master mode to control the slave select generation to finish data frames. It is recommended to program TCSR.FLEMD = TCSR.SELMD = TCSR.WAMD = TCSR.HPCMD = 0.
- **Frame length control:**
If TCSR.FLEMD = 1, bit field SCTR.FLE[4:0] is updated with TCI[4:0] and SCTR.FLE[5] becomes 0 if a transmit buffer input location TBUFx is written. This function can be used in all protocols to dynamically change the data frame length

Universal Serial Interface Channel (USIC)

between 1 and 32 data bits per data frame. It is recommended to program TCSR.SELMD = TCSR.WLEMD = TCSR.WAMD = TCSR.HPCMD = 0.

- **Select output control:**
If TCSR.SELMD = 1, bit field PCR.CTR[20:16] is updated with TCI[4:0] and PCR.CTR[23:21] becomes 0 if a transmit buffer input location TBUF_x is written. This function can be used in SSC master mode to define the targeted slave device(s). It is recommended to program TCSR.WLEMD = TCSR.FLEMD = TCSR.WAMD = TCSR.HPCMD = 0.
- **Word address control:**
If TCSR.WAMD = 1, bit TCSR.WA is updated with TCI[4] if a transmit buffer input location TBUF_x is written. This function can be used in IIS mode to define if the data word is transmitted on the right or the left channel. It is recommended to program TCSR.WLEMD = TCSR.FLEMD = TCSR.SELMD = TCSR.HPCMD = 0.
- **Hardware Port control:**
If TCSR.HPCMD = 1, bit field SCTR.DSM is updated with TCI[1:0] if a transmit buffer input location TBUF_x is written. This function can be used in SSC protocols to dynamically change the number of data input and output lines to set up for standard, dual and quad SSC formats.
Additionally, bit TCSR.HPCDIR is updated with TCI[2]. This function can be used in SSC protocols to control the pin(s) direction when the hardware port control function is enabled through CCR.HPCEN = 1. It is recommended to program TCSR.FLEMD = TCSR.WLEMD = TCSR.SELMD = TCSR.WAMD = 0.

17.2.5.4 Transmit Data Validation

The data word in the transmit buffer TBUF can be tagged valid or invalid for transmission by bit TCSR.TDV (transmit data valid). A combination of data flow related and event related criteria define whether the data word is considered valid for transmission. A data validation logic checks the start conditions for each data word. Depending on the result of the check, the transmit shift register is loaded with different values, according to the following rules:

- If a USIC channel is the communication master (it defines the start of each data word transfer), a data word transfer can only be started with valid data in the transmit buffer TBUF. In this case, the transmit shift register is loaded with the content of TBUF, that is not changed due to this action.
- If a USIC channel is a communication slave (it can not define the start itself, but has to react), a data word transfer requested by the communication master has to be started independently of the status of the data word in TBUF. If a data word transfer is requested and started by the master, the transmit shift register is loaded at the first corresponding shift clock edge either with the data word in TBUF (if it is valid for transmission) or with the level defined by bit SCTR.PDL (if the content of TBUF has not been valid at the transmission start). In both cases, the content of TBUF is not changed.

Universal Serial Interface Channel (USIC)

The control and status bits for the data validation are located in register TCSR. The data validation is based on the logic blocks shown in **Figure 17-18**.

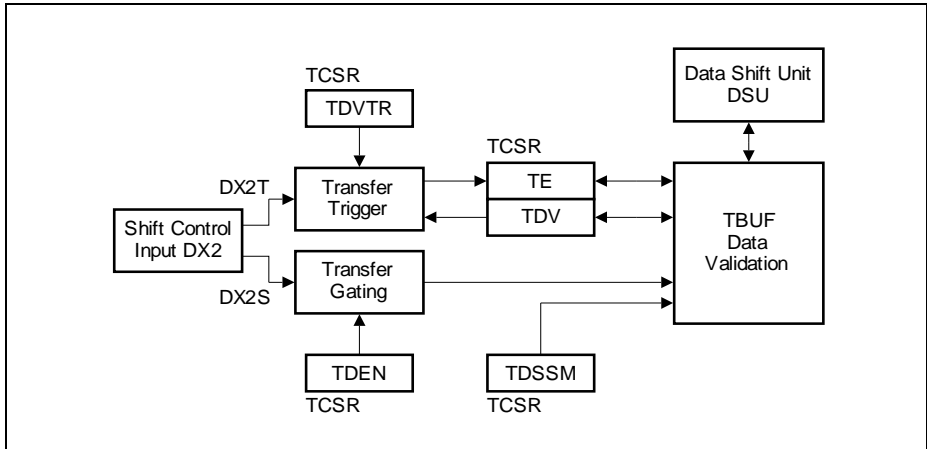


Figure 17-18 Transmit Data Validation

- A transfer gating logic enables or disables the data word transfer from TBUF under software or under hardware control. If the input stage DX2 is not needed for data shifting, signal DX2S can be used for gating purposes. The transfer gating logic is controlled by bit field TCSR.TDEN.
- A transfer trigger logic supports data word transfers related to events, e.g. timer based or related to an input pin. If the input stage DX2 is not needed for data shifting, signal DX2T can be used for trigger purposes. The transfer trigger logic is controlled by bit TCSR.TDVTR and the occurrence of a trigger event is indicated by bit TCSR.TE. For example, this can be used for triggering the data transfer upon receiving the Clear to Send (CTS) signal at DX2 in the RS-232 protocol.
- A data validation logic combining the inputs from the gating logic, the triggering logic and DSU signals. A transmission of the data word located in TBUF can only be started if the gating enables the start, bit TCSR.TDV = 1, and bit TCSR.TE = 1. The content of the transmit buffer TBUF should not be overwritten with new data while it is valid for transmission and a new transmission can start. If the content of TBUF has to be changed, it is recommended to clear bit TCSR.TDV by writing $FMR.MTDV = 10_B$ before updating the data. Bit TCSR.TDV becomes automatically set when TBUF is updated with new data. Another possibility are the interrupts TBI (for ASC and IIC) or RSI (for SSC and IIS) indicating that a transmission has started. While a transmission is in progress, TBUF can be loaded with new data. In this case the user has to take care that an update of the TBUF content takes place before a new transmission starts.

With this structure, the following data transfer functionality can be achieved:

Universal Serial Interface Channel (USIC)

- If bit TCSR.TDSSM = 0, the content of the transmit buffer TBUF is always considered as valid for transmission. The transfer trigger mechanism can be used to start the transfer of the same data word based on the selected event (e.g. on a timer base or an edge at a pin) to realize a kind of life-sign mechanism. Furthermore, in slave mode, it is ensured that always a correct data word is transmitted instead of the passive data level.
- Bit TCSR.TDSSM = 1 has to be programmed to allow word-by-word data transmission with a kind of single-shot mechanism. After each transmission start, a new data word has to be loaded into the transmit buffer TBUF, either by software write actions to one of the transmit buffer input locations TBUFx or by an optional data buffer (e.g. FIFO buffer). To avoid that data words are sent out several times or to allow data handling with an additional data buffer (e.g. FIFO), bit TCSR.TDSSM has to be 1.
- Bit TCSR.TDV becoming automatically set when a new data word is loaded into the transmit buffer TBUF, a transmission start can be requested by a write action of the data to be transmitted to at least the low byte of one of the transmit buffer input locations TBUFx. The additional information TCI can be used to control the data word length or other parameters independently for each data word by a single write access.
- Bit field FMR.MTDV allows software driven modification (set or clear) of bit TCSR.TDV. Together with the gating control bit field TCSR.TDEN, the user can set up the transmit data word without starting the transmission. A possible program sequence could be: clear TCSR.TDEN = 00_B, write data to TBUFx, clear TCSR.TDV by writing FMR.MTDV = 10_B, re-enable the gating with TCSR.TDEN = 01_B and then set TCSR.TDV under software control by writing FMR.MTDV = 01_B.

17.2.6 Operating the Receive Data Path

The receive data path is based on two sets of 16-bit wide receive shift registers RSR0[3:0] and RSR1[3:0] and a receive buffer for each of the set (RBUF0 and RBUF1). The data transfer parameters like data word length, data frame length, or the shift direction are controlled commonly for transmission and reception by the shift control registers.

Register RBUF01SR monitors the status of RBUF0 and RBUF1.

17.2.6.1 Receive Buffering

The receive shift registers cannot be directly accessed by software, but their contents are automatically loaded into the receive buffer registers RBUF0 (or RBUF1 respectively) if a complete data word has been received or the frame is finished. The received data words in RBUF0 or RBUF1 can be read out in the correct order directly from register RBUF or, optionally, from a FIFO buffer stage (see [Page 17-39](#)).

Universal Serial Interface Channel (USIC)

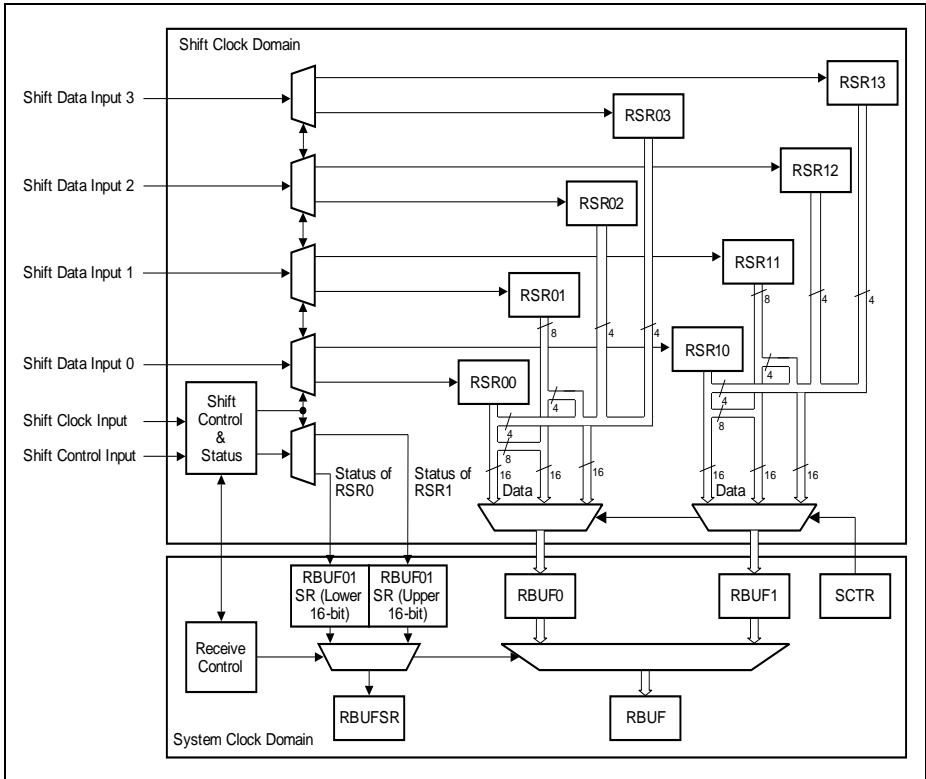


Figure 17-19 Receive Data Path

17.2.6.2 Receive Data Shift Mode

Receive data can be selected to be shifted in one, two or four bits at a time through the corresponding number of input stages and data input lines. This option allows the USIC to support protocols such as the Dual- and Quad-SSC. The selection is done through the bit field DSM in the shift control register SCTR.

Note: The bit field SCTR.DSM controls the data shift mode for both the transmit and receive paths to allow the transmission and reception of data through one to four data lines.

For the shift mode with two or four parallel data inputs, the data word and frame length must be in multiples of two or four respectively. The number of data shifts required to input a specific data word or data frame length is thus reduced by the factor of the

Universal Serial Interface Channel (USIC)

number of parallel data input lines. For example, to receive a 16-bit data word through four input lines, only four shifts are required.

Depending on the shift mode, different receive shift registers with different bit composition are used as shown in [Table 17-8](#). Note that the 'n' in the table denotes the shift number less one, i.e. for the first data shift $n = 0$, the second data shift $n = 1$ and continues until the total number of shifts less one is reached.

For all receive shift registers, whether the first bit shifted in is the MSB or LSB depends on the setting of SCTR.SDIR.

Table 17-9 Receive Shift Register Composition

Receive Shift Registers	Input stage used	Single Data Input (SCTR.DSM = 00 _B)	Two Data Inputs (SCTR.DSM = 10 _B)	Four Data Inputs (SCTR.DSM = 11 _B)
RSR _x 0	DX0	All data bits	Bit $n*2$	Bit $n*4$
RSR _x 1	DX3	Not used	Bit $n*2 + 1$	Bit $n*4 + 1$
RSR _x 2	DX4	Not used	Not used	Bit $n*4 + 2$
RSR _x 3	DX5	Not used	Not used	Bit $n*4 + 3$

17.2.6.3 Baud Rate Constraints

The following baud rate constraints have to be respected to ensure correct data reception and buffering. The user has to take care about these restrictions when selecting the baud rate and the data word length with respect to the module clock frequency f_{PERIPH} .

- A received data word in a receiver shift registers RSR_x[3:0] must be held constant for at least 4 periods of f_{PERIPH} in order to ensure correct loading of the related receiver buffer register RBUF_x.
- The shift control signal has to be constant inactive for at least 5 periods of f_{PERIPH} between two consecutive frames in order to correctly detect the end of a frame.
- The shift control signal has to be constant active for at least 1 period of f_{PERIPH} in order to correctly detect a frame (shortest frame).
- A minimum setup and hold time of the shift control signal with respect to the shift clock signal has to be ensured.

17.2.7 Hardware Port Control

Hardware port control is intended for SSC protocols with half-duplex configurations, where a single port pin is used for both input and output data functions, to control the pin direction through a dedicated hardware interface. All settings in Pn_IOCRy.PC_x, except for the input pull device selection and output driver type (open drain or push-pull), are overruled by the hardware port control.

Universal Serial Interface Channel (USIC)

Input pull device selection is done through the Pn_IOCRy.PC_x as before, while the output driver is fixed to push-pull-only in this mode.

One, two or four port pins can be selected with the hardware port control to support SSC protocols with multiple bi-directional data lines, such as dual- and quad-SSC. This selection and the enable/disable of the hardware port control is done through CCR.HPCEN. The direction of all selected pins is controlled through a single bit SCTR.HPCDIR.

SCTR.HPCDIR is automatically shadowed with the start of each data word to prevent changing of the pin direction in the middle of a data word transfer.

17.2.8 Operating the FIFO Data Buffer

The FIFO data buffers of a USIC module are built in a similar way, with transmit buffer and receive buffer capability for each channel. Depending on the device, the amount of available FIFO buffer area can vary. In the XMC4300, totally 64 buffer entries can be distributed among the transmit or receive FIFO buffers of both channels of the USIC module.

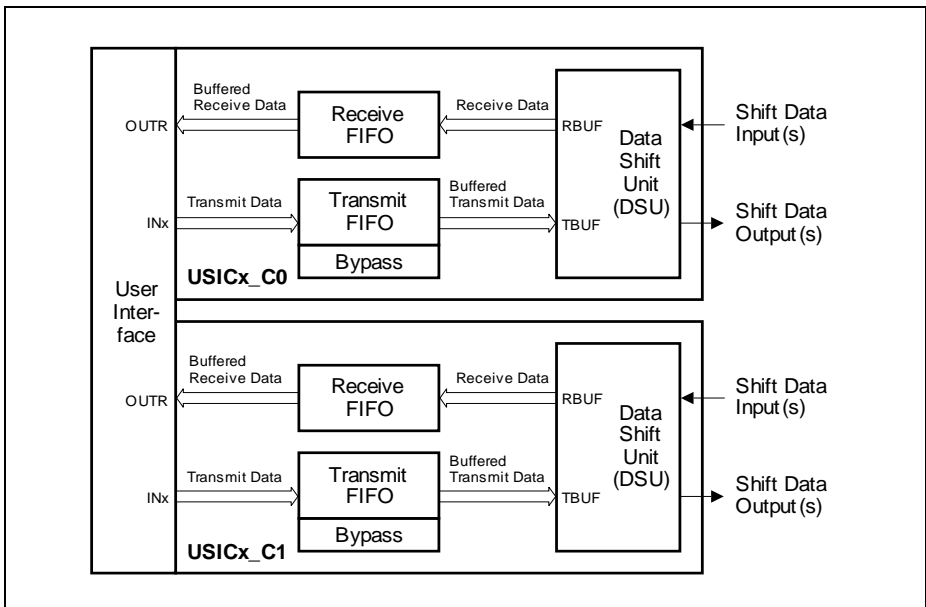


Figure 17-20 FIFO Buffer Overview

In order to operate the FIFO data buffers, the following issues have to be considered:

Universal Serial Interface Channel (USIC)

- FIFO buffer available and selected:
The transmit FIFO buffer and the bypass structure are only available if CCFG.TB = 1, whereas the receive FIFO buffer is only available if CCFG.RB = 1.
It is recommended to configure all buffer parameters while there is no data traffic for this USIC channel and the FIFO mechanism is disabled by TBCTR.SIZE = 0 (for transmit buffer) or RBCTR.SIZE = 0 (for receive buffer). The allocation of a buffer area by writing TBCTR or RBCTR has to be done while the corresponding FIFO buffer is disabled. The FIFO buffer interrupt control bits can be modified independently of data traffic.
- FIFO buffer setup:
The total amount of available FIFO buffer entries limits the length of the transmit and receive buffers for each USIC channel.
- Bypass setup:
In addition to the transmit FIFO buffer, a bypass can be configured as described on [Page 17-50](#).

17.2.8.1 FIFO Buffer Partitioning

If available, the FIFO buffer area consists of a defined number of FIFO buffer entries, each containing a data part and the associated control information (RCI for receive data, TCI for transmit data). One FIFO buffer entry represents the finest granularity that can be allocated to a receive FIFO buffer or a transmit FIFO buffer. All available FIFO buffer entries of a USIC module are located one after the other in the FIFO buffer area. The overall counting starts with FIFO entry 0, followed by 1, 2, etc.

For each USIC module, a certain number of FIFO entries is available, that can be allocated to the channels of the same USIC module. It is not possible to assign FIFO buffer area to USIC channels that are not located within the same USIC module.

For each USIC channel, the size of the transmit and the receive FIFO buffer can be chosen independently. For example, it is possible to allocate the full amount of available FIFO entries as transmit buffer for one USIC channel. Some possible scenarios of FIFO buffer partitioning are shown in [Figure 17-21](#).

Each FIFO buffer consists of a set of consecutive FIFO entries. The size of a FIFO data buffer can only be programmed as a power of 2, starting with 2 entries, then 4 entries, then 8 entries, etc. A FIFO data buffer can only start at a FIFO entry aligned to its size. For example, a FIFO buffer containing n entries can only start with FIFO entry 0, n , $2*n$, $3*n$, etc. and consists of the FIFO entries $[x*n, (x+1)*n-1]$, with x being an integer number (incl. 0). It is not possible to have “holes” with unused FIFO entries within a FIFO buffer, whereas there can be unused FIFO entries between two FIFO buffers.

Universal Serial Interface Channel (USIC)

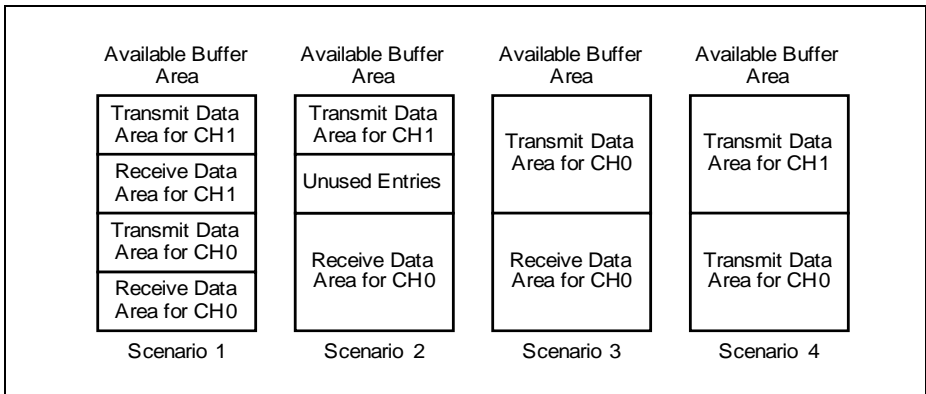


Figure 17-21 FIFO Buffer Partitioning

The data storage inside the FIFO buffers is based on pointers, that are internally updated whenever the data contents of the FIFO buffers have been modified. This happens automatically when new data is put into a FIFO buffer or the oldest data is taken from a FIFO buffer. As a consequence, the user program does not need to modify the pointers for data handling. Only during the initialization phase, the start entry of a FIFO buffer has to be defined by writing the number of the first FIFO buffer entry in the FIFO buffer to the corresponding bit field DPTR in register RBCTR (for a receive FIFO buffer) or TBCTR (for a transmit FIFO buffer) while the related bit field RBCTR.SIZE=0 (or TBCTR.SIZE = 0, respectively). The assignment of buffer entries to a FIFO buffer (regarding to size and pointers) must not be changed by software while the related USIC channel is taking part in data traffic.

17.2.8.2 Transmit Buffer Events and Interrupts

The transmit FIFO buffer mechanism detects the following events, that can lead to interrupts (if enabled):

- Standard transmit buffer event
- Transmit buffer error event

Standard Transmit Buffer Event

The standard transmit buffer event is triggered by the filling level of the transmit buffer (given by TRBSR.TBFLVL) exceeding (TBCTR.LOF = 1) or falling below (TBCTR.LOF = 0)¹⁾ a programmed limit (TBCTR.LIMIT).

¹⁾ If the standard transmit buffer event is used to indicate that new data has to be written to one of the INx locations, TBCTR.LOF = 0 should be programmed.

Universal Serial Interface Channel (USIC)

If the event trigger with TRBSR.STBT feature is disabled (TBCTR.STBTEN = 0), the trigger of the standard transmit buffer event is based on the transition of the fill level from equal to below or above the limit, not the fact of being below or above.

If TBCTR.STBTEN = 1, the transition of the fill level below or above the programmed limit additionally sets TRBSR.STBT. This bit triggers also the standard transmit buffer event whenever there is a transfer data to TBUF event or write data to INx event, depending on TBCTR.LOF setting.

The way TRBSR.STBT is cleared depends on the trigger mode (selected by TBCTR.STBTM). If TBCTR.STBTM = 0, TRBSR.STBT is cleared by hardware when the buffer fill level equals the programmed limit again (TRBSR.TBFLVL = TBCTR.LIMIT). If TBCTR.STBTM = 1, TRBSR.STBT is cleared by hardware when the buffer fill level equals the buffer size (TRBSR.TBFLVL = TBCTR.SIZE).

Note: The flag TRBSR.STBI is set only when the transmit buffer fill level exceeds or falls below the programmed limit (depending on TBCTR.LOF setting). Standard transmit buffer events triggered by TRBSR.STBT does not set the flag.

Figure 17-22 shows examples of the standard transmit buffer event with the different TBCTR.STBTEN and TBCTR.STBTM settings. These examples are meant to illustrate the hardware behaviour and might not always represent real application use cases.

Universal Serial Interface Channel (USIC)

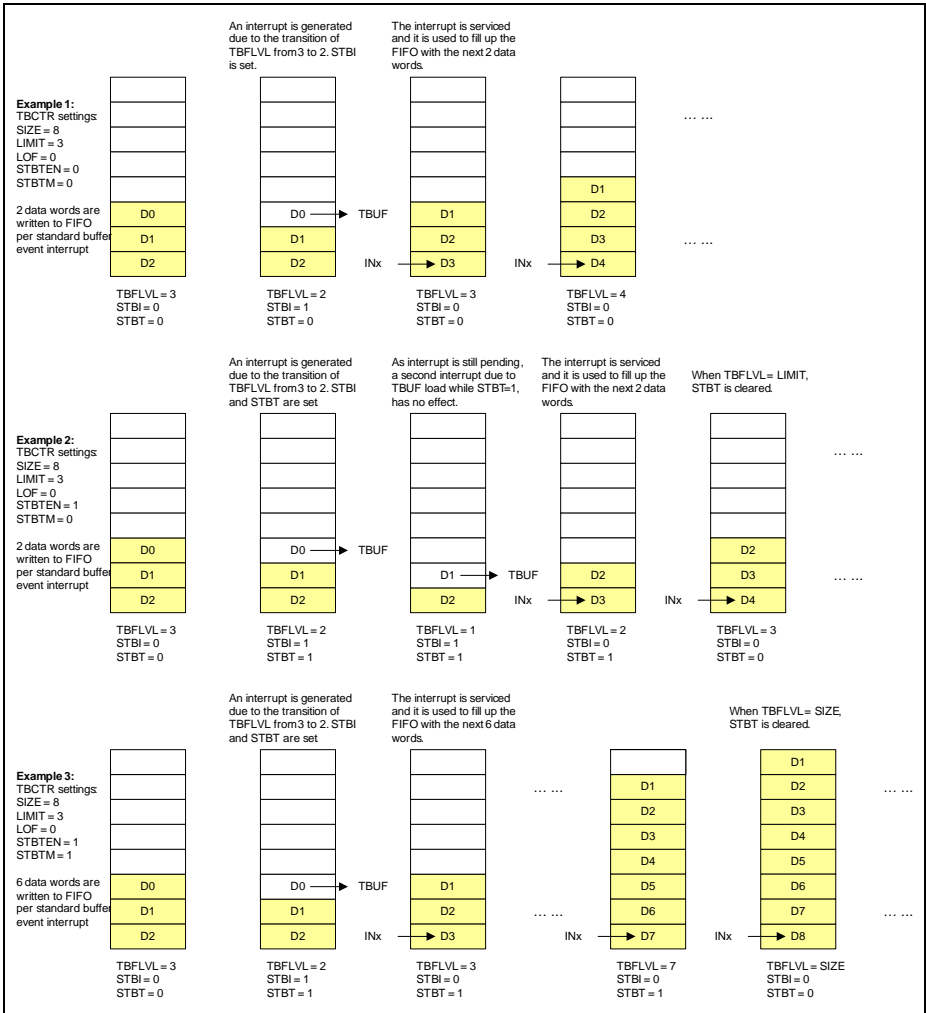


Figure 17-22 Standard Transmit Buffer Event Examples

Transmit Buffer Error Event

The transmit buffer error event is triggered when software has written to a full buffer. The written value is ignored.

Transmit Buffer Events and Interrupt Handling

Figure 17-23 shows the transmit buffer events and interrupts.

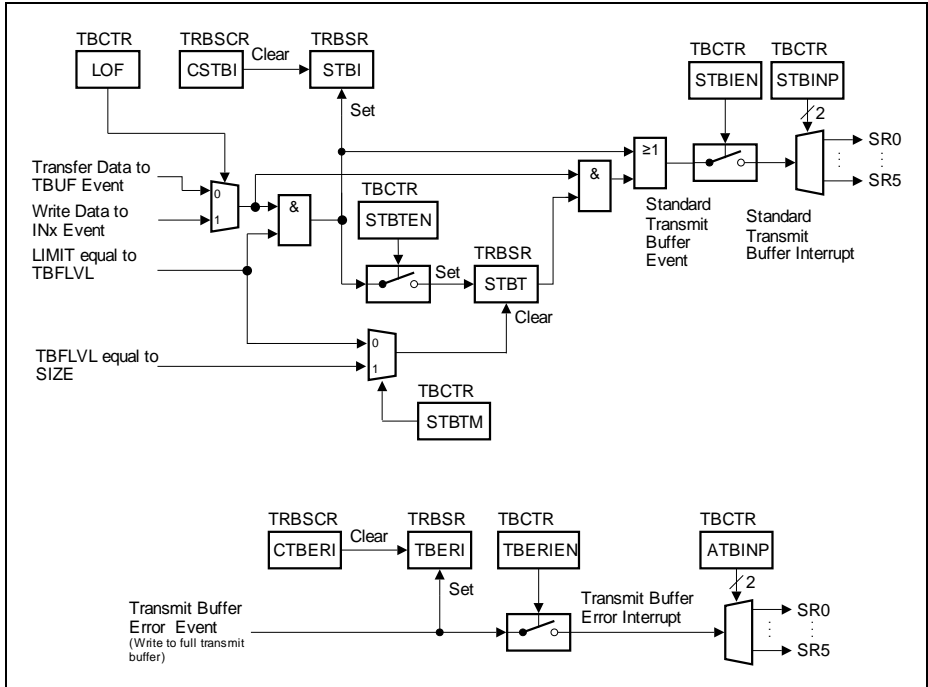


Figure 17-23 Transmit Buffer Events

Table 17-10 shows the registers, bits and bit fields to indicate the transmit buffer events and to control the interrupts related to the transmit FIFO buffers of a USIC channel.

Table 17-10 Transmit Buffer Events and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Standard transmit buffer event	TRBSR. STBI	TRBSCR. CSTBI	TBCTR. STBIEIEN	TBCTR. STBINP
	TRBSR. STBT	Cleared by hardware		
Transmit buffer error event (Write to full transmit buffer)	TRBSR. TBERI	TRBSCR. CTBERI	TBCTR. TBERIEN	TBCTR. ATBINP

17.2.8.3 Receive Buffer Events and Interrupts

The receive FIFO buffer mechanism detects the following events, that can lead to an interrupt (if enabled):

- Standard receive buffer event
- Alternative receive buffer event
- Receive buffer error event

The standard receive buffer event and the alternative receive buffer event can be programmed to two different modes, one referring to the filling level of the receive buffer, the other one related to a bit position in the receive control information RCI of the data word that becomes available in OUTR.

If the interrupt generation refers to the filling level of the receive FIFO buffer, only the standard receive buffer event is used, whereas the alternative receive buffer event is not used. This mode can be selected to indicate that a certain amount of data has been received, without regarding the content of the associated RCI.

If the interrupt generation refers to RCI, the filling level is not taken into account. Each time a new data word becomes available in OUTR, an event is detected. If bit RCI[4] = 0, a standard receive buffer event is signaled, otherwise an alternative receive buffer device (RCI[4] = 1). Depending on the selected protocol and the setting of RBCTR.RCIM, the value of RCI[4] can hold different information that can be used for protocol-specific interrupt handling (see protocol sections for more details).

Standard Receive Buffer Event in Filling Level Mode

In filling level mode (RBCTR.RNM = 0), the standard receive buffer event is triggered by the filling level of the receive buffer (given by TRBSR.RBFLVL) exceeding (RBCTR.LOF = 1) or falling below (RBCTR.LOF = 0) a programmed limit (RBCTR.LIMIT).¹⁾

If the event trigger with bit TRBSR.SRBT feature is disabled (RBCTR.SRBTEN = 0), the trigger of the standard receive buffer event is based on the transition of the fill level from equal to below or above the limit, not the fact of being below or above.

If RBCTR.SRBTEN = 1, the transition of the fill level below or above the programmed limit additionally sets the bit TRBSR.SRBT. This bit also triggers the standard receive buffer event each time there is a data read out event or new data received event, depending on RBCTR.LOF setting.

The way TRBSR.SRBT is cleared depends on the trigger mode (selected by RBCTR.SRBTM). If RBCTR.SRBTM = 0, TRBSR.SRBT is cleared by hardware when the buffer fill level equals the programmed limit again (TRBSR.RBFLVL =

¹⁾ If the standard receive buffer event is used to indicate that new data has to be read from OUTR, RBCTR.LOF = 1 should be programmed.

Universal Serial Interface Channel (USIC)

RBCTR.LIMIT). If RBCTR.SRBTM = 1, TRBSR.SRBT is cleared by hardware when the buffer fill level equals 0 (TRBSR.RBFLVL = 0).

Note: The flag TRBSR.SRBI is set only when the receive buffer fill level exceeds or falls below the programmed limit (depending on RBCTR.LOF setting). Standard receive buffer events triggered by TRBSR.SRBT does not set the flag.

Universal Serial Interface Channel (USIC)

Figure 17-24 shows examples of the standard receive buffer event with the different RBCTR.SRBTEN and RBCTR.SRBTM settings. These examples are meant to illustrate the hardware behaviour and might not always represent real application use cases.

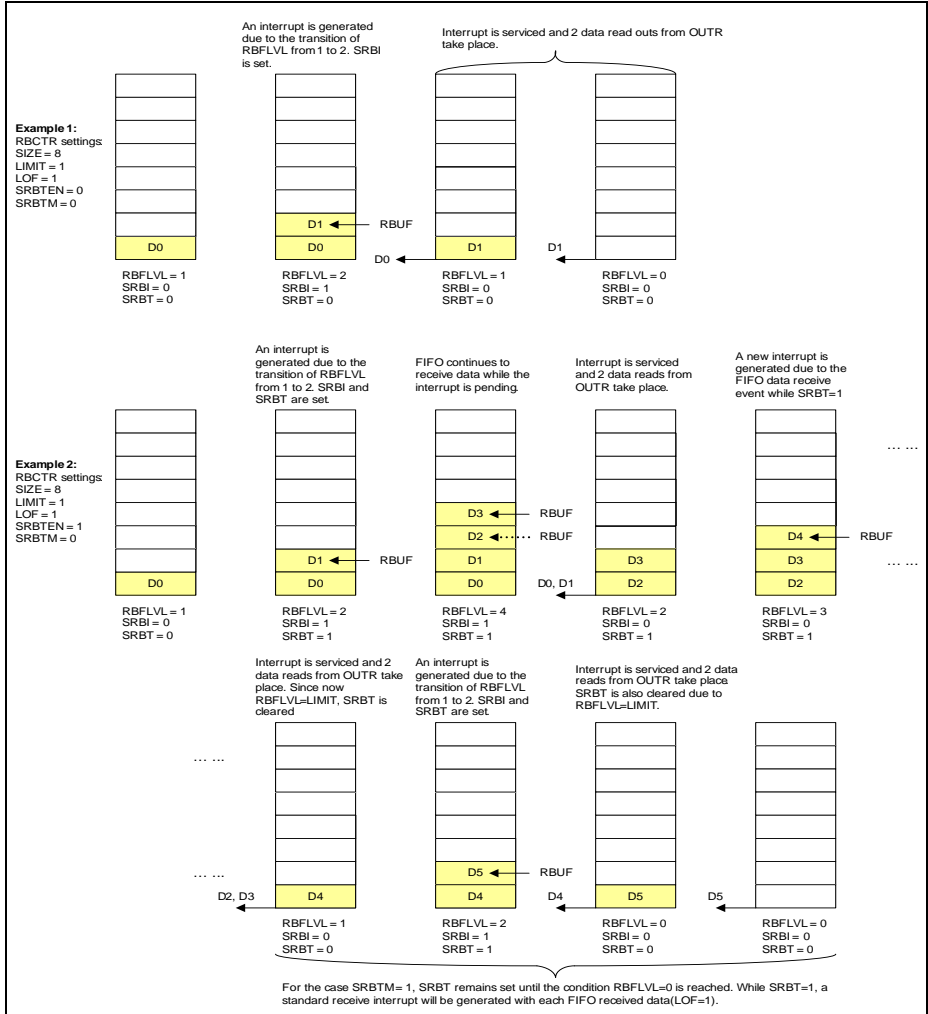


Figure 17-24 Standard Receive Buffer Event Examples

Universal Serial Interface Channel (USIC)

Standard and Alternate Receive Buffer Events in RCI Mode

In RCI mode (RBCTR.RNM = 1), the standard receive buffer event is triggered when the OUTR stage is updated with a new data value with RCI[4] = 0.

If the OUTR stage is updated with a new data value with RCI[4] = 1, an alternate receive buffer event is triggered instead.

Receive Buffer Error Event

The receive buffer error event is triggered if the software reads from an empty buffer, regardless of RBCTR.RNM value. The read data is invalid.

Receive Buffer Events and Interrupt Handling

Figure 17-25 shows the receiver buffer events and interrupts in filling level mode.

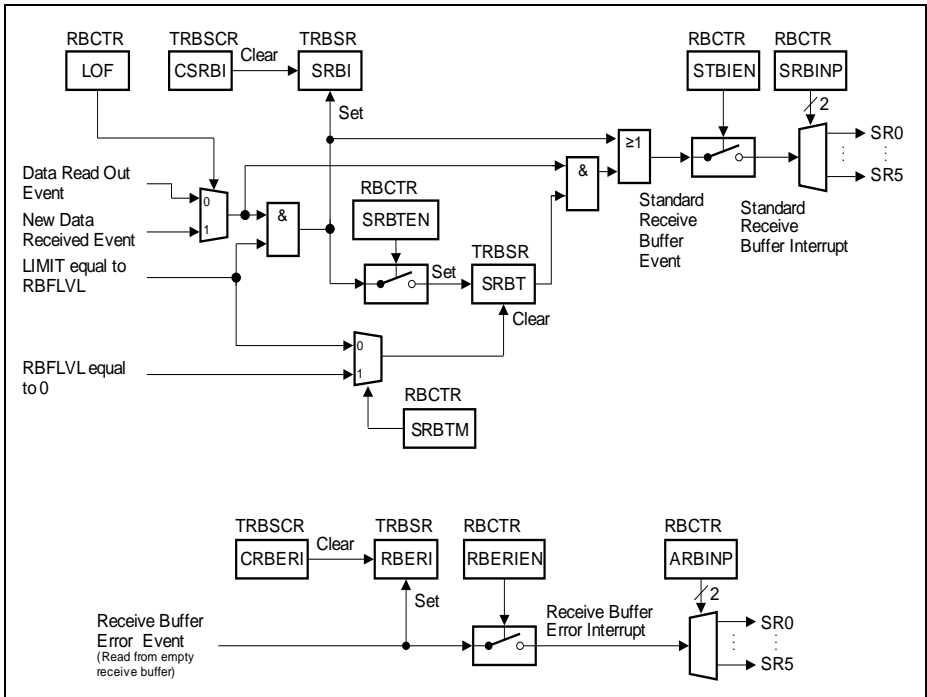


Figure 17-25 Receiver Buffer Events in Filling Level Mode

Universal Serial Interface Channel (USIC)

Figure 17-26 shows the receiver buffer events and interrupts in RCI mode.

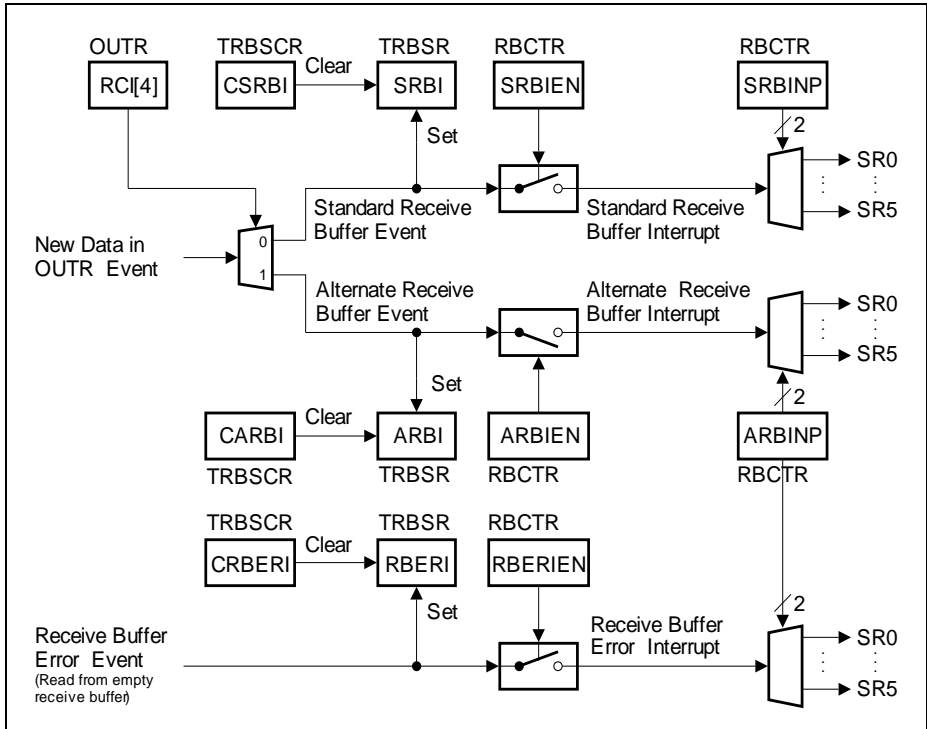


Figure 17-26 Receiver Buffer Events in RCI Mode

Table 17-11 shows the registers, bits and bit fields to indicate the receive buffer events and to control the interrupts related to the receive FIFO buffers of a USIC channel.

Table 17-11 Receive Buffer Events and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Standard receive buffer event	TRBSR. SRBI	TRBSCR. CSRBI	RBCTR. SRBIEN	RBCTR. SRBINP
	TRBSR. SRBT	Cleared by hardware		

Universal Serial Interface Channel (USIC)

Table 17-11 Receive Buffer Events and Interrupt Handling (cont'd)

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Alternative receive buffer event	TRBSR. ARBI	TRBSCR. CARBI	RBCTR. ARBIEN	RBCTR. ARBINP
Receive buffer error event	TRBSR. RBERI	TRBSCR. CRBERI	RBCTR. RBERIEN	RBCTR. ARBINTXDP

17.2.8.4 FIFO Buffer Bypass

The data bypass mechanism is part of the transmit FIFO control block. It allows to introduce a data word in the data stream without modifying the transmit FIFO buffer contents, e.g. to send a high-priority message. The bypass structure consists of a bypass data word of maximum 16 bits in register BYP and some associated control information in register BYPCR. For example, these bits define the word length of the bypass data word and configure a transfer trigger and gating mechanism similar to the one for the transmit buffer TBUF.

The bypass data word can be tagged valid or invalid for transmission by bit BYRCR.BDV (bypass data valid). A combination of data flow related and event related criteria define whether the bypass data word is considered valid for transmission. A data validation logic checks the start conditions for this data word. Depending on the result of the check, the transmit buffer register TBUF is loaded with different values, according to the following rules:

- Data from the transmit FIFO buffer or the bypass data can only be transferred to TBUF if TCSR.TDV = 0 (TBUF is empty).
- Bypass data can only be transferred to TBUF if the bypass is enabled by BYPCR.BDEN or the selecting gating condition is met.
- If the bypass data is valid for transmission and has either a higher transmit priority than the FIFO data or if the transmit FIFO is empty, the bypass data is transferred to TBUF.
- If the bypass data is valid for transmission and has a lower transmit priority than the FIFO buffer that contains valid data, the oldest transmit FIFO data is transferred to TBUF.
- If the bypass data is not valid for transmission and the FIFO buffer contains valid data, the oldest FIFO data is transferred to TBUF.
- If neither the bypass data is valid for transmission nor the transmit FIFO buffer contains valid data, TBUF is unchanged.

The bypass data validation is based on the logic blocks shown in [Figure 17-27](#).

- A transfer gating logic enables or disables the bypass data word transfer to TBUF under software or under hardware control. If the input stage DX2 is not needed for

Universal Serial Interface Channel (USIC)

data shifting, signal DX2S can be used for gating purposes. The transfer gating logic is controlled by bit field BYPCR.BDEN.

- A transfer trigger logic supports data word transfers related to events, e.g. timer based or related to an input pin. If the input stage DX2 is not needed for data shifting, signal DX2T can be used for trigger purposes. The transfer trigger logic is controlled by bit BYPCR.BDVTR.
- A bypass data validation logic combining the inputs from the gating logic, the triggering logic and TCSR.TDV.

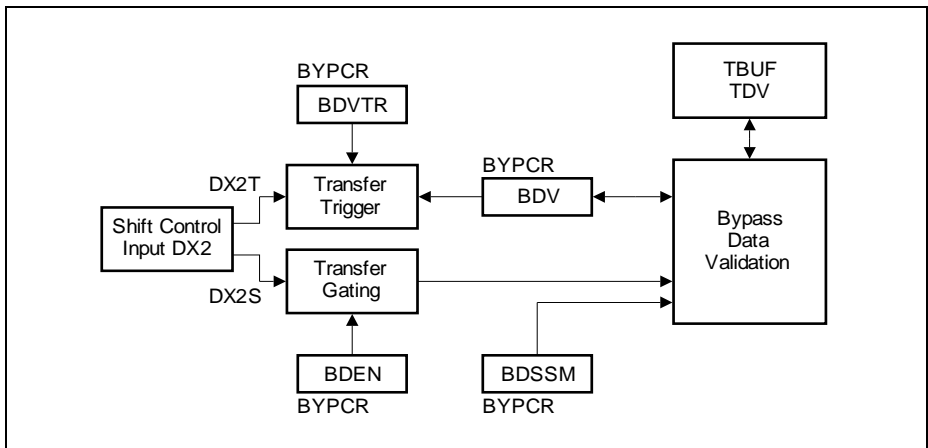


Figure 17-27 Bypass Data Validation

With this structure, the following bypass data transfer functionality can be achieved:

- Bit BYPCR.BDSSM = 1 has to be programmed for a single-shot mechanism. After each transfer of the bypass data word to TBUF, the bypass data word has to be tagged valid again. This can be achieved either by writing a new bypass data word to BYP or by DX2T if BDVTR = 1 (e.g. trigger on a timer base or an edge at a pin).
- Bit BYPCR.BDSSM = 0 has to be programmed if the bypass data is permanently valid for transmission (e.g. as alternative data if the data FIFO runs empty).

17.2.8.5 FIFO Access Constraints

The data in the shared FIFO buffer area is accessed by the hardware mechanisms for data transfer of each communication channel (for transmission and reception) and by software to read out received data or to write data to be transmitted. As a consequence, the data delivery rate can be limited by the FIFO mechanism. Each access by hardware to the FIFO buffer area has priority over a software access, that is delayed in case of an access collision.

In order to avoid data loss and stalling of the CPU due to delayed software accesses, the

Universal Serial Interface Channel (USIC)

baud rate, the word length and the software access mechanism have to be taken into account. Each access to the FIFO data buffer area by software or by hardware takes one period of f_{PERIPH} . Especially a continuous flow of very short, consecutive data words can lead to an access limitation.

17.2.8.6 Handling of FIFO Transmit Control Information

In addition to the transmit data, the transmit control information TCI can be transferred from the transmit FIFO or bypass structure to the USIC channel. Depending on the selected protocol and the enabled update mechanism, some settings of the USIC channel parameters can be modified. The modifications are based on the TCI of the FIFO data word loaded to TBUF or by the bypass control information if the bypass data is loaded into TBUF.

- TCSR.SELMD = 1: update of PCR.CTR[20:16] by FIFO TCI or BYPCR.BSELO with additional clear of PCR.CTR[23:21]
- TCSR.WLEMD = 1: update of SCTR.WLE and TCSR.EOF by FIFO TCI or BYPCR.BWLE (if the WLE information is overwritten by TCI or BWLE, the user has to take care that FLE is set accordingly)
- TCSR.FLEMD = 1: update of SCTR.FLE[4:0] by FIFO TCI or BYPCR.BWLE with additional clear of SCTR.FLE[5]
- TCSR.HPCMD = 1: update of SCTR.DSM and SCTR.HPCDIR by FIFO TCI or BYPCR.BHPC
- TCSR.WAMD = 1: update of TCSR.WA by FIFO TCI[4]

See [Section 17.2.5.3](#) for more details on TCI.

Universal Serial Interface Channel (USIC)

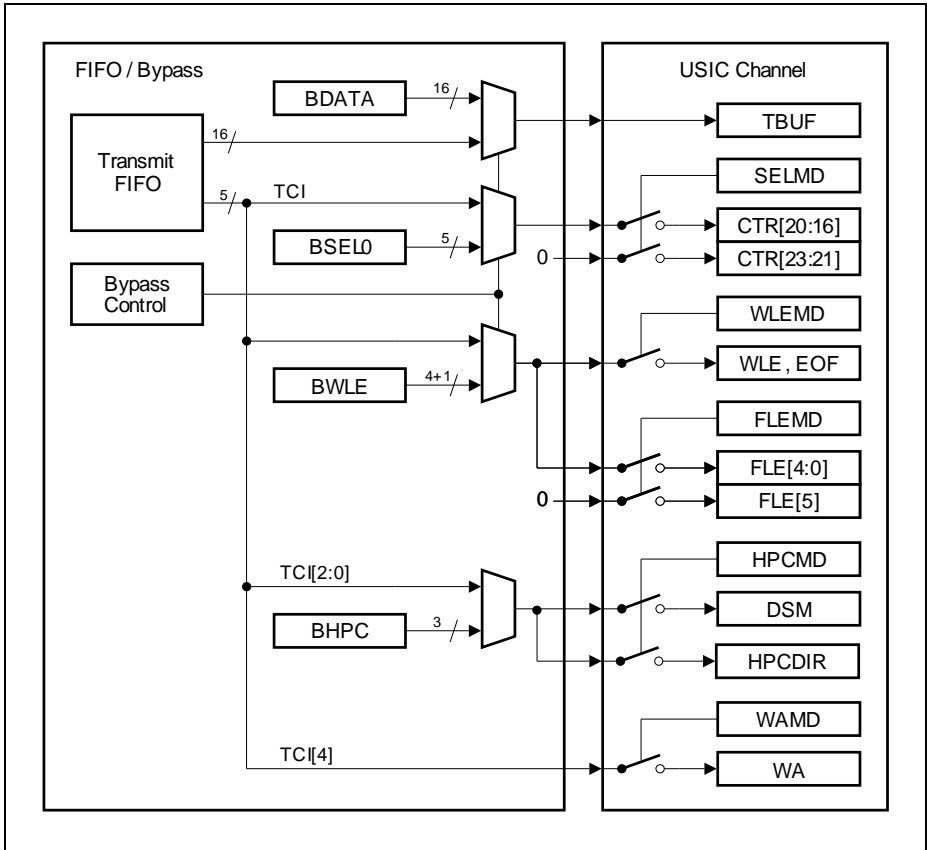


Figure 17-28 TCI Handling with FIFO / Bypass

Universal Serial Interface Channel (USIC)

17.3 Asynchronous Serial Channel (ASC = UART)

The asynchronous serial channel ASC covers the reception and the transmission of asynchronous data frames and provides a hardware LIN support. The receiver and transmitter being independent, frames can start at different points in time for transmission and reception. The ASC mode is selected by $CCR.MODE = 0010_B$ with $CCFG.ASC = 1$ (ASC mode available).

17.3.1 Signal Description

An ASC connection is characterized by the use of a single connection line between a transmitter and a receiver. The receiver input RXD signal is handled by the input stage DX0.

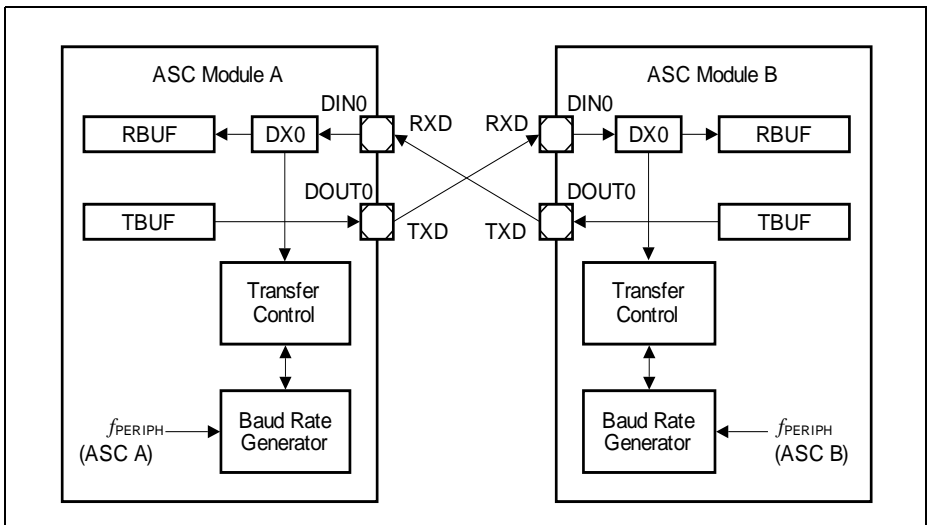


Figure 17-29 ASC Signal Connections for Full-Duplex Communication

For full-duplex communication, an independent communication line is needed for each transfer direction. **Figure 17-29** shows an example with a point-to-point full-duplex connection between two communication partners ASC A and ASC B.

For half-duplex or multi-transmitter communication, a single communication line is shared between the communication partners. **Figure 17-30** shows an example with a point-to-point half-duplex connection between ASC A and ASC B. In this case, the user has to take care that only one transmitter is active at a time. In order to support transmitter collision detection, the input stage DX1 can be used to monitor the level of the transmit line and to check if the line is in the idle state or if a collision occurred. There are two possibilities to connect the receiver input DIN0 to the transmitter output

Universal Serial Interface Channel (USIC)

DOUT0. Communication partner ASC A uses an internal connection with only the transmit pin TXD, that is delivering its input value as RXD to the DX0 input stage for reception and to DX1 to check for transmitter collisions. Communication partner ASC B uses an external connection between the two pins TXD and RXD.

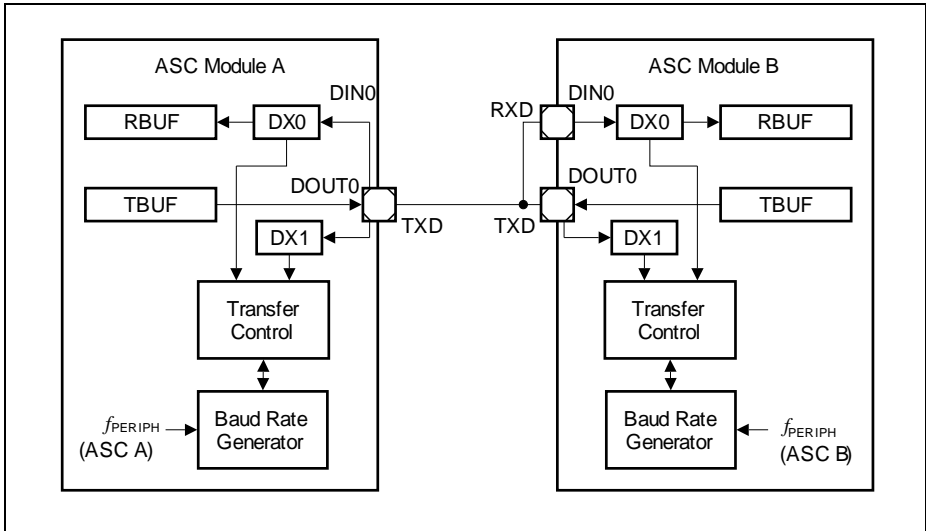


Figure 17-30 ASC Signal Connections for Half-Duplex Communication

17.3.2 Frame Format

A standard ASC frame is shown in [Figure 17-31](#). It consists of:

- An idle time with the signal level 1.
- One start of frame bit (SOF) with the signal level 0.
- A data field containing a programmable number of data bits (1-63).
- A parity bit (P), programmable for either even or odd parity. It is optionally possible to handle frames without parity bit.
- One or two stop bits with the signal level 1.

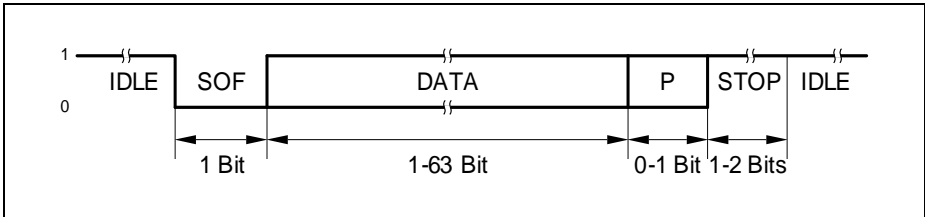


Figure 17-31 Standard ASC Frame Format

The protocol specific bits (SOF, P, STOP) are automatically handled by the ASC protocol state machine and do not appear in the data flow via the receive and transmit buffers.

17.3.2.1 Idle Time

The receiver and the transmitter independently check the respective data input lines (DX0, DX1) for being idle. The idle detection ensures that an SOF bit of a recently enabled ASC module does not collide with an already running frame of another ASC module.

In order to start the idle detection, the user software has to clear bits PSR.RXIDLE and/or PSR.TXIDLE, e.g. before selecting the ASC mode or during operation. If a bit is cleared by software while a data transfer is in progress, the currently running frame transfer is finished normally before starting the idle detection again. Frame reception is only possible if PSR.RXIDLE = 1 and frame transmission is only possible if PSR.TXIDLE = 1. The duration of the idle detection depends on the setting of bit PCR.IDM. In the case that a collision is not possible, the duration can be shortened and the bus can be declared as being idle by setting PCR.IDM = 0.

In the case that the complete idle detection is enabled by PCR.IDM = 1, the data input of DX0 is considered as idle (PSR.RXIDLE becomes set) if a certain number of consecutive passive bit times has been detected. The same scheme applies for the transmitter's data input of DX1. Here, bit PSR.TXIDLE becomes set if the idle condition of this input signal has been detected.

The duration of the complete idle detection is given by the number of programmed data bits per frame plus 2 (in the case without parity) or plus 3 (in the case with parity). The counting of consecutive bit times with 1 level restarts from the beginning each time an edge is found, after leaving a stop mode or if ASC mode becomes enabled.

If the idle detection bits PSR.RXIDLE and/or TXIDLE are cleared by software, the counting scheme is not stopped (no re-start from the beginning). As a result, the cleared bit(s) can become set immediately again if the respective input line still meets the idle criterion.

Please note that the idle time check is based on bit times, so the maximum time can be up to 1 bit time more than programmed value (but not less).

17.3.2.2 Start Bit Detection

The receiver input signal DIN0 (selected signal of input stage DX0) is checked for a falling edge. An SOF bit is detected when a falling edge occurs while the receiver is idle or after the sampling point of the last stop bit. To increase noise immunity, the SOF bit timing starts with the first falling edge that is detected. If the sampled bit value of the SOF is 1, the previous falling edge is considered to be due to noise and the receiver is considered to be idle again.

17.3.2.3 Data Field

The length of the data field (number of data bits) can be programmed by bit field SCTR.FLE. It can vary between 1 and 63 data bits, corresponding to values of SCTR.FLE = 0 to 62 (the value of 63 is reserved and must not be programmed in ASC mode).

The data field can consist of several data words, e.g. a transfer of 12 data bits can be composed of two 8-bit words, with the 12 bits being split into 8-bits of the first word and 4 bits of the second word. The user software has to take care that the transmit data is available in-time, once a frame has been started. If the transmit buffer runs empty during a running data frame, the passive data level (SCTR.PDL) is sent out.

The shift direction can be programmed by SCTR.SDIR. The standard setting for ASC frames with LSB first is achieved with the default setting SDIR = 0.

17.3.2.4 Parity Bit

The ASC allows parity generation for transmission and parity check for reception on frame base. The type of parity can be selected by bit field CCR.PM, common for transmission and reception (no parity, even or odd parity). If the parity handling is disabled, the ASC frame does not contain any parity bit. For consistency reasons, all communication partners have to be programmed to the same parity mode.

After the last data bit of the data field, the transmitter automatically sends out its calculated parity bit if parity generation has been enabled. The receiver interprets this bit as received parity and compares it to its internally calculated one. The received parity bit value and the result of the parity check are monitored in the receiver buffer status registers, RBUFSR and RBUF01SR, as receiver buffer status information. These registers contain bits to monitor a protocol-related argument (PAR) and protocol-related error indication (PERR).

17.3.2.5 Stop Bit(s)

Each ASC frame is completed by 1 or 2 of stop bits with the signal level 1 (same level as the idle level). The number of stop bits is programmable by bit PSR.STPB. A new start bit can be transferred directly after the last stop bit.

17.3.3 Operating the ASC

In order to operate the ASC protocol, the following issues have to be considered:

- **Select ASC mode:**
It is recommended to configure all parameters of the ASC that do not change during run time while $CCR.MODE = 0000_B$. Bit field $SCTR.TRM = 01_B$ has to be programmed. The configuration of the input stages has to be done while $CCR.MODE = 0000_B$ to avoid unintended edges of the input signals and the ASC mode can be enabled by $CCR.MODE = 0010_B$ afterwards.
- **Pin connections:**
Establish a connection of input stage DX0 with the selected receive data input pin (signal DINO) with $DX0CR.INSW = 0$ and configure a transmit data output pin (signal DOUT0). For collision or idle detection of the transmitter, the input stage DX1 has to be connected to the selected transmit output pin, also with $DX1CR.INSW = 0$. Additionally, program $DX2CR.INSW = 0$.
Due to the handling of the input data stream by the synchronous protocol handler, the propagation delay of the synchronization in the input stage has to be considered.
Note that the step to enable the alternate output port functions should only be done after the ASC mode is enabled, to avoid unintended spikes on the output.
- **Bit timing configuration:**
The desired baud rate setting has to be selected, comprising the fractional divider, the baud rate generator and the bit timing. Please note that not all feature combinations can be supported by the application at the same time, e.g. due to propagation delays. For example, the length of a frame is limited by the frequency difference of the transmitter and the receiver device. Furthermore, in order to use the average of samples ($SMD = 1$), the sampling point has to be chosen to respect the signal settling and data propagation times.
- **Data format configuration:**
The word length, the frame length, and the shift direction have to be set up according to the application requirements by programming the register SCTR. If required by the application, the data input and output signals can be inverted.
Additionally, the parity mode has to be configured ($CCR.PM$).

17.3.3.1 Bit Timing

In ASC mode, each bit (incl. protocol bits) is divided into time quanta in order to provide granularity in the sub-bit range to adjust the sample point to the application requirements. The number of time quanta per bit is defined by bit fields $BRG.DCTQ$ and the length of a time quantum is given by $BRG.PCTQ$.

In the example given in [Figure 17-32](#), one bit time is composed of 16 time quanta ($BRG.DCTQ = 15$). It is not recommended to program less than 4 time quanta per bit time.

Universal Serial Interface Channel (USIC)

Bit field PCR.SP determines the position of the sampling point for the bit value. The value of PCR.SP must not be set to a value greater than BRG.DCTQ. It is possible to sample the bit value only once per bit time or to take the average of samples. Depending on bit PCR.SMD, either the current input value is directly sampled as bit value, or a majority decision over the input values sampled at the latest three time quanta is taken into account. The standard ASC bit timing consists of 16 time quanta with sampling after 8 or 9 time quanta with majority decision.

The bit timing setup (number of time quanta and the sampling point definition) is common for the transmitter and the receiver. Due to independent bit timing blocks, the receiver and the transmitter can be in different time quanta or bit positions inside their frames. The transmission of a frame is aligned to the time quanta generation.

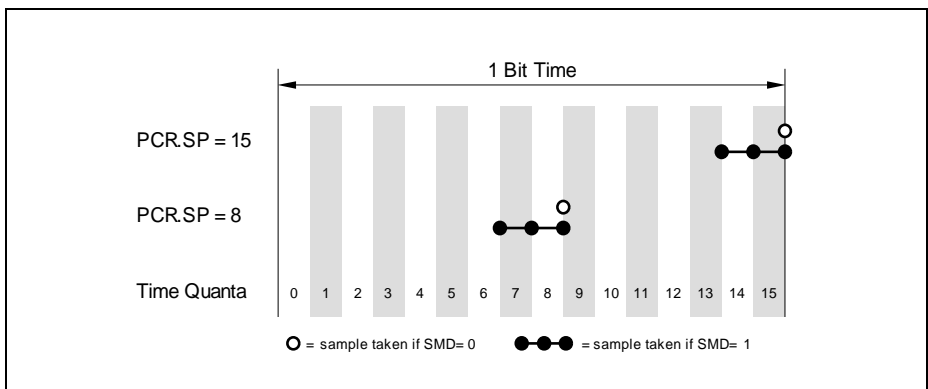


Figure 17-32 ASC Bit Timing

The sample point setting has to be adjusted carefully if collision or idle detection is enabled (via DX1 input signal), because the driver delay and some external delays have to be taken into account. The sample point for the transmit line has to be set to a value where the bit level is stable enough to be evaluated.

If the sample point is located late in the bit time, the signal itself has more time to become stable, but the robustness against differences in the clock frequency of transmitter and receiver decreases.

17.3.3.2 Baud Rate Generation

The baud rate f_{ASC} in ASC mode depends on the number of time quanta per bit time and their timing. The baud rate setting should only be changed while the transmitter and the receiver are idle. The bits in register BRG define the baud rate setting:

- BRG.CTQSEL
to define the input frequency f_{CTQIN} for the time quanta generation

Universal Serial Interface Channel (USIC)

- BRG.PCTQ
to define the length of a time quantum (division of f_{CTQIN} by 1, 2, 3, or 4)
- BRG.DCTQ
to define the number of time quanta per bit time

The standard setting is given by CTQSEL = 00_B ($f_{CTQIN} = f_{PDIV}$) and PPPEN = 0 ($f_{PPP} = f_{PIN}$). Under these conditions, the baud rate is given by:

$$f_{ASC} = f_{PIN} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \times \frac{1}{DCTQ + 1} \quad (17.6)$$

In order to generate slower frequencies, two additional divide-by-2 stages can be selected by CTQSEL = 10_B ($f_{CTQIN} = f_{SCLK}$) and PPPEN = 1 ($f_{PPP} = f_{MCLK}$), leading to:

$$f_{ASC} = \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \times \frac{1}{DCTQ + 1} \quad (17.7)$$

17.3.3.3 Noise Detection

The ASC receiver permanently checks the data input line of the DX0 stage for noise (the check is independent from the setting of bit PCR.SMD). Bit PSR.RNS (receiver noise) becomes set if the three input samples of the majority decision are not identical at the sample point for the bit value. The information about receiver noise gets accumulated over several bits in bit PSR.RNS (it has to be cleared by software) and can trigger a protocol interrupt each time noise is detected if enabled by PCR.RNIEN.

17.3.3.4 Collision Detection

In some applications, such as data transfer over a single data line shared by several sending devices (see [Figure 17-30](#)), several transmitters have the possibility to send on the same data output line TXD. In order to avoid collisions of transmitters being active at the same time or to allow a kind of arbitration, a collision detection has been implemented.

The data value read at the TXD input at the DX1 stage and the transmitted data bit value are compared after the sampling of each bit value. If enabled by PCR.CDEN = 1 and a bit sent is not equal to the bit read back, a collision is detected and bit PSR.COL is set. If enabled, bit PSR.COL = 1 disables the transmitter (the data output lines become 1) and generates a protocol interrupt. The content of the transmit shift register is considered as invalid, so the transmit buffer has to be programmed again.

17.3.3.5 Pulse Shaping

For some applications, the 0 level of transmitted bits with the bit value 0 is not applied at the transmit output during the complete bit time. Instead of driving the original 0 level,

Universal Serial Interface Channel (USIC)

only a 0 pulse is generated and the remaining time quanta of the bit time are driven with 1 level. The length of a bit time is not changed by the pulse shaping, only the signalling is changed.

In the standard ASC signalling scheme, the 0 level is signalled during the complete bit time with bit value 0 (ensured by programming PCR.PL = 000_B). In the case PCR.PL > 000_B, the transmit output signal becomes 0 for the number of time quanta defined by PCR.PL. In order to support correct reception with pulse shaping by the transmitter, the sample point has to be adjusted in the receiver according to the applied pulse length.

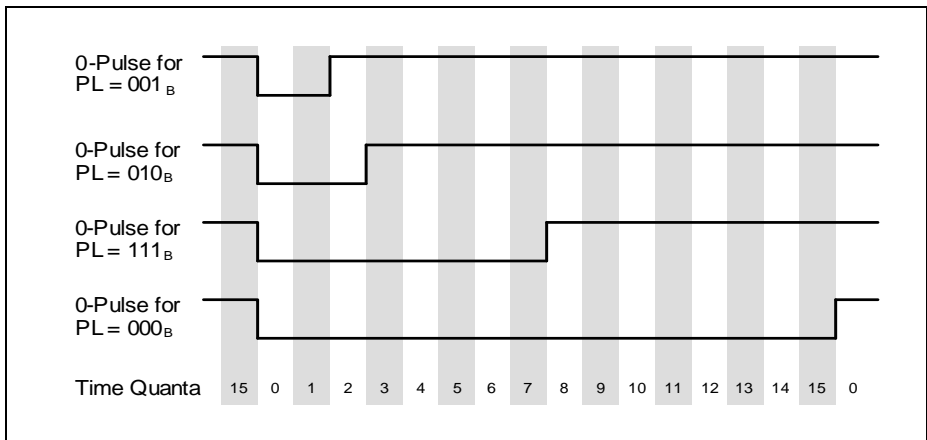


Figure 17-33 Transmitter Pulse Length Control

Figure 17-34 shows an example for the transmission of an 8-bit data word with LSB first and one stop bit (e.g. like for IrDA). The polarity of the transmit output signal has been inverted by SCTR.DOCFG = 01_B.

Universal Serial Interface Channel (USIC)

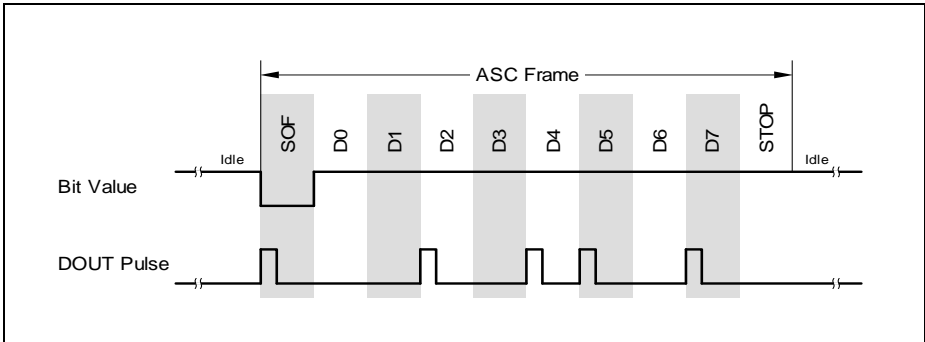


Figure 17-34 Pulse Output Example

17.3.3.6 Automatic Shadow Mechanism

The contents of the protocol control register PCR, as well as bit field SCTR.FLE are internally kept constant while a data frame is transferred by an automatic shadow mechanism (shadowing takes place with each frame start). The registers can be programmed all the time with new settings that are taken into account for the next data frame. During a data frame, the applied (shadowed) setting is not changed, although new values have been written after the start of the data frame.

Bit fields SCTR.WLE and SCTR.SDIR are shadowed automatically with the start of each data word. As a result, a data frame can consist of data words with a different length. It is recommended to change SCTR.SDIR only when no data frame is running to avoid interference between hardware and software.

Please note that the starting point of a data word can be different for a transmitter and a receiver. In order to ensure correct handling, it is recommended to modify SCTR.WLE only while transmitter and receiver are both idle. If the transmitter and the receiver are referring to the same data signal (e.g. in a LIN bus system), SCTR.WLE can be modified while a data transfer is in progress after the RSI event has been detected.

17.3.3.7 End of Frame Control

The number of bits per ASC frame is defined by bit field SCTR.FLE. In order to support different frame length settings for consecutively transmitted frames, this bit field can be modified by hardware. The automatic update mechanism is enabled by TCSR.FLEMD = 1 (in this case, bits TCSR.WLEMD, SELMD, WAMD and HPCMD have to be written with 0).

If enabled, the transmit control information TCI automatically overwrites the bit field TCSR.FLEMD when the ASC frame is started (leading to frames with 1 to 32 data bits). The TCI value represents the written address location of TBUFxx (without additional data

Universal Serial Interface Channel (USIC)

buffer) or INxx (with additional data buffer). With this mechanism, an ASC with 8 data bits is generated by writing a data word to TBUF07 (IN07, respectively).

17.3.3.8 Mode Control Behavior

In ASC mode, the following kernel modes are supported:

- Run Mode 0/1:
Behavior as programmed, no impact on data transfers.
- Stop Mode 0:
Bit PSR.TXIDLE is cleared. A new transmission is not started. A current transmission is finished normally. Bit PSR.RXIDLE is not modified. Reception is still possible. When leaving stop mode 0, bit TXIDLE is set according to PCR.IDM.
- Stop Mode 1:
Bit PSR.TXIDLE is cleared. A new transmission is not started. A current transmission is finished normally. Bit PSR.RXIDLE is cleared. A new reception is not possible. A current reception is finished normally. When leaving stop mode 1, bits TXIDLE and RXIDLE are set according to PCR.IDM.

17.3.3.9 Disabling ASC Mode

In order to switch off ASC mode without any data corruption, the receiver and the transmitter have to be both idle. This is ensured by requesting Stop Mode 1 in register KSCFG. After waiting for the end of the frame, the ASC mode can be disabled.

17.3.3.10 Protocol Interrupt Events

The following protocol-related events are generated in ASC mode and can lead to a protocol interrupt. The collision detection and the transmitter frame finished events are related to the transmitter, whereas the receiver events are given by the synchronization break detection, the receiver noise detection, the format error checks and the end of the received frame.

Please note that the bits in register PSR are not automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- Collision detection:
This interrupt indicates that the transmitted value (DOUT0) does not match with the input value of the DX1 input stage at the sample point of a bit. For more details refer to [Page 17-60](#).
- Transmitter frame finished:
This interrupt indicates that the transmitter has completely finished a frame. Bit PSR.TFF becomes set at the end of the last stop bit. The DOUT0 signal assignment to port pins can be changed while no transmission is in progress.
- Receiver frame finished:
This interrupt indicates that the receiver has completely finished a frame. Bit

Universal Serial Interface Channel (USIC)

PSR.RFF becomes set at the end of the last stop bit. The DIN0 signal assignment to port pins can be changed while no reception is in progress.

- Synchronization break detection:
This interrupt can be used in LIN networks to indicate the reception of the synchronization break symbol (at the beginning of a LIN frame).
- Receiver noise detection:
This interrupt indicates that the input value at the sample point of a bit and at the two time quanta before are not identical.
- Format error:
The bit value of the stop bit(s) is defined as 1 level for the ASC protocol. A format error is signalled if the sampled bit value of a stop bit is 0.

17.3.3.11 Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to ASC frame handling.

- Transmit buffer interrupt TBI:
Bit PSR.TBIF is set after the start of first data bit of a data word. This is the earliest point in time when a new data word can be written to TBUF.
With this event, bit TCSR.TDV is cleared and new data can be loaded to the transmit buffer.
- Transmit shift interrupt TSI:
Bit PSR.TSIF is set after the start of the last data bit of a data word.
- Receiver start interrupt RSI:
Bit PSR.RSIF is set after the sample point of the first data bit of a data word.
- Receiver interrupt RI and alternative interrupt AI:
Bit PSR.RIF is set after the sampling point of the last data bit of a data word if this data word is not directly followed by a parity bit (parity generation disabled or not the last word of a data frame).
If the data word is directly followed by a parity bit (last data word of a data frame and parity generation enabled), bit PSR.RIF is set after the sampling point of the parity bit if no parity error has been detected. If a parity error has been detected, bit PSR.AIF is set instead of bit PSR.RIF.
The first data word of a data frame is indicated by RBUF.SR.SOF = 1 for the received word.
Bit PSR.RIF is set for a receiver interrupt RI with WA = 0. Bit PSR.AIF is set for a alternative interrupt AI with WA = 1.

17.3.3.12 Baud Rate Generator Interrupt Handling

The baud rate generator interrupt indicate that the capture mode timer has reached its maximum value. With this event, the bit PSR.BRGIF is set.

17.3.3.13 Protocol-Related Argument and Error

The protocol-related argument (RBUFSR.PAR) and the protocol-related error (RBUFSR.PERR) are two flags that are assigned to each received data word in the corresponding receiver buffer status registers.

In ASC mode, the received parity bit is monitored by the protocol-related argument and the result of the parity check by the protocol-related error indication (0 = received parity bit equal to calculated parity value). This information being elaborated only for the last received data word of each data frame, both bit positions are 0 for data words that are not the last data word of a data frame or if the parity generation is disabled.

17.3.3.14 Receive Buffer Handling

If a receive FIFO buffer is available (CCFG.RB = 1) and enabled for data handling (RBCTR.SIZE > 0), it is recommended to set RBCTR.RCIM = 11_B in ASC mode. This leads to an indication that the data word has been the first data word of a new data frame if bit OUTR.RCI[0] = 1, a parity error is indicated by OUTR.RCI[4] = 1, and the received parity bit value is given by OUTR.RCI[3].

The standard receive buffer event and the alternative receive buffer event can be used for the following operations in RCI mode (RBCTR.RNM = 1):

- A standard receive buffer event indicates that a data word can be read from OUTR that has been received without parity error.
- An alternative receive buffer event indicates that a data word can be read from OUTR that has been received with parity error.

17.3.3.15 Sync-Break Detection

The receiver permanently checks the DIN0 signal for a certain number of consecutive bit times with 0 level. The number is given by the number of programmed bits per frame (SCTR.FLE) plus 2 (in the case without parity) or plus 3 (in the case with parity). If a 0 level is detected at a sample point of a bit after this event has been found, bit PSR.SBD is set and additionally, a protocol interrupt can be generated (if enabled by PCR.SBIEN = 1). The counting restarts from 0 each time a falling edge is found at input DIN0. This feature can be used for the detection of a synchronization break for slave devices in a LIN bus system (the master does not check for sync break).

For example, in a configuration for 8 data bits without parity generation, bit PCR.SBD is set after at the next sample point at 0 level after 10 complete bit times have elapsed (representing the sample point of the 11th bit time since the first falling edge).

17.3.3.16 Transfer Status Indication

The receiver status can be monitored by flag PSR[9] = BUSY if bit PCR.CTR[16] (receiver status enable RSTEN) is set. In this case, bit BUSY is set during a complete frame reception from the beginning of the start of frame bit to the end of the last stop bit.

Universal Serial Interface Channel (USIC)

The transmitter status can be monitored by flag PSR[9] = BUSY if bit PCR.CTR[17] (transmitter status enable TSTEN) is set. In this case, bit BUSY is set during a complete frame reception from the beginning of the start of frame bit to the end of the last stop bit. If both bits RSTEN and TSTEN are set, flag BUSY indicates the logical OR-combination of the receiver and the transmitter status. If both bits are cleared, flag BUSY is not modified depending on the transfer status (status changes are ignored).

17.3.4 ASC Protocol Registers

In ASC mode, the registers PCR and PSR handle ASC related information.

17.3.4.1 ASC Protocol Control Register

In ASC mode, the PCR register bits or bit fields are defined as described in this section.

PCR

Protocol Control Register [ASC Mode]

(3C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCLK														TSTEN	RSTEN
rw	r													rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL		SP				FFIEN	FEIEN	RNIEN	CDEEN	SBIEEN	IDM	STPB	SMD		
rw		rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SMD	0	rw	<p>Sample Mode</p> <p>This bit field defines the sample mode of the ASC receiver. The selected data input signal can be sampled only once per bit time or three times (in consecutive time quanta). When sampling three times, the bit value shifted in the receiver shift register is given by a majority decision among the three sampled values.</p> <p>0_B Only one sample is taken per bit time. The current input value is sampled.</p> <p>1_B Three samples are taken per bit time and a majority decision is made.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
STPB	1	rw	<p>Stop Bits</p> <p>This bit defines the number of stop bits in an ASC frame.</p> <p>0_B The number of stop bits is 1.</p> <p>1_B The number of stop bits is 2.</p>
IDM	2	rw	<p>Idle Detection Mode</p> <p>This bit defines if the idle detection is switched off or based on the frame length.</p> <p>0_B The bus idle detection is switched off and bits PSR.TXIDLE and PSR.RXIDLE are set automatically to enable data transfers without checking the inputs before.</p> <p>1_B The bus is considered as idle after a number of consecutive passive bit times defined by SCTR.FLE plus 2 (in the case without parity bit) or plus 3 (in the case with parity bit).</p>
SBIEN	3	rw	<p>Synchronization Break Interrupt Enable</p> <p>This bit enables the generation of a protocol interrupt if a synchronization break is detected. The automatic detection is always active, so bit SBD can be set independently of SBIEN.</p> <p>0_B The interrupt generation is disabled.</p> <p>1_B The interrupt generation is enabled.</p>
CDEN	4	rw	<p>Collision Detection Enable</p> <p>This bit enables the reaction of a transmitter to the collision detection.</p> <p>0_B The collision detection is disabled.</p> <p>1_B If a collision is detected, the transmitter stops its data transmission, outputs a 1, sets bit PSR.COL and generates a protocol interrupt. In order to allow data transmission again, PSR.COL has to be cleared by software.</p>
RNIEN	5	rw	<p>Receiver Noise Detection Interrupt Enable</p> <p>This bit enables the generation of a protocol interrupt if receiver noise is detected. The automatic detection is always active, so bit PSR.RNS can be set independently of PCR.RNIEN.</p> <p>0_B The interrupt generation is disabled.</p> <p>1_B The interrupt generation is enabled.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
FEIEN	6	rw	<p>Format Error Interrupt Enable</p> <p>This bit enables the generation of a protocol interrupt if a format error is detected. The automatic detection is always active, so bits PSR.FER0/FER1 can be set independently of PCR.FEIEN.</p> <p>0_B The interrupt generation is disabled. 1_B The interrupt generation is enabled.</p>
FFIEN	7	rw	<p>Frame Finished Interrupt Enable</p> <p>This bit enables the generation of a protocol interrupt if the receiver or the transmitter reach the end of a frame. The automatic detection is always active, so bits PSR.RFF or PSR.TFF can be set independently of PCR.FFIEN.</p> <p>0_B The interrupt generation is disabled. 1_B The interrupt generation is enabled.</p>
SP	[12:8]	rw	<p>Sample Point</p> <p>This bit field defines the sample point of the bit value. The sample point must not be located outside the programmed bit timing ($PCR.SP \leq BRG.DCTQ$).</p>
PL	[15:13]	rw	<p>Pulse Length</p> <p>This bit field defines the length of a 0 data bit, counted in time quanta, starting with the time quantum 0 of each bit time. Each bit value that is a 0 can lead to a 0 pulse that is shorter than a bit time, e.g. for IrDA applications. The length of a bit time is not changed by PL, only the length of the 0 at the output signal.</p> <p>The pulse length must not be longer than the programmed bit timing ($PCR.PL \leq BRG.DCTQ$).</p> <p>This bit field is only taken into account by the transmitter and is ignored by the receiver.</p> <p>000_B The pulse length is equal to the bit length (no shortened 0). 001_B The pulse length of a 0 bit is 2 time quanta. 010_B The pulse length of a 0 bit is 3 time quanta. ... 111_B The pulse length of a 0 bit is 8 time quanta.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
RSTEN	16	rw	<p>Receiver Status Enable</p> <p>This bit enables the modification of flag PSR[9] = BUSY according to the receiver status.</p> <p>0_B Flag PSR[9] is not modified depending on the receiver status.</p> <p>1_B Flag PSR[9] is set during the complete reception of a frame.</p>
TSTEN	17	rw	<p>Transmitter Status Enable</p> <p>This bit enables the modification of flag PSR[9] = BUSY according to the transmitter status.</p> <p>0_B Flag PSR[9] is not modified depending on the transmitter status.</p> <p>1_B Flag PSR[9] is set during the complete transmission of a frame.</p>
MCLK	31	rw	<p>Master Clock Enable</p> <p>This bit enables the generation of the master clock MCLK.</p> <p>0_B The MCLK generation is disabled and the MCLK signal is 0.</p> <p>1_B The MCLK generation is enabled.</p>
0	[30:18]	r	<p>Reserved</p> <p>Returns 0 if read; should be written with 0.</p>

17.3.4.2 ASC Protocol Status Register

In ASC mode, the PSR register bits or bit fields are defined as described in this section. The bits and bit fields in register PSR are not cleared by hardware.

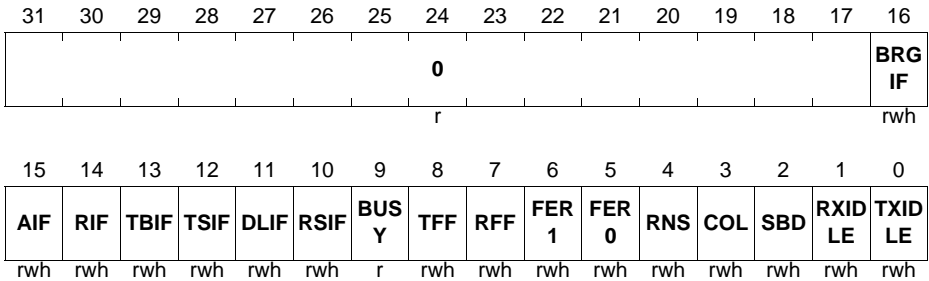
The flags in the PSR register can be cleared by writing a 1 to the corresponding bit position in register PSCR. Writing a 1 to a bit position in PSR sets the corresponding flag, but does not lead to further actions (no interrupt generation). Writing a 0 has no effect. The PSR flags should be cleared by software before enabling a new protocol.

Universal Serial Interface Channel (USIC)

PSR

Protocol Status Register [ASC Mode] (48_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TXIDLE	0	rwh	<p>Transmission Idle</p> <p>This bit shows if the transmit line (DX1) has been idle. A frame transmission can only be started if TXIDLE is set.</p> <p>0_B The transmitter line has not yet been idle. 1_B The transmitter line has been idle and frame transmission is possible.</p>
RXIDLE	1	rwh	<p>Reception Idle</p> <p>This bit shows if the receive line (DX0) has been idle. A frame reception can only be started if RXIDLE is set.</p> <p>0_B The receiver line has not yet been idle. 1_B The receiver line has been idle and frame reception is possible.</p>
SBD	2	rwh	<p>Synchronization Break Detected¹⁾</p> <p>This bit is set if a programmed number of consecutive bit values with level 0 has been detected (called synchronization break, e.g. in a LIN bus system).</p> <p>0_B A synchronization break has not yet been detected. 1_B A synchronization break has been detected.</p>
COL	3	rwh	<p>Collision Detected¹⁾</p> <p>This bit is set if a collision has been detected (with PCR.CDEN = 1).</p> <p>0_B A collision has not yet been detected and frame transmission is possible. 1_B A collision has been detected and frame transmission is not possible.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
RNS	4	rwh	Receiver Noise Detected¹⁾ This bit is set if receiver noise has been detected. 0 _B Receiver noise has not been detected. 1 _B Receiver noise has been detected.
FER0	5	rwh	Format Error in Stop Bit 0¹⁾ This bit is set if a 0 has been sampled in the stop bit 0 (called format error 0). 0 _B A format error 0 has not been detected. 1 _B A format error 0 has been detected.
FER1	6	rwh	Format Error in Stop Bit 1¹⁾ This bit is set if a 0 has been sampled in the stop bit 1 (called format error 1). 0 _B A format error 1 has not been detected. 1 _B A format error 1 has been detected.
RFF	7	rwh	Receive Frame Finished¹⁾ This bit is set if the receiver has finished the last stop bit. 0 _B The received frame is not yet finished. 1 _B The received frame is finished.
TFF	8	rwh	Transmitter Frame Finished¹⁾ This bit is set if the transmitter has finished the last stop bit. 0 _B The transmitter frame is not yet finished. 1 _B The transmitter frame is finished.
BUSY	9	r	Transfer Status BUSY This bit indicates the receiver status (if PCR.RSTEN = 1) or the transmitter status (if PCR.TSTEN = 1) or the logical OR combination of both (if PCR.RSTEN = PCR.TSTEN = 1). 0 _B A data transfer does not take place. 1 _B A data transfer currently takes place.
RSIF	10	rwh	Receiver Start Indication Flag 0 _B A receiver start event has not occurred. 1 _B A receiver start event has occurred.
DLIF	11	rwh	Data Lost Indication Flag 0 _B A data lost event has not occurred. 1 _B A data lost event has occurred.
TSIF	12	rwh	Transmit Shift Indication Flag 0 _B A transmit shift event has not occurred. 1 _B A transmit shift event has occurred.

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
TBIF	13	rwh	Transmit Buffer Indication Flag 0 _B A transmit buffer event has not occurred. 1 _B A transmit buffer event has occurred.
RIF	14	rwh	Receive Indication Flag 0 _B A receive event has not occurred. 1 _B A receive event has occurred.
AIF	15	rwh	Alternative Receive Indication Flag 0 _B An alternative receive event has not occurred. 1 _B An alternative receive event has occurred.
BRGIF	16	rwh	Baud Rate Generator Indication Flag 0 _B A baud rate generator event has not occurred. 1 _B A baud rate generator event has occurred.
0	[31:17]	r	Reserved Returns 0 if read; should be written with 0.

1) This status bit can generate a protocol interrupt (see [Page 17-22](#)). The general interrupt status flags are described in the general interrupt chapter.

17.3.5 Hardware LIN Support

In order to support the LIN protocol, bit TCSR.FLEMD = 1 should be set for the master. For slave devices, it can be cleared and the fixed number of 8 data bits has to be set (SCTR.FLE = 7_H). For both, master and slave devices, the parity generation has to be switched off (CCR.PM = 00_B) and transfers take place with LSB first (SCTR.SDIR = 0) and 1 stop bit (PCR.STPB = 0).

The Local Interconnect Network (LIN) data exchange protocol contains several symbols that can all be handled in ASC mode. Each single LIN symbol represents a complete ASC frame. The LIN bus is a master-slave bus system with a single master and multiple slaves (for the exact definition please refer to the official LIN specification).

A complete LIN frame contains the following symbols:

- Synchronization break:

The master sends a synchronization break to signal the beginning of a new frame. It contains at least 13 consecutive bit times at 0 level, followed by at least one bit time at 1 level (corresponding to 1 stop bit). Therefore, TBUF11 if the transmit buffer is used, (or IN11 if the FIFO buffer is used) has to be written with 0 (leading to a frame with SOF followed by 12 data bits at 0 level).

A slave device shall detect 11 consecutive bit times at 0 level, which done by the synchronization break detection. Bit PSR.SBD is set if such an event is detected and a protocol interrupt can be generated. Additionally, the received data value of 0 appears in the receive buffer and a format error is signaled.

Universal Serial Interface Channel (USIC)

If the baud rate of the slave has to be adapted to the master, the baud rate measurement has to be enabled for falling edges by setting $BRG.TMEN = 1$, $DX0CR.CM = 10_H$ and $DX1CR.CM = 00_H$ before the next symbol starts.

- Synchronization byte:
The master sends this symbol after writing the data value 55_H to TBUF07 (or IN07). A slave device can either receive this symbol without any further action (and can discard it) or it can use the falling edges for baud rate measurement. Bit $PSR.TSIF = 1$ (with optionally the corresponding interrupt) indicates the detection of a falling edge and the capturing of the elapsed time since the last falling edge in $CMTR.CTV$. Valid captured values can be read out after the second, third, fourth and fifth activation of $TSIF$. After the fifth activation of $TSIF$ within this symbol, the baud rate detection can be disabled ($BRG.TMEN = 0$) and $BRG.PDIV$ can be programmed with the captured $CMTR.CTV$ value divided by twice the number of time quanta per bit (assuming $BRG.PCTQ = 00_B$).
- Other symbols:
The other symbols of a LIN frame can be handled with ASC data frames without specific actions.

If LIN frames should be sent out on a frame base by the LIN master, the input $DX2$ can be connected to external timers to trigger the transmit actions (e.g. the synchronization break symbol has been prepared but is started if a trigger occurs). Please note that during the baud rate measurement of the ASC receiver, the ASC transmitter of the same USIC channel can still perform a transmission.

17.4 Synchronous Serial Channel (SSC)

The synchronous serial channel SSC covers the data transfer function of an SPI-like module. It can handle reception and transmission of synchronous data frames between a device operating in master mode and at least one device in slave mode. Besides the standard SSC protocol consisting of one input and one output data line, SSC protocols with two (Dual-SSC) or four (Quad-SSC) input/output data lines are also supported. The SSC mode is selected by $CCR.MODE = 0001_B$ with $CCFG.SSC = 1$ (SSC mode is available).

17.4.1 Signal Description

A synchronous SSC data transfer is characterized by a simultaneous transfer of a shift clock signal together with the transmit and/or receive data signal(s) to determine when the data is valid (definition of transmit and sample point).

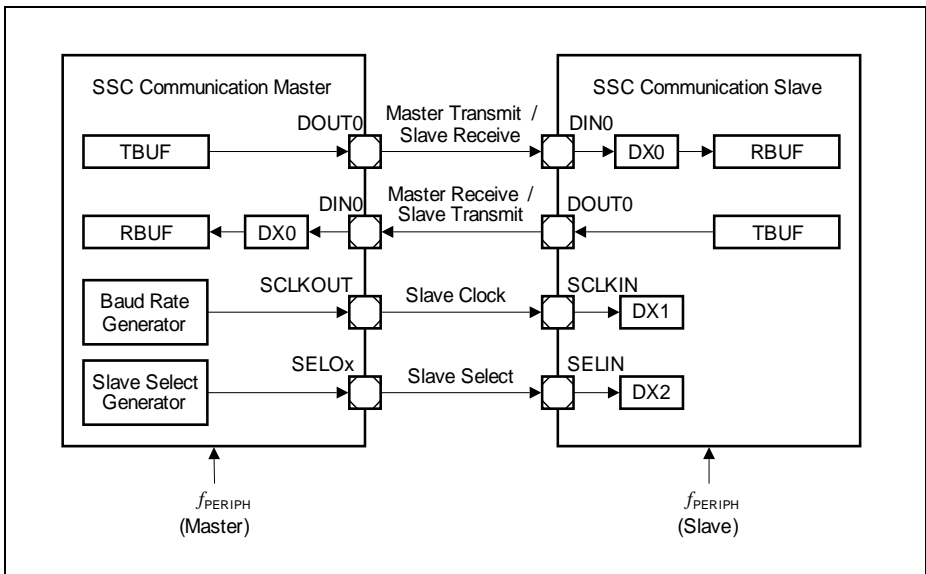


Figure 17-35 SSC Signals for Standard Full-Duplex Communication

In order to explicitly indicate the start and the end of a data transfer and to address more than one slave devices individually, the SSC module supports the handling of slave select signals. They are optional and are not necessarily needed for SSC data transfers. The SSC module supports up to 8 different slave select output signals for master mode operation (named $SELOx$, with $x = 0-7$) and 1 slave select input $SELIN$ for slave mode. In most applications, the slave select signals are active low.

Universal Serial Interface Channel (USIC)

A device operating in master mode controls the start and end of a data frame, as well as the generation of the shift clock and slave select signals. This comprises the baud rate setting for the shift clock and the delays between the shift clock and the slave select output signals. If several SSC modules are connected together, there can be only one SSC master at a time, but several slaves. Slave devices receive the shift clock and optionally a slave select signal(s). For the programming of the input stages DXn please refer to [Page 17-22](#).

Table 17-12 SSC Communication Signals

SSC Mode	Receive Data	Transmit Data	Shift Clock	Slave Select(s)
Standard SSC Master	MRST ¹⁾ , input DIN0, handled by DX0	MTSR ²⁾ , Output DOUT0	Output SCLKOUT	Output(s) SELOx
Standard SSC Slave	MTSR, input DIN0, handled by DX0	MRST, Output DOUT0	Input SCLKIN, handled by DX1	input SELIN, handled by DX2
Dual-SSC Master	MRST[1:0], input DIN[1:0], handled by DX0 and DX3	MTSR[1:0], Output DOUT[1:0]	Output SCLKOUT	Output(s) SELOx
Dual-SSC Slave	MTSR[1:0], input DIN[1:0], handled by DX0 and DX3	MRST[1:0], Output DOUT[1:0]	Input SCLKIN, handled by DX1	input SELIN, handled by DX2
Quad-SSC Master	MRST[3:0], input DIN[3:0], handled by DX0, DX3, DX4 and DX5	MTSR[3:0], Output DOUT[3:0]	Output SCLKOUT	Output(s) SELOx
Quad-SSC Slave	MTSR[3:0], input DIN[3:0], handled by DX0, DX3, DX4 and DX5	MRST[3:0], Output DOUT[3:0]	Input SCLKIN, handled by DX1	input SELIN, handled by DX2

1) MRST = master receive slave transmit, also known as MISO = master in slave out

2) MTSR = master transmit slave receive, also known as MOSI = master out slave in

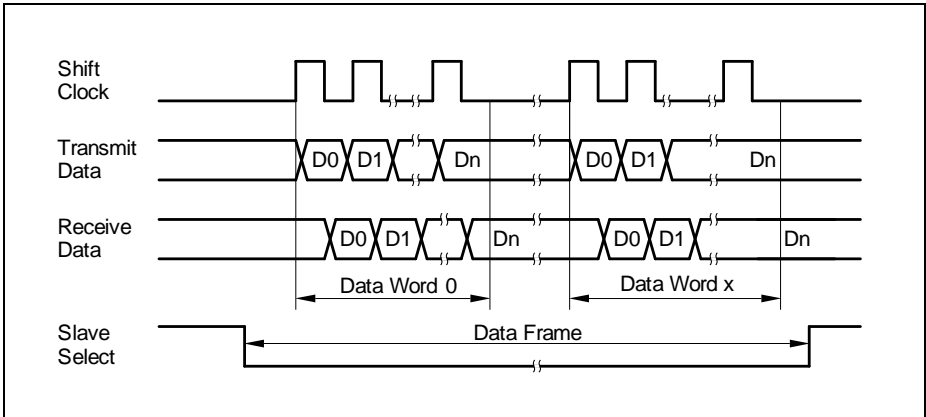


Figure 17-36 4-Wire SSC Standard Communication Signals

17.4.1.1 Transmit and Receive Data Signals

In standard SSC half-duplex mode, a single data line is used, either for data transfer from the master to a slave or from a slave to the master. In this case, MRST and MTSR are connected together, one signal as input, the other one as output, depending on the data direction. The user software has to take care about the data direction to avoid data collision (e.g. by preparing dummy data of all 1s for transmission in case of a wired AND connection with open-drain drivers, by enabling/disabling push/pull output drivers or by switching pin direction with hardware port control enabled). In full-duplex mode, data transfers take place in parallel between the master device and a slave device via two independent data signals MTSR and MRST, as shown in [Figure 17-35](#).

The receive data input signal DIN0 is handled by the input stage DX0. In master mode (referring to MRST) as well as in slave mode (referring to MTSR), the data input signal DIN0 is taken from an input pin. The signal polarity of DOUT0 (data output) with respect to the data bit value can be configured in block DOCFG (data output configuration) by bit field SCTR.DOCFG.

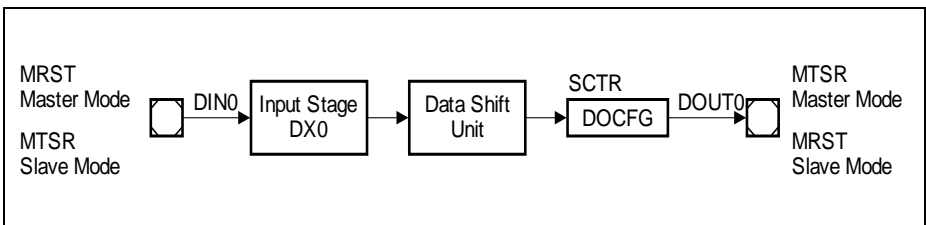


Figure 17-37 SSC Data Signals

Universal Serial Interface Channel (USIC)

For dual- and quad-SSC modes that require multiple input and output data lines to be used, additional input stages, DINx and DOUTx signals need to be set up.

17.4.1.2 Shift Clock Signals

The shift clock signal is handled by the input stage DX1. In slave mode, the signal SCLKIN is received from an external master, so the DX1 stage has to be connected to an input pin. The input stage can invert the received input signal to adapt to the polarity of SCLKIN to the function of the data shift unit (data transmission on rising edges, data reception on falling edges).

In master mode, the shift clock is generated by the internal baud rate generator. The output signal SCLK of the baud rate generator is taken as shift clock input for the data shift unit. The internal signal SCLK is made available for external slave devices by signal SCLKOUT. For complete closed loop delay compensation in a slave mode, SCLKOUT can also take the transmit shift clock from the input stage DX1. The selection is done through the bit BRG.SCLKOSEL. See [Section 17.4.6.3](#).

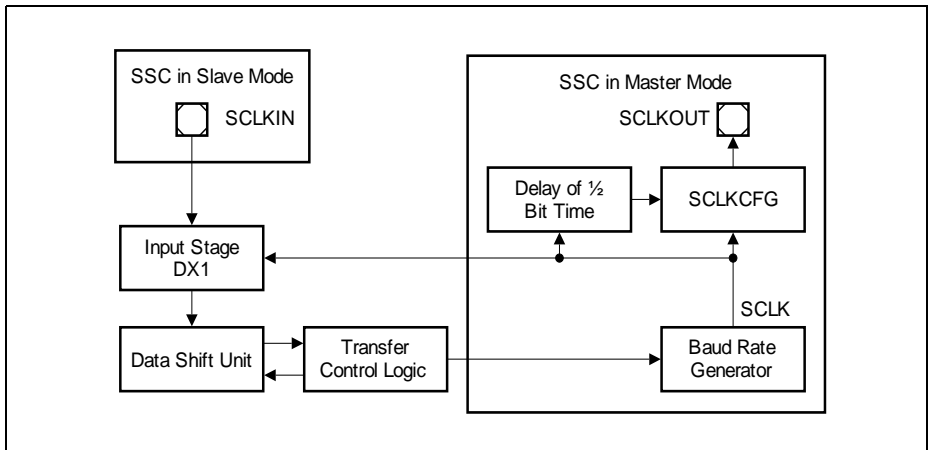


Figure 17-38 SSC Shift Clock Signals

Due to the multitude of different SSC applications, in master mode, there are different ways to configure the shift clock output signal SCLKOUT with respect to SCLK. This is done in the block SCLKCFG (shift clock configuration) by bit field BRG.SCLKCFG, allowing 4 possible settings, as shown in [Figure 17-39](#).

- No delay, no polarity inversion (SCLKCFG = 00_B, SCLKOUT equals SCLK):
The inactive level of SCLKOUT is 0, while no data frame is transferred. The first data bit of a new data frame is transmitted with the first rising edge of SCLKOUT and the first data bit is received in with the first falling edge of SCLKOUT. The last data bit of a data frame is transmitted with the last rising clock edge of SCLKOUT and the last

Universal Serial Interface Channel (USIC)

data bit is received in with the last falling edge of SCLKOUT. This setting can be used in master and in slave mode. It corresponds to the behavior of the internal data shift unit.

- No delay, polarity inversion (SCLKCFG = 01_B):
The inactive level of SCLKOUT is 1, while no data frame is transferred. The first data bit of a new data frame is transmitted with the first falling clock edge of SCLKOUT and the first data bit is received with the first rising edge of SCLKOUT. The last data bit of a data frame is transmitted with the last falling edge of SCLKOUT and the last data bit is received with the last rising edge of SCLKOUT.
This setting can be used in master and in slave mode. For slave mode, bit field DX1CR.DPOL = 1_B has to be programmed.
- SCLKOUT is delayed by 1/2 shift clock period, no polarity inversion (SCLKCFG = 10_B):
The inactive level of SCLKOUT is 0, while no data frame is transferred.
The first data bit of a new data frame is transmitted 1/2 shift clock period before the first rising clock edge of SCLKOUT. Due to the delay, the next data bits seem to be transmitted with the falling edges of SCLKOUT. The last data bit of a data frame is transmitted 1/2 period of SCLKOUT before the last rising clock edge of SCLKOUT. The first data bit is received 1/2 shift clock period before the first falling edge of SCLKOUT. Due to the delay, the next data bits seem to be received with the rising edges of SCLKOUT. The last data bit is received 1/2 period of SCLKOUT before the last falling clock edge of SCLKOUT.
This setting can be used only in master mode and not in slave mode (the connected slave has to provide the first data bit before the first SCLKOUT edge, e.g. as soon as it is addressed by its slave select).
- SCLKOUT is delayed by 1/2 shift clock period, polarity inversion (SCLKCFG = 11_B):
The inactive level of SCLKOUT is 1, while no data frame is transferred.
The first data bit of a new data frame is transmitted 1/2 shift clock period before the first falling clock edge of SCLKOUT. Due to the delay, the next data bits seem to be transmitted with the rising edges of SCLKOUT. The last data bit of a data frame is transmitted 1/2 period of SCLKOUT before the last falling clock edge of SCLKOUT. The first data bit is received 1/2 shift clock period before the first rising edge of SCLKOUT. Due to the delay, the next data bits seem to be received with the falling edges of SCLKOUT. The last data bit is received 1/2 period of SCLKOUT before the last rising clock edge of SCLKOUT.
This setting can be used only in master mode and not in slave mode (the connected slave has to provide the first data bit before the first SCLKOUT edge, e.g. as soon as it is addressed by its slave select).

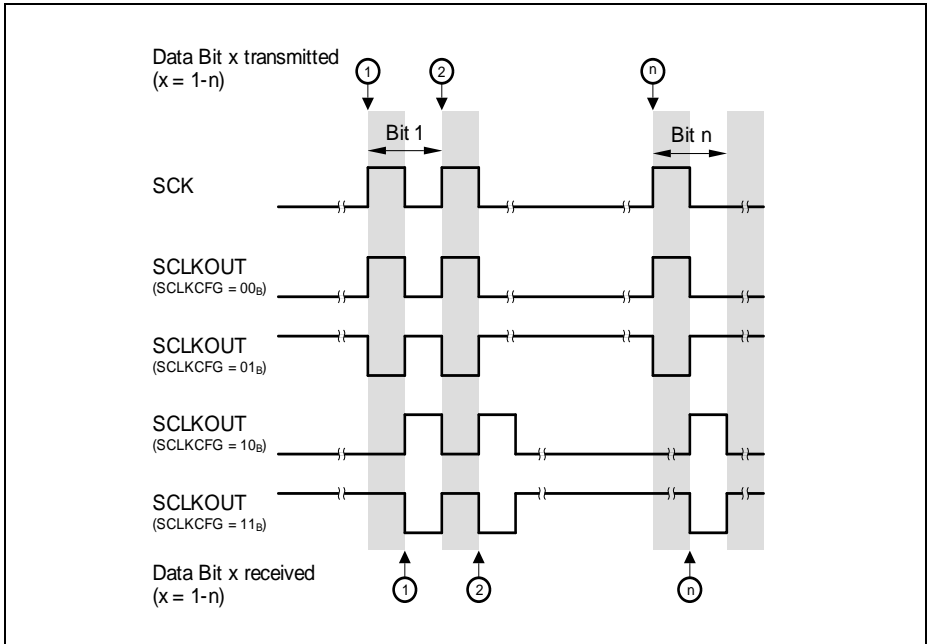


Figure 17-39 SCLKOUT Configuration in SSC Master Mode

Note: If a configuration with delay is selected and a slave select line is used, the slave select delays have to be set up accordingly.

In SSC slave mode, the bit PCR.SLPHSEL can be used to configure the clock phase of the data shift.

- When SLPHSEL = 0_B, the slave SSC transmits data bits with each leading edge of the selected shift clock input (SCLKIN) and receives data bits with each trailing edge of SCLKIN
- When SLPHSEL = 1_B, the slave SSC transmits the first data bit once the selected slave select input (SELIN) becomes active. If SELIN is not used, the DX2 stage has to deliver a 1-level to the data shift unit to shift out the first bit. Subsequent data bits are then transmitted with each trailing edge of SCLKIN. The SSC slave receives all data bits with each leading edge of SCLKIN.

For both settings, the clock polarity is determined by bit 0 of SCLKCFG.

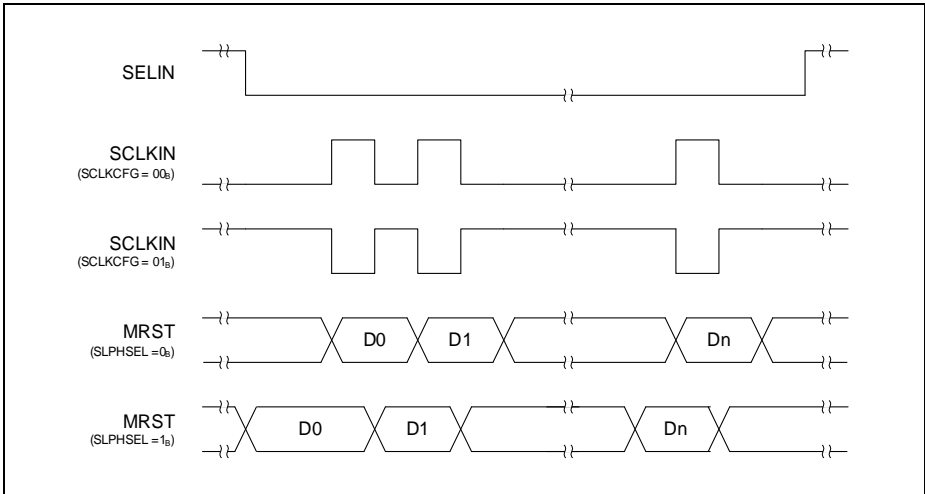


Figure 17-40 SLPHSEL Configuration in SSC Slave Mode

17.4.1.3 Slave Select Signals

The slave select signal is handled by the input stage DX2. In slave mode, the input signal SELIN is received from an external master via an input pin. The input stage can invert the received input signal to adapt the polarity of signal SELIN to the function of the data shift unit (the module internal signals are considered as high active, so a data transfer is only possible while the slave select input of the data shift unit is at 1-level, otherwise, shift clock pulses are ignored and do not lead to data transfers). If an input signal SELIN is low active, it should be inverted in the DX2 input stage.

In master mode, a master slave select signal MSLS is generated by the internal slave select generator. In order to address different external slave devices independently, the internal MSLS signal is made available externally via up to 8 SEL0x output signals that can be configured by the block SELCFG (select configuration).

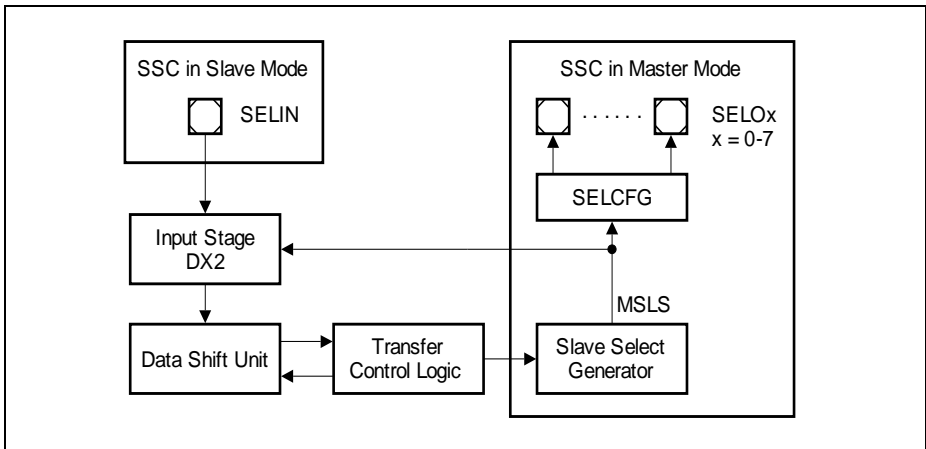


Figure 17-41 SSC Slave Select Signals

The control of the SELCFG block is based on protocol specific bits and bit fields in the protocol control register PCR. For the generation of the MSLS signal please refer to [Section 17.4.3.2](#).

- PCR.SELCTR to chose between direct and coded select mode
- PCR.SELINV to invert the SELO_x outputs
- PCR.SELO[7:0] as individual value for each SELO_x line

The SELCFG block supports the following configurations of the SELO_x output signals:

- Direct Select Mode (SELCTR = 1):
Each SELO_x line (with x = 0-7) can be directly connected to an external slave device. If bit x in bit field SELO is 0, the SELO_x output is permanently inactive. A SELO_x output becomes active while the internal signal MSLS is active (see [Section 17.4.3.2](#)) and bit x in bit field SELO is 1. Several external slave devices can be addressed in parallel if more than one bit in bit field SELO are set during a data frame. The number of external slave devices that can be addressed individually is limited to the number of available SELO_x outputs.
- Coded Select Mode (SELCTR = 0):
The SELO_x lines (with x = 1-7) can be used as addresses for an external address decoder to increase the number of external slave devices. These lines only change with the start of a new data frame and have no other relation to MSLS. Signal SELO₀ can be used as enable signal for the external address decoder. It is active while MSLS is active (during a data frame) and bit 0 in bit field SELO is 1. Furthermore, in coded select mode, this output line is delayed by one cycle of f_{PERIPH} compared to MSLS to allow the other SELO_x lines to stabilize before enabling the address decoder.

17.4.2 Operating the SSC

This chapter contains SSC issues, that are of general interest and not directly linked to either master mode or slave mode.

17.4.2.1 Automatic Shadow Mechanism

The contents of the baud rate control register BRG, bit fields SCTR.FLE as well as the protocol control register PCR are internally kept constant while a data frame is transferred (= while MSLS is active) by an automatic shadow mechanism. The registers can be programmed all the time with new settings that are taken into account for the next data frame. During a data frame, the applied (shadowed) setting is not changed, although new values have been written after the start of the data frame.

Bit fields SCTR.WLE, SCTR.DSM, SCTR.HPCDIR and SCTR.SDIR are shadowed automatically with the start of each data word. As a result, a data frame can consist of data words with a different length, or data words that are transmitted or received through different number of data lines. It is recommended to change SCTR.SDIR only when no data frame is running to avoid interference between hardware and software.

Please note that the starting point of a data word are different for a transmitter (first bit transmitted) and a receiver (first bit received). In order to ensure correct handling, it is recommended to refer to the receive start interrupt RSI before modifying SCTR.WLE. If TCSR.WLEMD = 1, it is recommended to update TCSR and TBUFxx after the receiver start interrupt has been generated.

17.4.2.2 Mode Control Behavior

In SSC mode, the following kernel modes are supported:

- Run Mode 0/1:
Behavior as programmed, no impact on data transfers.
- Stop Mode 0/1:
The content of the transmit buffer is considered as not valid for transmission. Although being considered as 0, bit TCSR.TDV it is not modified by the stop mode condition.
In master mode, a currently running word transfer is finished normally, but no new data word is started (the stop condition is not considered as end-of-frame condition). In slave mode, a currently running word transfer is finished normally. Passive data will be sent out instead of a valid data word if a data word transfer is started by the external master while the slave device is in stop mode. In order to avoid passive slave transmit data, it is recommended not to program stop mode for an SSC slave device if the master device does not respect the slave device's stop mode.

17.4.2.3 Disabling SSC Mode

In order to disable SSC mode without any data corruption, the receiver and the transmitter have to be both idle. This is ensured by requesting Stop Mode 1 in register KSCFG. After Stop Mode 1 has been acknowledged by KSCFG.2 = 1, the SSC mode can be disabled.

17.4.2.4 Data Frame Control

An SSC data frame can consist of several consecutive data words that may be separated by an inter-word delay. Without inter-word delay, the data words seem to form a longer data word, being equivalent to a data frame. The length of the data words are most commonly identical within a data frame, but may also differ from one word to another. The data word length information (defined by SCTR.WLE) is evaluated for each new data word, whereas the frame length information (defined by SCTR.FLE) is evaluated at the beginning at each start of a new frame.

The length of an SSC data frame can be defined in two different ways:

- By the number of bits per frame:
If the number of bits per data frame is defined (frame length FLE), a slave select signal is not necessarily required to indicate the start and the end of a data frame. If the programmed number of bits per frame is reached within a data word, the frame is considered as finished and remaining data bits in the last data word are ignored and are not transferred.
This method can be applied for data frames with up to 63 data bits.
- By the slave select signal:
If the number of bits per data frame is not known, the start/end information of a data frame is given by a slave select signal. If a deactivation of the slave select signal is detected within a data word, the frame is considered as finished and remaining data bits in the last data word are ignored and are not transferred.
This method has to be applied for frames with more than 63 data bits (programming limit of FLE). The advantage of slave select signals is the clearly defined start and end condition of data frames in a data stream. Furthermore, slave select signals allow to address slave devices individually.

17.4.2.5 Parity Mode

The SSC allows parity generation for transmission and parity check for reception on frame base. The type of parity can be selected by bit field CCR.PM, common for transmission and reception (no parity, even or odd parity). If the parity handling is disabled, the SSC frame does not contain any parity bit. For consistency reasons, all communication partners have to be programmed to the same parity mode.

Universal Serial Interface Channel (USIC)

If parity generation has been enabled, the transmitter automatically extends the clock by one cycle after the last data word of the data frame, and sends out its calculated parity bit in this cycle.

Figure 17-42 shows how a parity bit is added to the transmitted data bits of a frame. The number of the transmitted bits of a complete frame with parity is always one more than that without parity. The parity bit is transmitted as the last bit of a frame, following the data bits, independent of the shift direction (SCTR.SDIR).

Note: For dual and quad SSC protocols, the parity bit will be transmitted and received only on DOUT0 and DX0 respectively in the extended clock cycle.

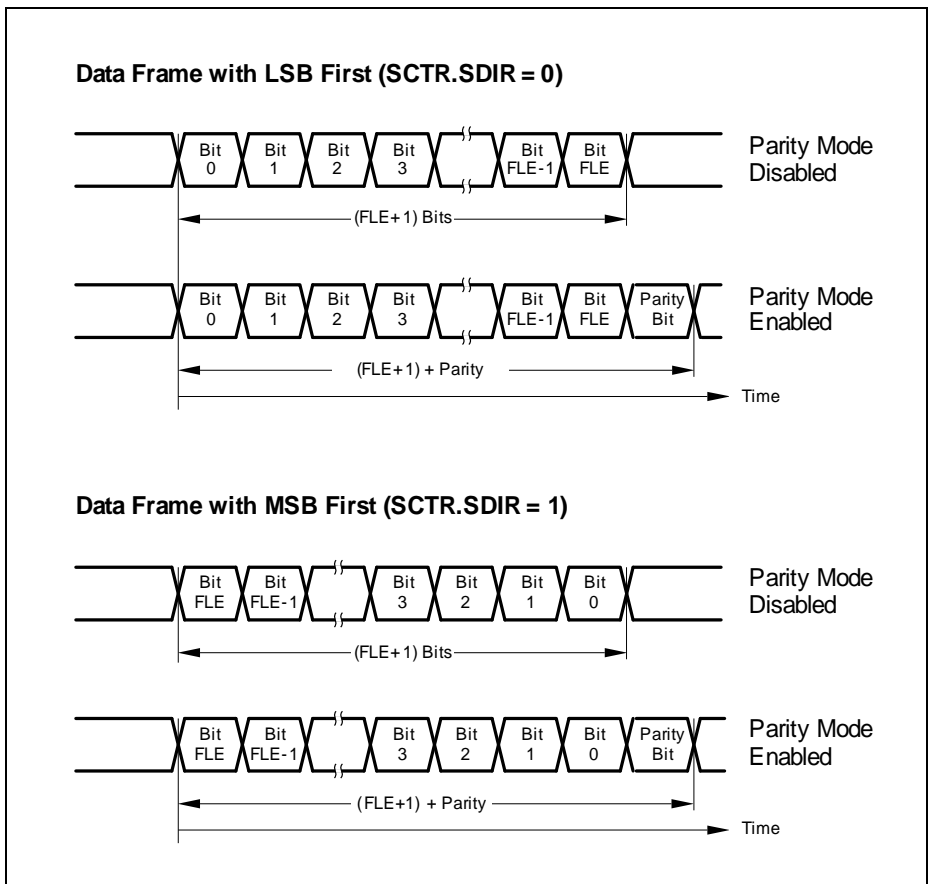


Figure 17-42 Data Frames without/with Parity

Universal Serial Interface Channel (USIC)

Similarly, after the receiver receives the last word of a data frame as defined by FLE, it expects an additional one clock cycle, which will contain the parity bit. The receiver interprets this bit as received parity and separates it from the received data. The received parity bit value is instead monitored in the protocol-related argument (PAR) of the receiver buffer status registers as receiver buffer status information. The receiver compares the bit to its internally calculated parity and the result of the parity check is indicated by the flag PSR.PARERR. The parity error event generates a protocol interrupt if PCR.PARIEN = 1.

Parity bit generation and detection is not supported for the following cases:

- When frame length is 64 data bits or greater, i.e. $FLE = 63_H$;
- When in slave mode, the end of frame occurs before the number of data bits defined by FLE is reached.
- When in slave mode, the content of the TBUF is not valid at the transmission start. In this case, the slave outputs the level defined by SCTR.PDL including for the parity bit. This might result in the detection of a parity error at the master.

17.4.2.6 Transfer Mode

In SSC mode, bit field SCTR.TRM = 01_B has to be programmed to allow data transfers. Setting SCTR.TRM = 00_B disables and stops the data transfer immediately.

17.4.2.7 Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to SSC frame handling.

- Transmit buffer interrupt TBI:
Bit PSR.TBIF is set after the start of first data bit of a data word.
- Transmit shift interrupt TSI:
Bit PSR.TSIF is set after the start of the last data bit of a data word.
- Receiver start interrupt RSI:
Bit PSR.RSIF is set after the reception of the first data bit of a data word.
With this event, bit TCSR.TDV is cleared and new data can be loaded to the transmit buffer.
- Receiver interrupt RI:
The reception of the second, third, and all subsequent words in a multi-word frame is always indicated by RBUFSR.SOF = 0. Bit PSR.RIF is set after the reception of the last data bit of a data word if RBUFSR.SOF = 0.
Bit RBUFSR.SOF indicates whether the received data word has been the first data word of a multi-word frame or some subsequent word. In SSC mode, it decides if alternative interrupt or receive interrupt is generated.
- Alternative interrupt AI:
The reception of the first word in a frame is always indicated by RBUFSR.SOF = 1. This is true both in case of reception of multi-word frames and single-word frames. In SSC mode, this results in setting PSR.AIF.

17.4.2.8 Baud Rate Generator Interrupt Handling

The baud rate generator interrupt indicate that the capture mode timer has reached its maximum value. With this event, the bit PSR.BRGIF is set.

17.4.2.9 Protocol-Related Argument and Error

The protocol-related argument (RBUFSR.PAR) and the protocol-related error (RBUFSR.PERR) are two flags that are assigned to each received data word in the corresponding receiver buffer status registers.

In SSC mode, the received parity bit is monitored by the protocol-related argument. The received start of frame indication is monitored by the protocol-related error indication (0 = received word is not the first word of a frame, 1 = received word is the first word of a new frame).

Note: For SSC, the parity error event indication bit is located in the PSR register.

17.4.2.10 Receive Buffer Handling

If a receive FIFO buffer is available (CCFG.RB = 1) and enabled for data handling (RBCTR.SIZE > 0), it is recommended to set RBCTR.RCIM = 01_B in SSC mode. This leads to an indication that the data word has been the first data word of a new data frame if bit OUTR.RCI[4] = 1, and the word length of the received data is given by OUTR.RCI[3:0].

The standard receive buffer event and the alternative receive buffer event can be used for the following operation in RCI mode (RBCTR.RNM = 1):

- A standard receive buffer event indicates that a data word can be read from OUTR that has not been the first word of a data frame.
- An alternative receive buffer event indicates that the first data word of a new data frame can be read from OUTR.

17.4.2.11 Multi-IO SSC Protocols

The SSC implements the following three features to support multiple data input/output SSC protocols, such as the dual- and quad-SSC:

1. Data Shift Mode ([Section 17.2.5.2](#))
Configures the data for transmission and reception using one, two or four data lines in parallel, through the bit field SCTR.DSM.
2. Hardware Port Control ([Section 17.2.7](#))
Sets up a dedicated hardware interface to control the direction of the pins overlaid with both DIN_x and DOUT_x functions, through the bit SCTR.HPCDIR.
3. Transmit Control Information ([Section 17.2.5.3](#))
Allows the dynamic control of both the shift mode and pin direction during data transfers by writing to SCTR.DSM and SCTR.HPCDIR with TCI.

Universal Serial Interface Channel (USIC)

Figure 17-43 shows an example of a Quad-SSC protocol, which requires the master SSC to first transmit a command byte (to request a quad output read from the slave) and a dummy byte through a single data line. At the end of the dummy byte, both master and slave SSC switches to quad data lines, and with the roles of transmitter and receiver reversed. The master SSC then receives the data four bits per shift clock from the slave through the MRST[3:0] lines.

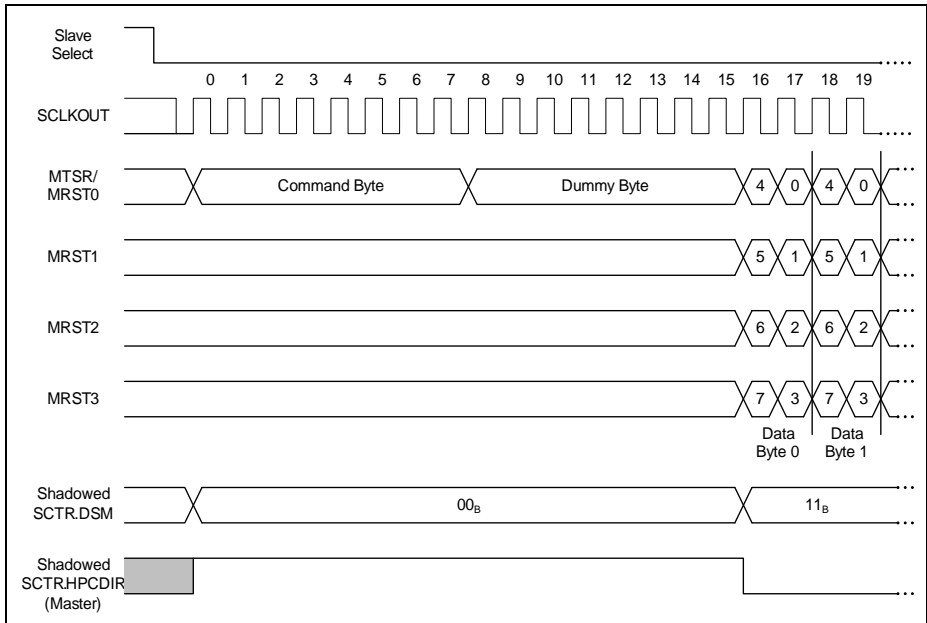


Figure 17-43 Quad-SSC Example

To work with the quad-SSC protocol in the given example, the following issues have to be additionally considered on top of those defined in [Section 17.4.3](#) and [Section 17.4.4](#):

- During the initialization phase:
 - Set CCR.HPCEN to 11_B to enable the dedicated hardware interface to the DX0/DOUT0, DX3/DOUT1, DX4/DOUT2 and DX5/DOUT3 pins.
 - Set TCSR.[4:0] to 10_H to enable hardware port control in TC1
- To start the data transfer:
 - For the master SSC, write the command and dummy bytes into TBUF04 to select a single data line in output mode and initiate the data transfer.
 - For the slave SSC, dummy data can be preloaded into TBUF00 to select a single data line in input mode.
- To switch to quad data lines and pin direction:

Universal Serial Interface Channel (USIC)

- For the master SSC, write subsequent dummy data to TBUF03 to select quad data lines in input mode to read in valid slave data.
- For the slave SSC, write valid data to TBUF07 for transmission through quad data lines in output mode.

Note: If DOUT[1:0] or DOUT[3:0] pins are already enabled with CCR.HPCEN but SCTR.DSM does not use all pins, the unused DOUTx pins output the passive data level defined by SCTR.PDL.

Figure 17-44 shows the connections for the Quad-SSC example.

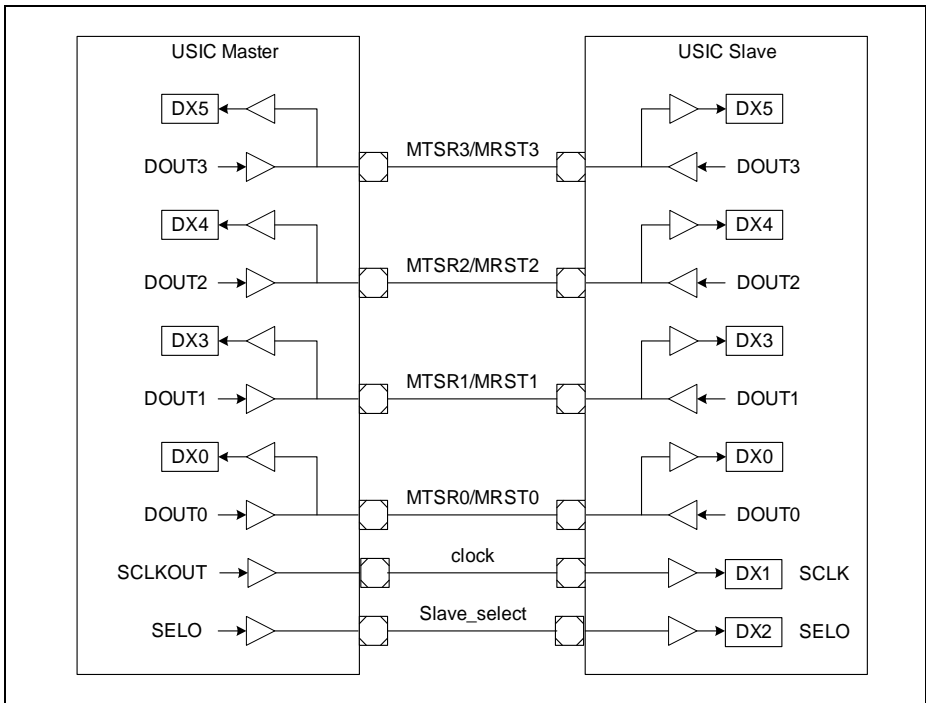


Figure 17-44 Connections for Quad-SSC Example

17.4.3 Operating the SSC in Master Mode

In order to operate the SSC in master mode, the following issues have to be considered:

- Select SSC mode:
It is recommended to configure all parameters of the SSC that do not change during run time while CCR.MODE = 0000_B. Bit field SCTR.TRM = 01_B has to be programmed. The configuration of the input stages has to be done while

Universal Serial Interface Channel (USIC)

CCR.MODE = 0000_B to avoid unintended edges of the input signals and the SSC mode can be enabled by CCR.MODE = 0001_B afterwards.

- Pin connections:
Establish a connection of the input stage (DX0, DX3, DX4, DX5) with the selected receive data input pin (DIN[3:0]) with DXnCR.INSW = 1 and configure the transmit data output pin (DOUT[3:0]). One, two or four such connections may be needed depending on the protocol. For half-duplex configurations, hardware port control can be also used to establish the required connections.
- Baud rate generation:
The desired baud rate setting has to be selected, comprising the fractional divider and the baud rate generator. Bit DX1CR.INSW = 0 has to be programmed to use the baud rate generator output SCLK directly as input for the data shift unit. Configure a shift clock output pin (signal SCLKOUT).
- Slave select generation:
The slave select delay generation has to be enabled by setting PCR.MSLSEN = 1 and the programming of the time quanta counter setting. Bit DX2CR.INSW = 0 has to be programmed to use the slave select generator output MSLS as input for the data shift unit. Configure slave select output pins (signals SELOx) if needed.
- Data format configuration:
The word length, the frame length, the shift direction and shift mode have to be set up according to the application requirements by programming the register SCTR.

Note: The USIC can only receive in master mode if it is transmitting, because the master frame handling refers to bit TDV of the transmitter part.

Note: The step to enable the alternate output port functions should only be done after the SSC mode is enabled, to avoid unintended spikes on the output.

17.4.3.1 Baud Rate Generation

The baud rate (determining the length of one data bit) of the SSC is defined by the frequency of the SCLK signal (one period of f_{SCLK} represents one data bit). The SSC baud rate generation does not imply any time quanta counter.

In a standard SSC application, the phase relation between the optional MCLK output signal and SCLK is not relevant and can be disabled (BRG.PPPEN = 0). In this case, the SCLK signal directly derives from the protocol input frequency f_{PIN} . In the exceptional case that a fixed phase relation between the MCLK signal and SCLK is required (e.g. when using MCLK as clock reference for external devices), the additional divider by 2 stage has to be taken into account (BRG.PPPEN = 1).

Universal Serial Interface Channel (USIC)

The adjustable divider factor is defined by bit field BRG.PDIV.

$$f_{SCLK} = \frac{f_{PIN}}{2} \times \frac{1}{PDIV + 1} \quad \text{if } PPPEN = 0$$

$$f_{SCLK} = \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} \quad \text{if } PPPEN = 1$$
(17.8)

17.4.3.2 MSLS Generation

The slave select signals indicate the start and the end of a data frame and are also used by the communication master to individually select the desired slave device. A slave select output of the communication master becomes active a programmable time before a data part of the frame is started (leading delay T_{ld}), necessary to prepare the slave device for the following communication. After the transfer of a data part of the frame, it becomes inactive again a programmable time after the end of the last bit (trailing delay T_{td}) to respect the slave hold time requirements. If data frames are transferred back-to-back one after the other, the minimum time between the deactivation of the slave select and the next activation of a slave select is programmable (next-frame delay T_{nf}). If a data frame consists of more than one data word, an optional delay between the data words can also be programmed (inter-word delay T_{iw}).

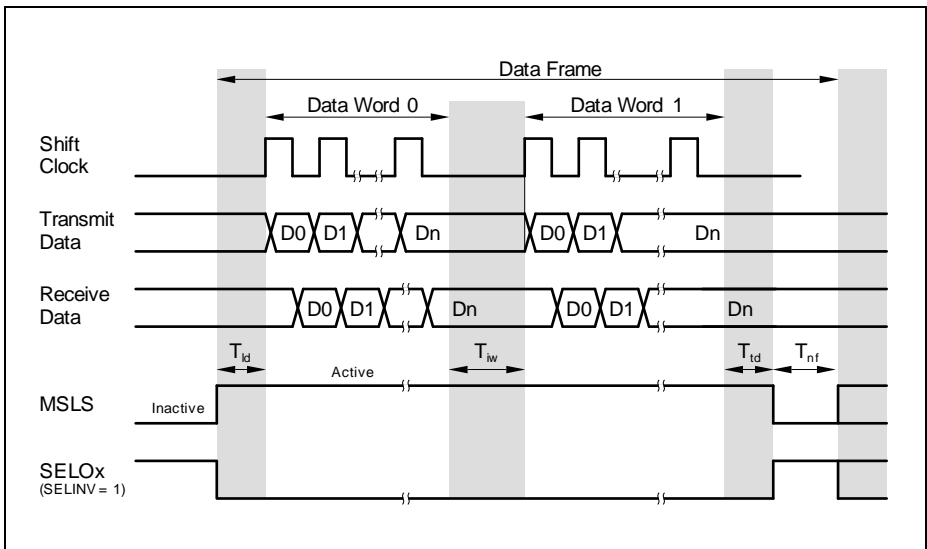


Figure 17-45 MSLS Generation in SSC Master Mode

In SSC master mode, the slave select delays are defined as follows:

Universal Serial Interface Channel (USIC)

- **Leading delay T_{ld} :**
The leading delay starts if valid data is available for transmission. The internal signal MSLS becomes active with the start of the leading delay. The first shift clock edge (rising edge) of SCLK is generated by the baud rate generator after the leading delay has elapsed.
- **Trailing delay T_{td}**
The trailing delay starts at the end of the last SCLK cycle of a data frame. The internal signal MSLS becomes inactive with the end of the trailing delay.
- **Inter-word delay T_{iw} :**
This delay is optional and can be enabled/disabled by PCR.TIWEN. If the inter-word delay is disabled (TIWEN = 0), the last data bit of a data word is directly followed by the first data bit of the next data word of the same data frame. If enabled (TIWEN = 1), the inter-word delay starts at the end of the last SCLK cycle of a data word. The first SCLK cycle of the following data word of the same data frame is started when the inter-word delay has elapsed. During this time, no shift clock pulses are generated and signal MSLS stays active. The communication partner has time to “digest” the previous data word or to prepare for the next one.
- **Next-frame delay T_{nf} :**
The next-frame delay starts at the end of the trailing delay. During this time, no shift clock pulses are generated and signal MSLS stays inactive. A frame is considered as finished after the next-frame delay has elapsed.

17.4.3.3 Automatic Slave Select Update

If the number of bits per SSC frame and the word length are defined by bit fields SCTR.FLE and SCTR.WLE, the transmit control information TCI can be used to update the slave select setting PCR.CTR[23:16] to control the SELOx select outputs. The automatic update mechanism is enabled by TCSR.SELMD = 1 (bits TCSR.WLEMD, FLEMD, and WAMD have to be cleared). In this case, the TCI of the first data word of a frame defines the slave select setting of the complete frame due to the automatic shadow mechanism (see [Page 17-62](#)).

17.4.3.4 Slave Select Delay Generation

The slave select delay generation is based on time quanta. The length of a time quantum (defined by the period of the f_{CTQIN}) and the number of time quanta per delay can be programmed.

In standard SSC applications, the leading delay T_{ld} and the trailing delay T_{td} are mainly used to ensure stability on the input and output lines as well as to respect setup and hold times of the input stages. These two delays have the same length (in most cases shorter than a bit time) and can be programmed with the same set of bit fields.

- BRG.CTQSEL
to define the input frequency f_{CTQIN} for the time quanta generation for T_{ld} and T_{td}
- BRG.PCTQ
to define the length of a time quantum (division of f_{CTQIN} by 1, 2, 3, or 4) for T_{ld} and T_{td}
- BRG.DCTQ
to define the number of time quanta for the delay generation for T_{ld} and T_{td}

The inter-word delay T_{iw} and the next-frame delay T_{nf} are used to handle received data or to prepare data for the next word or frame. These two delays have the same length (in most cases in the bit time range) and can be programmed with a second, independent set of bit fields.

- PCR.CTQSEL1
to define the input frequency f_{CTQIN} for the time quanta generation for T_{nf} and T_{iw}
- PCR.PCTQ1
to define the length of a time quantum (division of f_{CTQIN} by 1, 2, 3, or 4) for T_{nf} and T_{iw}
- PCR.DCTQ1
to define the number of time quanta for the delay generation for T_{nf} and T_{iw}
- PCR.TIWEN
to enable/disable the inter-word delay T_{iw}

Each delay depends on the length of a time quantum and the programmed number of time quanta given by the bit fields CTQSEL/CTQSEL1, PCTQ/DCTQ and PCTQ1/DCTQ1 (the coding of CTQSEL1 is similar to CTQSEL, etc.). To provide a high flexibility in programming the delay length, the input frequencies can be selected between several possibilities (e.g. based on bit times or on the faster inputs of the protocol-related divider). The delay times are defined as follows:

$$T_{ld} = T_{td} = \frac{(PCTQ + 1) \times (DCTQ + 1)}{f_{CTQIN}} \quad (17.9)$$

$$T_{iw} = T_{nf} = \frac{(PCTQ1 + 1) \times (DCTQ1 + 1)}{f_{CTQIN}}$$

17.4.3.5 Protocol Interrupt Events

The following protocol-related events generated in SSC mode and can lead to a protocol interrupt. They are related to the start and the end of a data frame. After the start of a data frame a new setting could be programmed for the next data frame and after the end of a data frame the SSC connections to pins can be changed.

Please note that the bits in register PSR are not all automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- **MSLS Interrupt:**
This interrupt indicates in master mode (MSLS generation enabled) that a data frame has started (activation of MSLS) and has been finished (deactivation of MSLS). Any change of the internal MSLS signal sets bit PSR.MSLSEV and additionally, a protocol interrupt can be generated if PCR.MSLSIEN = 1. The actual state of the internal MSLS signal can be read out at PSR.MSLS to take appropriate actions when this interrupt has been detected.
- **DX2T Interrupt:**
This interrupt monitors edges of the input signal of the DX2 stage (although this signal is not used as slave select input for data transfers). A programmable edge detection for the DX2 input signal sets bit PSR.DX2TEV and additionally, a protocol interrupt can be generated if PCR.DX2TIEN = 1. The actual state of the selected input signal can be read out at PSR.DX2S to take appropriate actions when this interrupt has been detected.
- **Parity Error Interrupt:**
This interrupt indicates that there is a mismatch in the received parity bit (in RBUFSR.PAR) with the calculated parity bit of the last received word of a data frame.

17.4.3.6 End-of-Frame Control

The information about the frame length is required for the MSLS generator of the master device. In addition to the mechanism based on the number of bits per frame (selected with $SCTR.FLE < 63$), the following alternative mechanisms for end of frame handling are supported. It is recommended to set $SCTR.FLE = 63$ (if several end of frame mechanisms are activated in parallel, the first end condition being found finishes the frame).

- **Software-based start of frame indication TCSR.SOF:**
This mechanism can be used if software handles the TBUF data without data FIFO. If bit SOF is set, a valid content of TBUF is considered as first word of a new frame. Bit SOF has to be set before the content of TBUF is transferred to the transmit shift register, so it is recommended to write it before writing data to TBUF. A current data word transfer is finished completely and the slave select delays T_{td} and T_{nf} are applied before starting a new data frame with T_{td} and the content of TBUF. For software-handling of bit SOF, bit $TCSR.WLEMD = 0$ has to be programmed. In this case, all $TBUF[31:0]$ address locations show an identical behavior (TCI not taken into account for data handling).
- **Software-based end of frame indication TCSR.EOF:**
This mechanism can be used if software handles the TBUF data without data FIFO. If bit EOF is set, a valid content of TBUF is considered as last word of a new frame. Bit EOF has to be set before the content of TBUF is transferred to the transmit shift register, so it is recommended to write it before writing data to TBUF. The data word in TBUF is sent out completely and the slave select delays T_{td} and T_{nf} are applied. A new data frame can start with T_{td} with the next valid TBUF value. For software-handling of bit EOF, bit $TCSR.WLEMD = 0$ has to be programmed. In this case, all $TBUF[31:0]$ address locations show an identical behavior (TCI not taken into account for data handling).
- **Software-based address related end of frame handling:**
This mechanism can be used if software handles the TBUF data without data FIFO. If bit $TCSR.WLEMD = 1$, the address of the written $TBUF[31:0]$ is used as transmit control information $TCI[4:0]$ to update $SCTR.WLE (= TCI[3:0])$ and $TCSR.EOF (= TCI[4])$ for each data word. The written $TBUF[31:0]$ address location defines the word length and the end of a frame (locations $TBUF[31:16]$ lead to a frame end). For example, writing transmit data to $TBUF[07]$ results in a data word of 8-bit length without finishing the frame, whereas writing transmit data to $TBUF[31]$ leads to a data word length of 16 bits, followed by T_{td} , the deactivation of MSLS and T_{nf} . If $TCSR.WLEMD = 1$, bits $TCSR.EOF$ and SOF , as well as $SCTR.WLE$ must not be written by software after writing data to a TBUF location. Furthermore, it is recommended to clear bits $TCSR.SELMD$, $FLEMD$ and $WAMD$.
- **FIFO-based address related end of frame handling:**
This mechanism can be used if a data FIFO is used to store the transmit data. The general behavior is similar to the software-based address related end of frame

Universal Serial Interface Channel (USIC)

handling, except that transmit data is not written to the locations TBUF[31:0], but to the FIFO input locations IN[31:0] instead. In this case, software must not write to any of the TBUF locations.

- **TBUF related end of frame handling:**
If bit PCR.FEM = 0, an end of frame is assumed if the transmit buffer TBUF does not contain valid transmit data at the end of a data word transmission (TCSR.TDV = 0 or in Stop Mode). In this case, the software has to take care that TBUF does not run empty during a data frame in Run Mode. If bit PCR.FEM = 1, signal MSLS stays active while the transmit buffer is waiting for new data (TCSR.TDV = 1 again) or until Stop Mode is left.
- **Explicit end of frame by software:**
The software can explicitly stop a frame by clearing bit PSR.MSLS by writing a 1 to the related bit position in register PSCR. This write action immediately clears bit PSR.MSLS, whereas the internal MSLS signal becomes inactive after finishing a currently running word transfer and respecting the slave select delays T_{id} and T_{nf} .

17.4.4 Operating the SSC in Slave Mode

In order to operate the SSC in slave mode, the following issues have to be considered:

- **Select SSC mode:**
It is recommended to configure all parameters of the SSC that do not change during run time while $CCR.MODE = 0000_B$. Bit field $SCTR.TRM = 01_B$ has to be programmed. The configuration of the input stages has to be done while $CCR.MODE = 0000_B$ to avoid unintended edges of the input signals and the SSC mode can be enabled afterwards by $CCR.MODE = 0001_B$.
- **Pin connections:**
Establish the connection of the input stage (DX0, DX3, DX4, DX5) with the selected receive data input pin (DIN[3:0]) with $DXnCR.INSW = 1$ and configure the transmit data output pin (DOUT[3:0]). One, two or four such connections may be needed depending on the protocol. For half-duplex configurations, hardware port control can be also used to establish the required connections.
Establish a connection of input stage DX1 with the selected shift clock input pin (signal SCLKIN) with $DX1CR.INSW = 1$.
Establish a connection of input stage DX2 with the selected slave select input pin (signal SELIN) with $DX2CR.INSW = 1$. If no slave select input signal is used, the DX2 stage has to deliver a 1-level to the data shift unit to allow data reception and transmission. If a slave device is not selected (DX2 stage delivers a 0 to the data shift unit) and a shift clock pulse are received, the incoming data is not received and the DOUTx signal outputs the passive data level defined by $SCTR.PDL$.
Note that the step to enable the alternate output port functions should only be done after the SSC mode is enabled, to avoid unintended spikes on the output.
- **Baud rate generation:**
The baud rate generator is not needed and can be switched off by the fractional divider.
- **Data format configuration:**
If required, the shift mode can be set up for reception and/or transmission of two or four data bits at one time by programming the register SCTR.
- **Slave select generation:**
The slave select delay generation is not needed and can be switched off. The bits and bit fields MSLSEN, SELCTR, SELINV, CTQSEL1, PCTQ1, DCTQ1, MSLSIEN, SELO[7:0], and TIWEN in register PCR are not necessary and can be programmed to 0.

17.4.4.1 Protocol Interrupts

The following protocol-related events generated in SSC mode and can lead to a protocol interrupt. They are related to the start and the end of a data frame. After the start of a data frame a new setting could be programmed for the next data frame and after the end of a data frame the SSC connections to pins can be changed.

Universal Serial Interface Channel (USIC)

Please note that the bits in register PSR are not all automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- **MSLS event:**
The MSLS generation being switched off, this event is not available.
- **DX2T event:**
The slave select input signal SELIN is handled by the DX2 stage and the edges of the selected signal can generate a protocol interrupt. This interrupt allows to indicate that a data frame has started and/or that a data frame has been completely finished. A programmable edge detection for the DX2 input signal activates DX2T, sets bit PSR.DX2TEV and additionally, a protocol interrupt can be generated if PCR.DX2TIEN = 1. The actual state of the selected input signal can be read out at PSR.DX2S to take appropriate actions when this interrupt has been detected.
- **Parity Error Interrupt:**
This interrupt indicates that there is a mismatch in the received parity bit (in RBUF.SR.PAR) with the calculated parity bit of the last received word of a data frame.

17.4.4.2 End-of-Frame Control

In slave mode, the following possibilities exist to determine the frame length. The slave device either has to refer to an external slave select signal, or to the number of received data bits.

- **Frame length known in advance by the slave device, no slave select:**
In this case bit field SCTR.FLE can be programmed to the known value (if it does not exceed 63 bits). A currently running data word transfer is considered as finished if the programmed frame length is reached.
- **Frame length not known by the slave, no slave select:**
In this case, the slave device's software has to decide on data word base if a frame is finished. Bit field SCTR.FLE can be either programmed to the word length SCTR.WLE, or to its maximum value to disable the slave internal frame length evaluation by counting received bits.
- **Slave device addressed via slave select signal SELIN:**
If the slave device is addressed by a slave select signal delivered by the communication master, the frame start and end information are given by this signal. In this case, bit field SCTR.FLE should be programmed to its maximum value to disable the slave internal frame length evaluation.

Universal Serial Interface Channel (USIC)

17.4.5 SSC Protocol Registers

In SSC mode, the registers PCR and PSR handle SSC related information.

17.4.5.1 SSC Protocol Control Registers

In SSC mode, the PCR register bits or bit fields are defined as described in this section.

PCR

Protocol Control Register [SSC Mode]

(3C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCLK	0				SLPHSEL	TIWEN	SELO								
rw	rw				rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DX2TIEN	MSLSIEN	PARIEN	DCTQ1				PCTQ1		CTQSEL1	FEM	SELINV	SELCTR	MSLSEN		
rw	rw	rw	rw				rw		rw	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
MSLSEN	0	rw	<p>MSLS Enable</p> <p>This bit enables/disables the generation of the master slave select signal MSLS. If the SSC is a transfer slave, the SLS information is read from a pin and the internal generation is not needed. If the SSC is a transfer master, it has to provide the MSLS signal.</p> <p>0_B The MSLS generation is disabled (MSLS = 0). This is the setting for SSC slave mode.</p> <p>1_B The MSLS generation is enabled. This is the setting for SSC master mode.</p>
SELCTR	1	rw	<p>Select Control</p> <p>This bit selects the operating mode for the SELO[7:0] outputs.</p> <p>0_B The coded select mode is enabled.</p> <p>1_B The direct select mode is enabled.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
SELINV	2	rw	<p>Select Inversion</p> <p>This bit defines if the polarity of the SELO[7:0] outputs in relation to the master slave select signal MSLS.</p> <p>0_B The SELO outputs have the same polarity as the MSLS signal (active high).</p> <p>1_B The SELO outputs have the inverted polarity to the MSLS signal (active low).</p>
FEM	3	rw	<p>Frame End Mode</p> <p>This bit defines if a transmit buffer content that is not valid for transmission is considered as an end of frame condition for the slave select generation.</p> <p>0_B The current data frame is considered as finished when the last bit of a data word has been sent out and the transmit buffer TBUF does not contain new data (TDV = 0).</p> <p>1_B The MSLS signal is kept active also while no new data is available and no other end of frame condition is reached. In this case, the software can accept delays in delivering the data without automatic deactivation of MSLS in multi-word data frames.</p>
CTQSEL1	[5:4]	rw	<p>Input Frequency Selection</p> <p>This bit field defines the input frequency f_{CTQIN} for the generation of the slave select delays T_{iw} and T_{nf}.</p> <p>00_B $f_{CTQIN} = f_{PDIV}$</p> <p>01_B $f_{CTQIN} = f_{PPP}$</p> <p>10_B $f_{CTQIN} = f_{SCLK}$</p> <p>11_B $f_{CTQIN} = f_{MCLK}$</p>
PCTQ1	[7:6]	rw	<p>Divider Factor PCTQ1 for T_{iw} and T_{nf}</p> <p>This bit field represents the divider factor PCTQ1 (range = 0 - 3) for the generation of the inter-word delay and the next-frame delay.</p> $T_{iw} = T_{nf} = 1/f_{CTQIN} \times (PCTQ1 + 1) \times (DCTQ1 + 1)$
DCTQ1	[12:8]	rw	<p>Divider Factor DCTQ1 for T_{iw} and T_{nf}</p> <p>This bit field represents the divider factor DCTQ1 (range = 0 - 31) for the generation of the inter-word delay and the next-frame delay.</p> $T_{iw} = T_{nf} = 1/f_{CTQIN} \times (PCTQ1 + 1) \times (DCTQ1 + 1)$

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
PARIEN	13	rw	<p>Parity Error Interrupt Enable</p> <p>This bit enables/disables the generation of a protocol interrupt with the detection of a parity error.</p> <p>0_B A protocol interrupt is not generated with the detection of a parity error.</p> <p>1_B A protocol interrupt is generated with the detection of a parity error.</p>
MSLSIEN	14	rw	<p>MSLS Interrupt Enable</p> <p>This bit enables/disables the generation of a protocol interrupt if the state of the MSLS signal changes (indicated by PSR.MSLSEV = 1).</p> <p>0_B A protocol interrupt is not generated if a change of signal MSLS is detected.</p> <p>1_B A protocol interrupt is generated if a change of signal MSLS is detected.</p>
DX2TIEN	15	rw	<p>DX2T Interrupt Enable</p> <p>This bit enables/disables the generation of a protocol interrupt if the DX2T signal becomes activated (indicated by PSR.DX2TEV = 1).</p> <p>0_B A protocol interrupt is not generated if DX2T is activated.</p> <p>1_B A protocol interrupt is generated if DX2T is activated.</p>
SELO	[23:16]	rw	<p>Select Output</p> <p>This bit field defines the setting of the SELO[7:0] output lines.</p> <p>0_B The corresponding SELO_x line cannot be activated.</p> <p>1_B The corresponding SELO_x line can be activated (according to the mode selected by SELCTR).</p>
TIWEN	24	rw	<p>Enable Inter-Word Delay T_{iw}</p> <p>This bit enables/disables the inter-word delay T_{iw} after the transmission of a data word.</p> <p>0_B No delay between data words of the same frame.</p> <p>1_B The inter-word delay T_{iw} is enabled and introduced between data words of the same frame.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
SLPHSEL	25	rw	<p>Slave Mode Clock Phase Select</p> <p>This bit selects the clock phase for the data shifting in slave mode.</p> <p>0_B Data bits are shifted out with the leading edge of the shift clock signal and latched in with the trailing edge.</p> <p>1_B The first data bit is shifted out when the data shift unit receives a low to high transition from the DX2 stage. Subsequent bits are shifted out with the trailing edge of the shift clock signal. Data bits are always latched in with the leading edge.</p>
MCLK	31	rw	<p>Master Clock Enable</p> <p>This bit enables/disables the generation of the master clock output signal MCLK, independent from master or slave mode.</p> <p>0_B The MCLK generation is disabled and output MCLK = 0.</p> <p>1_B The MCLK generation is enabled.</p>
0	[30:26]	rw	<p>Reserved</p> <p>Returns 0 if read; should be written with 0.</p>

Universal Serial Interface Channel (USIC)

17.4.5.2 SSC Protocol Status Register

In SSC mode, the PSR register bits or bit fields are defined as described in this section. The bits and bit fields in register PSR are not cleared by hardware.

The flags in the PSR register can be cleared by writing a 1 to the corresponding bit position in register PSCR. Writing a 1 to a bit position in PSR sets the corresponding flag, but does not lead to further actions (no interrupt generation). Writing a 0 has no effect. The PSR flags should be cleared by software before enabling a new protocol.

PSR

Protocol Status Register [SSC Mode] (48_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															BRG IF
r															rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	RIF	TBIF	TSIF	DLIF	RSIF	0					PAR ERR	DX2 TEV	MSL SEV	DX2 S	MSL S
rwh	rwh	rwh	rwh	rwh	rwh	r					rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
MSLS	0	rwh	MSLS Status This bit indicates the current status of the MSLS signal. It must be cleared by software to stop a running frame. 0 _B The internal signal MSLS is inactive (0). 1 _B The internal signal MSLS is active (1).
DX2S	1	rwh	DX2S Status This bit indicates the current status of the DX2S signal that can be used as slave select input SELIN. 0 _B DX2S is 0. 1 _B DX2S is 1.
MSLSEV	2	rwh	MSLS Event Detected¹⁾ This bit indicates that the MSLS signal has changed its state since MSLSEV has been cleared. Together with the MSLS status bit, the activation/deactivation of the MSLS signal can be monitored. 0 _B The MSLS signal has not changed its state. 1 _B The MSLS signal has changed its state.

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
DX2TEV	3	rwh	DX2T Event Detected¹⁾ This bit indicates that the DX2T trigger signal has been activated since DX2TEV has been cleared. 0 _B The DX2T signal has not been activated. 1 _B The DX2T signal has been activated.
PARERR	4	rwh	Parity Error Event Detected¹⁾ This bit indicates that there is a mismatch in the received parity bit (in RBUF _{SR} .PAR) with the calculated parity bit of the last received word of the data frame. 0 _B A parity error event has not been activated. 1 _B A parity error event has been activated.
RSIF	10	rwh	Receiver Start Indication Flag 0 _B A receiver start event has not occurred. 1 _B A receiver start event has occurred.
DLIF	11	rwh	Data Lost Indication Flag 0 _B A data lost event has not occurred. 1 _B A data lost event has occurred.
TSIF	12	rwh	Transmit Shift Indication Flag 0 _B A transmit shift event has not occurred. 1 _B A transmit shift event has occurred.
TBIF	13	rwh	Transmit Buffer Indication Flag 0 _B A transmit buffer event has not occurred. 1 _B A transmit buffer event has occurred.
RIF	14	rwh	Receive Indication Flag 0 _B A receive event has not occurred. 1 _B A receive event has occurred.
AIF	15	rwh	Alternative Receive Indication Flag 0 _B An alternative receive event has not occurred. 1 _B An alternative receive event has occurred.
BRGIF	16	rwh	Baud Rate Generator Indication Flag 0 _B A baud rate generator event has not occurred. 1 _B A baud rate generator event has occurred.
0	[9:5], [31:17]	r	Reserved Returns 0 if read; not modified in SSC mode.

1) This status bit can generate a protocol interrupt in SSC mode (see [Page 17-22](#)). The general interrupt status flags are described in the general interrupt chapter.

17.4.6 SSC Timing Considerations

The input and output signals have to respect certain timings in order to ensure correct data reception and transmission. In addition to module internal timings (due to input filters, reaction times on events, etc.), also the timings from the input pin via the input stage (T_{in}) to the module and from the module via the output driver stage to the pin (T_{out}), as well as the signal propagation on the wires (T_{prop}) have to be taken into account.

Please note that there might be additional delays in the DXn input stages, because the digital filter and the synchronization stages lead to systematic delays, that have to be considered if these functions are used.

17.4.6.1 Closed-loop Delay

A system-inherent limiting factor for the baud rate of an SSC connection is the closed-loop delay. In a typical application setup, a communication master device is connected to a slave device in full-duplex mode with independent lines for transmit and receive data. In a general case, all transmitters refer to one shift clock edge for transmission and all receivers refer to the other shift clock edge for reception. The master device's SSC module sends out the transmit data, the shift clock and optionally the slave select signal. Therefore, the baud rate generation (BRG) and slave select generation (SSG) are part of the master device. The frame control is similar for SSC modules in master and slave mode, the main difference is the fact which module generates the shift clock and optionally, the slave select signals.

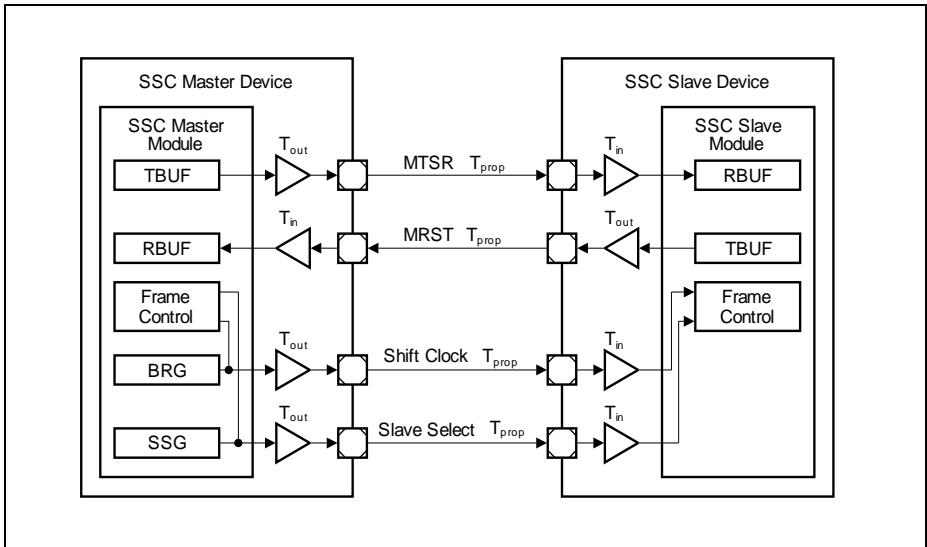


Figure 17-46 SSC Closed-loop Delay

Universal Serial Interface Channel (USIC)

The signal path between the SSC modules of the master and the slave device includes the master's output driver, the wiring to the slave device and the slave device's input stage. With the received shift clock edges, the slave device receives the master's transmit data and transmits its own data back to the master device, passing by a similar signal path in the other direction. The master module receives the slave's transmit data related to its internal shift clock edges. In order to ensure correct data reception in the master device, the slave's transmit data has to be stable (respecting setup and hold times) as master receive data with the next shift clock edge of the master (generally 1/2 shift clock period). To avoid data corruption, the accumulated delays of the input and output stages, the signal propagation on the wiring and the reaction times of the transmitter/receiver have to be carefully considered, especially at high baud rates.

In the given example, the time between the generation of the shift clock signal and the evaluation of the receive data by the master SSC module is given by the sum of $T_{\text{out_master}} + 2 \times T_{\text{prop}} + T_{\text{in_slave}} + T_{\text{out_slave}} + T_{\text{in_master}}$ + module reaction times + input setup times.

The input path is characterized by an input delay depending mainly on the input stage characteristics of the pads. The output path delay is determined by the output driver delay and its slew rate, the external load and current capability of the driver. The device specific values for the input/output driver are given in the Data Sheet.

Universal Serial Interface Channel (USIC)

Figure 17-47 describes graphically the closed-loop delay and the effect of two delay compensation options discussed in [Section 17.4.6.2](#) and [Section 17.4.6.3](#).

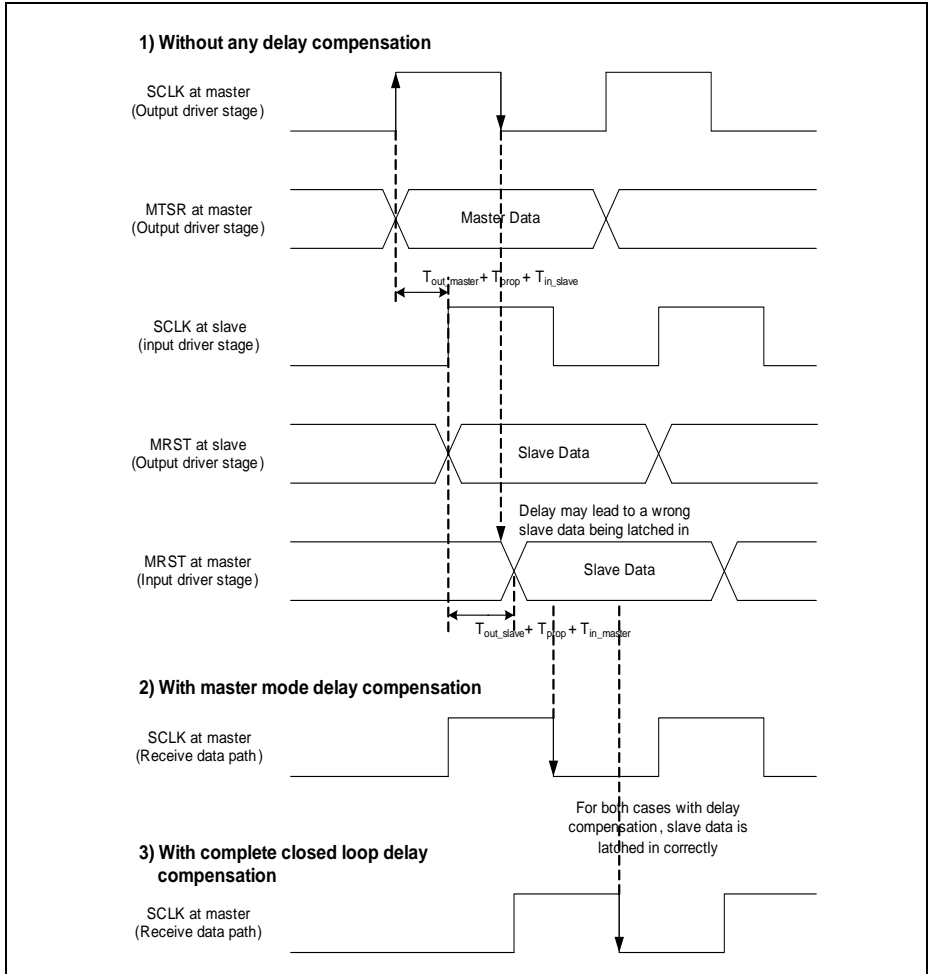


Figure 17-47 SSC Closed-loop Delay Timing Waveform

Universal Serial Interface Channel (USIC)

17.4.6.2 Delay Compensation in Master Mode

A higher baud rate can be reached by delay compensation in master mode. This compensation is possible if (at least) the shift clock pin is bidirectional.

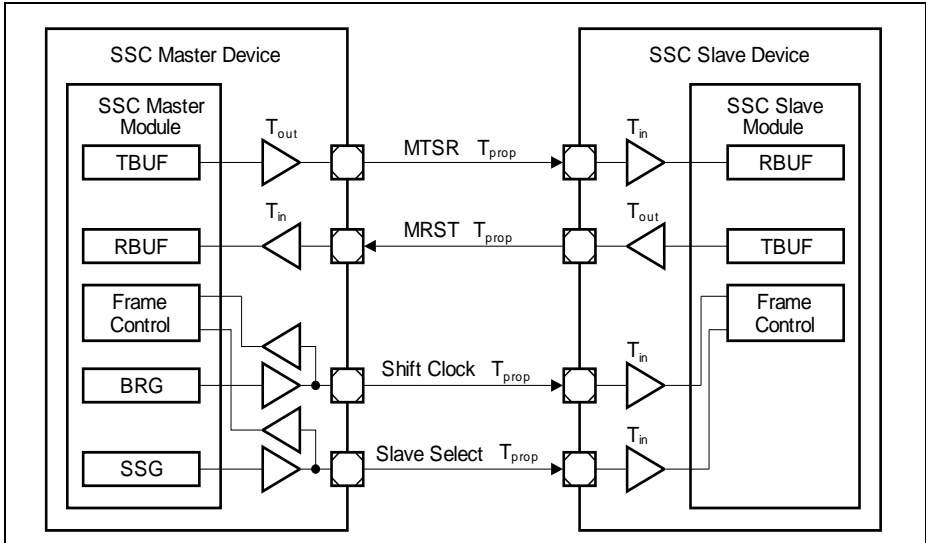


Figure 17-48 SSC Master Mode with Delay Compensation

If the receive shift clock signal in master mode is directly taken from the input function in parallel to the output signal, the output delay of the master device's shift clock output is compensated and only the difference between the input delays of the master and the slave devices have to be taken into account instead of the complete master's output delay and the slave's input delay of the shift clock path. The delay compensation is enabled with $DX1CR.DCEN = 1$ while $DX1CR.INSW = 0$ (transmit shift clock is taken from the baud-rate generator).

In the given example, the time between the evaluation of the shift clock signal and the receive data by the master SSC module is reduced by $T_{in_master} + T_{out_master}$.

Although being a master mode, the shift clock input and optionally the slave select signal are not directly connected internally to the data shift unit, but are taken as external signals from input pins. The delay compensation does not lead to additional pins for the SSC communication if the shift clock output pin (slave select output pin, respectively) is/are bidirectional. In this case, the input signal is decoupled from other internal signals, because it is related to the signal level at the pin itself.

Universal Serial Interface Channel (USIC)

17.4.6.3 Complete Closed-loop Delay Compensation

Alternatively, the complete closed-loop delay can be compensated by using one additional pin on both the SSC master and slave devices for the SSC communication.

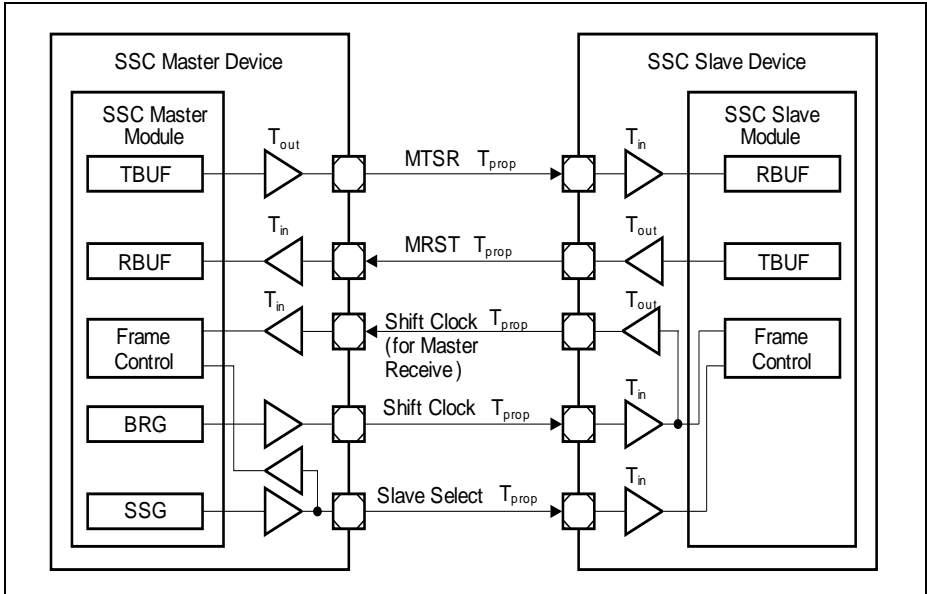


Figure 17-49 SSC Complete Closed-loop Delay Compensation

The principle behind this delay compensation method is to have the slave feedback the shift clock back to the master, which uses it as the receive shift clock. By going through a complete closed-loop signal path, the receive shift clock is thus fully compensated.

The slave has to setup the SCLKOUT pin function to output the shift clock by setting the bit BRG.SCLKOSEL to 1, while the master has to setup the DX1 pin function to receive the shift clock from the slave and enable the delay compensation with DX1CR.DCEN = 1 and DX1CR.INSW = 0.

17.5 Inter-IC Bus Protocol (IIC)

The IIC protocol of the USIC refers to the IIC bus specification [20]. Contrary to that specification, the USIC device assumes rise/fall times of the bus signals of max. 300 ns in all modes. Please refer to the pad characteristics in the AC/DC chapter for the driver capability. CBUS mode and HS mode are not supported.

The IIC mode is selected by $CCR.MODE = 0100_B$ with $CCFG.IIC = 1$ (IIC mode available).

17.5.1 Introduction

USIC IIC Features:

- Two-wire interface, with one line for shift clock transfer and synchronization (shift clock SCL), the other one for the data transfer (shift data SDA)
- Communication in standard mode (100 kBit/s) or in fast mode (up to 400 kBit/s)
- Support of 7-bit addressing, as well as 10-bit addressing
- Master mode operation, where the IIC controls the bus transactions and provides the clock signal.
- Slave mode operation, where an external master controls the bus transactions and provides the clock signal.
- Multi-master mode operation, where several masters can be connected to the bus and bus arbitration can take place, i.e. the IIC module can be master or slave. The master/slave operation of an IIC bus participant can change from frame to frame.
- Efficient frame handling (low software effort), also allowing DMA transfers
- Powerful interrupt handling due to multitude of indication flags
- Compensation support for input delays

17.5.1.1 Signal Description

An IIC connection is characterized by two wires (SDA and SCL). The output drivers for these signals must have open-drain characteristics to allow the wired-AND connection of all SDA lines together and all SCL lines together to form the IIC bus system. Due to this structure, a high level driven by an output stage does not necessarily lead immediately to a high level at the corresponding input. Therefore, each SDA or SCL connection has to be input and output at the same time, because the input function always monitors the level of the signal, also while sending.

- Shift data SDA: input handled by DX0 stage, output signal DOUT0
- Shift clock SCL: input handled by DX1 stage, output signal SCLKOUT

Figure 17-29 shows a connection of two IIC bus participants (modules IIC A and IIC B) using the USIC. In this example, the pin assignment of module IIC A shows separate pins for the input and output signals for SDA and SCL. This assignment can be used if the application does not provide pins having DOUT0 and a DX0 stage input for the same pin

Universal Serial Interface Channel (USIC)

(similar for SCLKOUT and DX1). The pin assignment of module IIC B shows the connection of DOUT0 and a DX0 input at the same pin, also for SCLKOUT and a DX1 input.

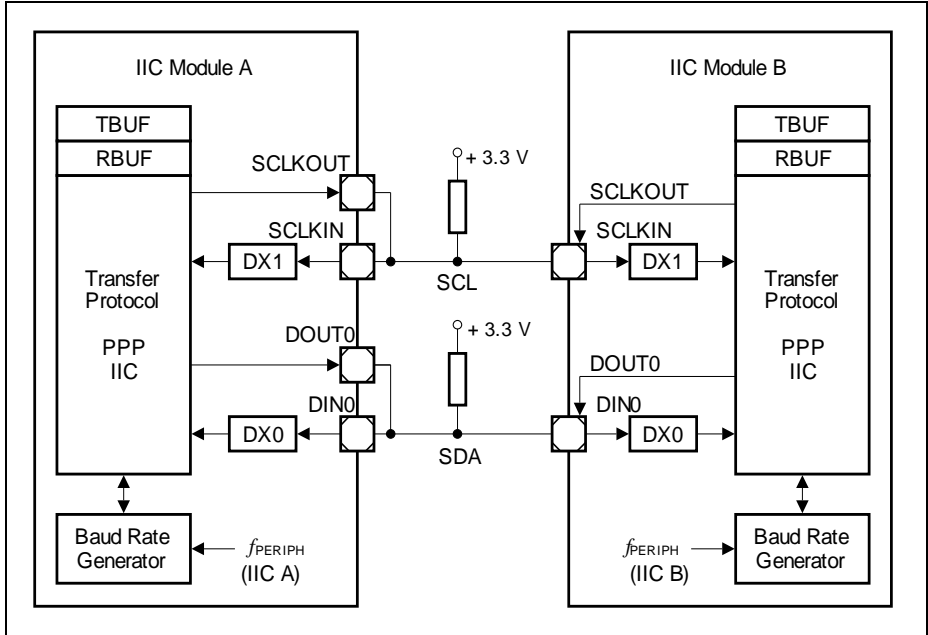


Figure 17-50 IIC Signal Connections

17.5.1.2 Symbols

A symbol is a sequence of edges on the lines SDA and SCL. Symbols contain 10 or 25 time quanta t_q , depending on the selected baud rate. The baud rate generator determines the length of the time quanta t_q , the sequence of edges in a symbol is handled by the IIC protocol pre-processor, and the sequence of symbols can be programmed by the user according to the application needs.

The following symbols are defined:

- Bus idle:
SDA and SCL are high. No data transfer takes place currently.
- Data bit symbol:
SDA stable during the high phase of SCL. SDA then represents the transferred bit value. There is one clock pulse on SCL for each transferred bit of data. During data transfers SDA may only change while SCL is low.

Universal Serial Interface Channel (USIC)

- **Start symbol:**
Signal SDA being high followed by a falling edge of SDA while SCL is high indicates a start condition. This start condition initiates a data transfer over the IIC bus after the bus has been idle.
- **Repeated start symbol:**
This start condition initiates a data transfer over the bus after a data symbol when the bus has not been idle. Therefore, SDA is set high and SCL low, followed by a start symbol.
- **Stop symbol:**
A rising edge on SDA while SCL is high indicates a stop condition. This stop condition terminates a data transfer to release the bus to idle state. Between a start condition and a stop condition an arbitrary number of bytes may be transferred.

17.5.1.3 Frame Format

Data is transferred by the 2-line IIC bus (SDA, SCL) using a protocol that ensures reliable and efficient transfers. The sender of a (data) byte receives and checks the value of the following acknowledge field. The IIC being a wired-AND bus system, a 0 of at least one device leads to a 0 on the bus, which is received by all devices.

A data word consists of 8 data bit symbols for the data value, followed by another data bit symbol for the acknowledge bit. The data word can be interpreted as address information (after a start symbol) or as transferred data (after the address).

In order to be able to receive an acknowledge signal, the sender of the data bits has to release the SDA line by sending a 1 as acknowledge value. Depending on the internal state of the receiver, the acknowledge bit is either sent active or passive.

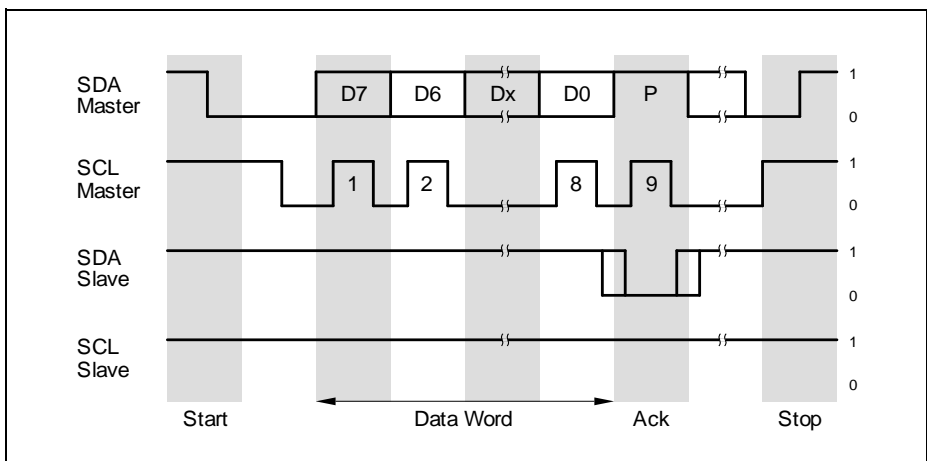


Figure 17-51 IIC Frame Example (simplified)

17.5.2 Operating the IIC

In order to operate the IIC protocol, the following issues have to be considered:

- **Select IIC mode:**
It is recommended to configure all parameters of the IIC that do not change during run time while $CCR.MODE = 0000_B$. Bit field $SCTR.TRM = 11_B$ should be programmed. The configuration of the input stages has to be done while $CCR.MODE = 0000_B$ to avoid unintended edges of the input signals and the IIC mode can be enabled by $CCR.MODE = 0100_B$ afterwards.
- **Pin connections:**
Establish a connection of input stage DX0 (with $DX0CR.DPOL = 0$) to the selected shift data pin SDA (signal DINO) with $DX0CR.INSW = 0$ and configure the transmit data output signal DOUT0 (with $SCTR.DOCFG = 00_B$) to the same pin. If available, this can be the same pin for input and output, or connect the selected input pin and the output pin to form the SDA line.
The same mechanism applies for the shift clock line SCL. Here, signal SCLKOUT (with $BRG.SCLKCFG = 00_B$) and an input of the DX1 stage have to be connected (with $DX1CR.DPOL = 0$).
The input stage DX2 is not used for the IIC protocol.
If the digital input filters are enabled in the DX0/1 stages, their delays have to be taken into account for correct calculation of the signal timings.
The pins used for SDA and SCL have to be set to open-drain mode to support the wired-AND structure of the IIC bus lines.
Note that the step to enable the alternate output port functions should only be done after the IIC mode is enabled, to avoid unintended spikes on the output.
- **Bit timing configuration:**
In standard mode (100 kBit/s) a minimum module frequency of 2 MHz is necessary, whereas in fast mode (400 kBit/s) a minimum of 10 MHz is required. Additionally, if the digital filter stage should be used to eliminate spikes up to 50 ns, a filter frequency of 20 MHz is necessary.
There could be an uncertainty in the SCL high phase timing of maximum $1/f_{PPP}$ if another IIC participant lengthens the SCL low phase on the bus.
More details are given in [Section 17.5.3](#).
- **Data format configuration:**
The data format has to be configured for 8 data bits ($SCTR.WLE = 7$), unlimited data flow ($SCTR.FLE = 3F_H$), and MSB shifted first ($SCTR.SDIR = 1$). The parity generation has to be disabled ($CCR.PM = 00_B$).
- **General hints:**
The IIC slave module becomes active (for reception or transmission) if it is selected by the address sent by the master. In the case that the slave sends data to the master, it uses the transmit path. So a master must not request to read data from the slave address defined for its own channel in order to avoid collisions.
The built-in error detection mechanisms are only activated while the IIC module is

Universal Serial Interface Channel (USIC)

taking part in IIC bus traffic.

If the slave can not deal with too high frequencies, it can lengthen the low phase of the SCL signal.

For data transfers according to the IIC specification, the shift data line SDA shall only change while $SCL = 0$ (defined by IIC bus specification).

17.5.2.1 Transmission Chain

The IIC bus protocol requiring a kind of in-bit-response during the arbitration phase and while a slave is transmitting, the resulting loop delay of the transmission chain can limit the reachable maximal baud rate, strongly depending on the bus characteristics (bus load, module frequency, etc.).

Figure 17-50 shows the general signal path and the delays in the case of a slave transmission. The shift clock SCL is generated by the master device, output on the wire, then it passes through the input stage and the input filter. Now, the edges can be detected and the SDA data signal can be generated accordingly. The SDA signal passes through the output stage and the wire to the master receiver part. There, it passes through the input stage and the input filter before it is sampled.

This complete loop has to be finished (including all settling times to obtain stable signal levels) before the SCL signal changes again. The delays in this path have to be taken into account for the calculation of the baud rate as a function of f_{PERIPH} and f_{PPP} .

17.5.2.2 Byte Stretching

If a device is selected as transceiver and should transmit a data byte but the transmit buffer TBUF does not contain valid data to be transmitted, the device ties down $SCL = 0$ at the end of the previous acknowledge bit. The waiting period is finished if new valid data has been detected in TBUF.

17.5.2.3 Master Arbitration

During the address and data transmission, the master transmitter checks at the rising edge of SCL for each data bit if the value it is sending is equal to the value read on the SDA line. If yes, the next data bit values can be 0. If this is not the case (transmitted value = 1, value read = 0), the master has lost the transmit arbitration. This is indicated by status flag PSR.ARL and can generate a protocol interrupt if enabled by PCR.ARLIEN.

When the transmit arbitration has been lost, the software has to initialize the complete frame again, starting with the first address byte together with the start condition for a new master transmit attempt. Arbitration also takes place for the ACK bit.

17.5.2.4 Non-Acknowledge and Error Conditions

In case of a non-acknowledge or an error, the TCSR.TDV flag remains set, but no further transmission will take place. User software must invalidate the transmit buffer and disable transmissions (by writing $\text{FMRL.MTDV} = 10_{\text{B}}$), before configuring the transmission (by writing TBUF) again with appropriate values to react on the previous event. In the case the FIFO data buffer is used, additionally the FIFO buffer needs to be flushed and filled again.

17.5.2.5 Mode Control Behavior

In multi-master mode, only run mode 0 and stop mode 0 are supported, the other modes must not be programmed.

- **Run Mode 0:**
Behavior as programmed. If $\text{TCSR.TDV} = 0$ (no new valid TBUF entry found) when a new TBUF entry needs to be processed, the IIC module waits for TDV becoming set to continue operation.
- **Run Mode 1:**
Behavior as programmed. If in master mode, $\text{TCSR.TDV} = 0$ (no new valid TBUF entry found) when a new TBUF entry needs to be processed, the IIC module sends a stop condition to finish the frame. In slave mode, no difference to run mode 0.
- **Stop Mode 0:**
Bit TCSR.TDV is internally considered as 0 (the bit itself is not modified by the stop mode). A currently running word is finished normally, but no new word is started in case of master mode (wait for TDV active).
Bit TDV being considered as 0 for master and slave, the slave will force a wait state on the bus if read by an external master, too.
Additionally, it is not possible to force the generation of a STOP condition out of the wait state. The reason is, that a master read transfer must be finished with a not-acknowledged followed by a STOP condition to allow the slave to release his SDA line. Otherwise the slave may force the SDA line to 0 (first data bit of next byte) making it impossible to generate the STOP condition (rising edge on SDA).
To continue operation, the mode must be switched to run mode 0
- **Stop Mode 1:**
Same as stop mode 0, but additionally, a master sends a STOP condition to finish the frame.
If stop mode 1 is requested for a master device after the first byte of a 10 bit address, a stop condition will be sent out. In this case, a slave device will issue an error interrupt.

17.5.2.6 Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to IIC frame handling. As the data input and output pins are the same in IIC protocol, a IIC transmitter also receives the

Universal Serial Interface Channel (USIC)

output data at its input pin. However, no receive related interrupts will be generated in this case.

- **Transmit buffer event:**
The transmit buffer event indication flag PSR.TBIF is set when the content of the transmit buffer TBUF has been loaded to the transmit shift register, indicating that the action requested by the TBUF entry has started.
With this event, bit TCSR.TDV is cleared. This interrupt can be used to write the next TBUF entry while the last one is in progress (handled by the transmitter part).
- **Receive event:**
This receive event indication flag PSR.RIF indicates that a new data byte has been written to the receive buffer RBUF0/1 (except for the first data byte of a new frame, that is indicated by an alternative receive interrupt). The flag becomes set when the data byte is received (after the falling edge of SCL). This interrupt can be used to read out the received data while a new data byte can be in progress (handled by the receiver part).
- **Alternate receive event:**
The alternative receive event indication flag AIF is based on bit RBUFSR[9] (same as RBUF[9]), indicating that the received data word has been the first data word of a new data frame.
- **Transmit shift event:**
The transmit shift event indication flag TSIF is set after the start of the last data bit of a data byte.
- **Receive start event:**
The receive start event indication flag RSIF is set after the sample point of the first data bit of a data byte.

Note: The transmit shift and receive start events can be ignored if the application does not require them during the IIC data transfer.

17.5.2.7 IIC Protocol Interrupt Events

The following protocol-related events are generated in IIC mode and can lead to a protocol interrupt.

Please note that the bits in register PSR are not all automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- start condition received at a correct position in a frame (PSR.SCR)
- repeated start condition received at a correct position in a frame (PSR.RSCR)
- stop condition transferred at a correct position in a frame (PSR.PCR)
- master arbitration lost (PSR.ARL)
- slave read requested (PSR.SRR)
- acknowledge received (PSR.ACK)
- non-acknowledge received (PSR.NACK)
- start condition not at the expected position in a frame (PSR.ERR)

Universal Serial Interface Channel (USIC)

- stop condition not at the expected position in a frame (PSR.ERR)
- as slave, 10-bit address interrupted by a stop condition after the first address byte (PSR.ERR)
- TDF slave code in master mode (PSR.WTDF)
- TDF master code in slave mode (PSR.WTDF)
- Reserved TDF code found (PSR.WDTF)
- Start condition code during a running frame in master mode (PSR.WTDF)
- Data byte transmission code after transfer direction has been changed to reception (master read) in master mode (PSR.WTDF)

If a wrong TDF code is found in TBUF, the error event is active until the TDF value is either corrected or invalidated. If the related interrupt is enabled, the interrupt handler should check PSR.WDTF first and correct or invalidate TBUF, before dealing with the other possible interrupt events.

17.5.2.8 Baud Rate Generator Interrupt Handling

The baud rate generator interrupt indicate that the capture mode timer has reached its maximum value. With this event, the bit PSR.BRGIF is set.

17.5.2.9 Receiver Address Acknowledge

After a (repeated) start condition, the master sends a slave address to identify the target device of the communication. The start address can comprise one or two address bytes (for 7 bit or for 10 bit addressing schemes). After an address byte, a slave sensitive to the transmitted address has to acknowledge the reception.

Therefore, the slave's address can be programmed in the device, where it is compared to the received address. In case of a match, the slave answers with an acknowledge (SDA = 0). Slaves that are not targeted answer with a non-acknowledge (SDA = 1).

In addition to the match of the programmed address, another address byte value has to be answered with an acknowledge if the slave is capable to handle the corresponding requests. The address byte 00_H indicates a general call address, that can be acknowledged. The value 01_H stands for a start byte generation, that is not acknowledged

In order to allow selective acknowledges for the different values of the address byte(s), the following control mechanism is implemented:

- The address byte 00_H is acknowledged if bit PCR.ACK00 is set.
- The address byte 01_H is not acknowledged.
- The first 7 bits of a received first address byte are compared to the programmed slave address (PCR.SLAD[15:9]). If these bits match, the slave sends an acknowledge. In addition to this, if the slave address is programmed to 1111 0XX_B, the slave device waits for a second address byte and compares it also to PCR.SLAD[7:0] and sends an acknowledge accordingly to cover the 10 bit addressing mode. The user has to

Universal Serial Interface Channel (USIC)

take care about reserved addresses (refer to IIC specification for more detailed description). Only the address 1111 0XX_b is supported.

Under each of these conditions, bit PSR.SLSEL will be set when the addressing delivered a match. This bit is cleared automatically by a (repeated) start condition.

17.5.2.10 Receiver Handling

A selected slave receiver always acknowledges a received data byte. If the receive buffers RBUF0/1 are already full and can not accept more data, the respective register is overwritten (PSR.DLI becomes set in this case and a protocol interrupt can be generated).

An address reception also uses the registers RBUF0/1 to store the address before checking if the device is selected. The received addresses do not set RDV0/1, so the addresses are not handled like received data.

17.5.2.11 Receiver Status Information

In addition to the received data byte, some IIC protocol related information is stored in the 16-bit data word of the receive buffer. The received data byte is available at the bit positions RBUF[7:0], whereas the additional information is monitored at the bit positions RBUF[12:8]. This structure allows to identify the meaning of each received data byte without reading additional registers, also when using a FIFO data buffer.

- RBUF[8]:
Value of the received acknowledge bit. This information is also available in RBUF_{SR}[8] as protocol argument.
- RBUF[9]:
A 1 at this bit position indicates that after a (repeated) start condition followed by the address reception the first data byte of a new frame has been received. A 0 at this bit position indicates further data bytes. This information is also available in RBUF_{SR}[9], allowing different interrupt routines for the address and data handling.
- RBUF[10]:
A 0 at this bit position indicates that the data byte has been received when the device has been in slave mode, whereas a 1 indicates a reception in master mode.
- RBUF[11]:
A 1 at this bit position indicates an incomplete/erroneous data byte in the receive buffer caused by a wrong position of a START or STOP condition in the frame. The bit is not identical to the frame error status bit in PSR, because the bit in the PSR has to be cleared by software ("sticky" bit), whereas RBUF[11] is evaluated data byte by data byte. If RBUF[11] = 0, the received data byte has been correct, independent of former errors.
- RBUF[12]:
A 0 at this bit position indicates that the programmed address has been received. A 1 indicates a general call address.

17.5.3 Symbol Timing

The symbol timing of the IIC is determined by the master stimulating the shift clock line SCL. It is different in each of the modes.

- 100 kBaud standard mode (PCR.STIM = 0):
The symbol timing is based on 10 time quanta t_q per symbol. A minimum module clock frequency $f_{\text{PERIPH}} = 2 \text{ MHz}$ is required.
- 400 kBaud fast mode (PCR.STIM = 1):
The symbol timing is based on 25 time quanta t_q per symbol. A minimum module clock frequency $f_{\text{PERIPH}} = 10 \text{ MHz}$ is required.

The baud rate setting should only be changed while the transmitter and the receiver are idle or CCR.MODE = 0. The bits in register BRG define the length of a time quantum t_q that is given by one period of f_{PCTQ} .

- BRG.CTQSEL
to define the input frequency f_{CTQIN} for the time quanta generation
- BRG.PCTQ
to define the length of a time quantum (division of f_{CTQIN} by 1, 2, 3, or 4)
- BRG.DCTQ
to define the number of time quanta per symbol (number of $t_q = \text{DCTQ} + 1$)

The standard setting is given by CTQSEL = 00_B ($f_{\text{CTQIN}} = f_{\text{PDIV}}$) and PPPEN = 0 ($f_{\text{PPP}} = f_{\text{IN}}$). Under these conditions, the frequency f_{PCTQ} is given by:

$$f_{\text{PCTQ}} = f_{\text{PIN}} \times \frac{1}{\text{PDIV} + 1} \times \frac{1}{\text{PCTQ} + 1} \quad (17.10)$$

To respect the specified SDA hold time of 300 ns for standard mode and fast mode after a falling edge of signal SCL, a hold delay t_{HDEL} has been introduced. It also prevents an erroneous detection of a start or a stop condition. The length of this delay can be programmed by bit field PCR.HDEL. Taking into account the input sampling and output update, bit field HDEL can be programmed according to:

(17.11)

$$\text{HDEL} \geq 300 \text{ ns} \times f_{\text{PPP}} - \left(3 \times \frac{f_{\text{PPP}}}{f_{\text{PERIPH}}} \right) + 1 \quad \text{with digital filter and HDELmin} = 2$$

$$\text{HDEL} \geq 300 \text{ ns} \times f_{\text{PPP}} - \left(3 \times \frac{f_{\text{PPP}}}{f_{\text{PERIPH}}} \right) + 2 \quad \text{without digital filter and HDELmin} = 1$$

If the digital input filter is used, HDEL compensates the filter delay of 2 filter periods (f_{PPP} should be used) in case of a spike on the input signal. This ensures that a data bit on the SDA line changing just before the rising edge or behind the falling edge of SCL will not be treated as a start or stop condition.

17.5.3.1 Start Symbol

Figure 17-52 shows the general start symbol timing.

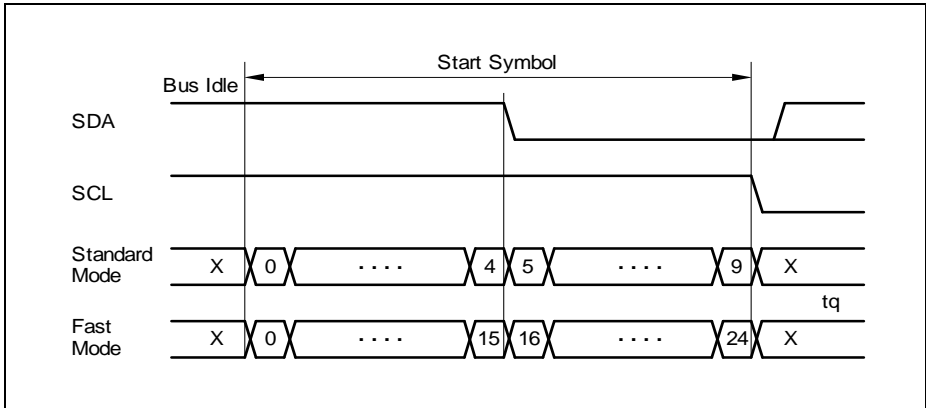


Figure 17-52 Start Symbol Timing

17.5.3.2 Repeated Start Symbol

During the first part of a repeated start symbol, an SCL low value is driven for the specified number of time quanta. Then a high value is output. After the detection of a rising edge at the SCL input, a normal start symbol is generated, as shown in Figure 17-53.

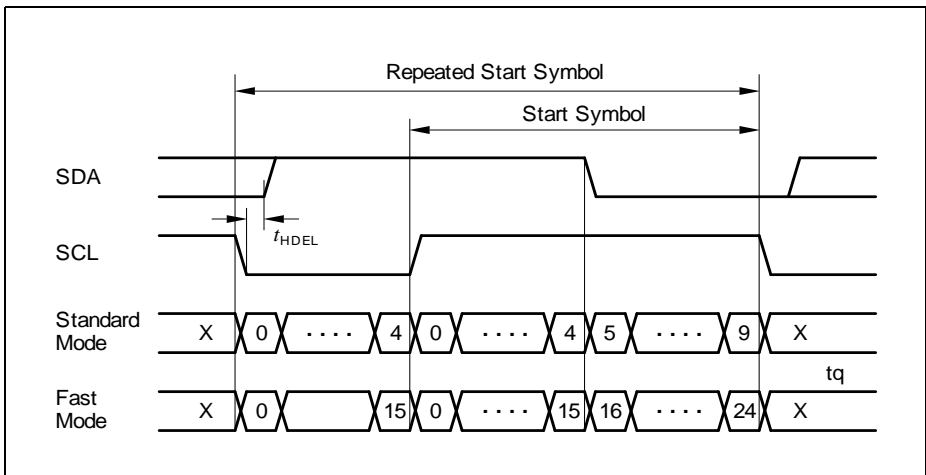


Figure 17-53 Repeated Start Symbol Timing

17.5.3.3 Stop Symbol

Figure 17-54 shows the stop symbol timing.

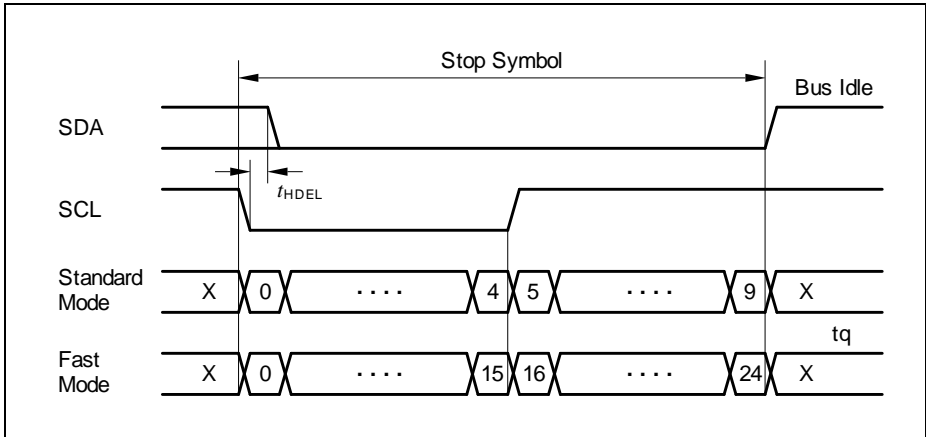


Figure 17-54 Stop Symbol Timing

17.5.3.4 Data Bit Symbol

Figure 17-55 shows the general data bit symbol timing.

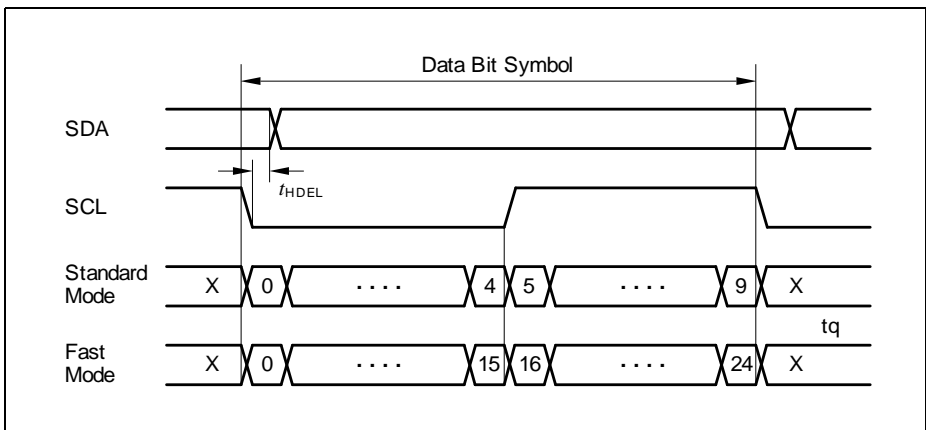


Figure 17-55 Data Bit Symbol

Output SDA changes after the time t_{HDEL} defined by PCR.HDEL has elapsed if a falling edge is detected at the SCL input to respect the SDA hold time. The value of PCR.HDEL allows compensation of the delay of the SCL input path (sampling, filtering).

Universal Serial Interface Channel (USIC)

In the case of an acknowledge transmission, the USIC IIC waits for the receiver indicating that a complete byte has been received. This adds an additional delay of 3 periods of f_{PERIPH} to the path. The minimum module input frequency has to be selected properly to ensure the SDA setup time to SCL rising edge.

17.5.4 Data Flow Handling

The handling of the data flow and the sequence of the symbols in an IIC frame is controlled by the IIC transmitter part of the USIC communication channel. The IIC bus protocol is byte-oriented, whereas a USIC data buffer word can contain up to 16 data bits. In addition to the data byte to be transmitted (located at TBUF[7:0]), bit field TDF (transmit data format) to control the IIC sequence is located at the bit positions TBUF[10:8]. The TDF code defines for each data byte how it should be transmitted (IIC master or IIC slave), and controls the transmission of (repeated) start and stop symbols. This structure allows the definition of a complete IIC frame for an IIC master device only by writing to TBUFx or by using a FIFO data buffer mechanism, because no other control registers have to be accessed. Alternatively, polling of the ACK and NACK bits in PSR register can be performed, and the next data byte is transmitted only after an ACK is received.

If a wrong or unexpected TDF code is encountered (e.g. due to a software error during setup of the transmit buffer), a stop condition will be sent out by the master. This leads to an abort of the currently running frame. A slave module waits for a valid TDF code and sets SCL = 0. The software then has to invalidate the unexpected TDF code and write a valid one.

Please note that during an arbitration phase in multi-master bus systems an unpredictable bus behavior may occur due to an unexpected stop condition.

17.5.4.1 Transmit Data Formats

The following transmit data formats are available in master mode:

Table 17-13 Master Transmit Data Formats

TDF Code	Description
000 _B	Send data byte as master This format is used to transmit a data byte from the master to a slave. The transmitter sends its data byte (TBUF[7:0]), receives and checks the acknowledge bit sent by the slave.
010 _B	Receive data byte and send acknowledge This format is used by the master to read a data byte from a slave. The master acknowledges the transfer with a 0-level to continue the transfer. The content of TBUF[7:0] is ignored.

Universal Serial Interface Channel (USIC)

Table 17-13 Master Transmit Data Formats (cont'd)

TDF Code	Description
011 _B	Receive data byte and send not-acknowledge This format is used by the master to read a data byte from a slave. The master does not acknowledge the transfer with a 1-level to finish the transfer. The content of TBUF[7:0] is ignored.
100 _B	Send start condition If TBUF contains this entry while the bus is idle, a start condition will be generated. The content of TBUF[7:0] is taken as first address byte for the transmission (bits TBUF[7:1] are the address, the LSB is the read/write control).
101 _B	Send repeated start condition If TBUF contains this entry and SCL = 0 and a byte transfer is not in progress, a repeated start condition will be sent out if the device is the current master. The current master is defined as the device that has set the start condition (and also won the master arbitration) for the current message. The content of TBUF[7:0] is taken as first address byte for the transmission (bits TBUF[7:1] are the address, the LSB is the read/write control).
110 _B	Send stop condition If the current master has finished its last byte transfer (including acknowledge), it sends a stop condition if this format is in TBUF. The content of TBUF[7:0] is ignored.
111 _B	Reserved This code must not be programmed. No additional action except releasing the TBUF entry and setting the error bit in PSR (that can lead to a protocol interrupt).

The following transmit data format is available in slave mode (the symbols in a frame are controlled by the master and the slave only has to send data if it has been “asked” by the master):

Table 17-14 Slave Transmit Data Format

TDF Code	Description
001 _B	Send data byte as slave This format is used to transmit a data byte from a slave to the master. The transmitter sends its data byte (TBUF[7:0]) plus the acknowledge bit as a 1.

Universal Serial Interface Channel (USIC)

17.5.4.2 Valid Master Transmit Data Formats

Due to the IIC frame format definitions, only some specific sequences of TDF codes are possible and valid. If the USIC IIC module detects a wrong TDF code in a running frame, the transfer is aborted and flag PCR.WTDF is set. Additionally, an interrupt can be generated if enabled by the user. In case of a wrong TDF code, the frame will be aborted immediately with a STOP condition if the USIC IIC master still owns the SDA line. But if the accessed slave owns the SDA line (read transfer), the master must perform a dummy read with a non-acknowledge so that the slave releases the SDA line before a STOP condition can be sent. The received data byte of the dummy read will be stored in RBUF0/1, but RDV0/1 will not be set. Therefore the dummy read will not generate a receive interrupt and the data byte will not be stored into the receive FIFO.

If the transfer direction has changed in the current frame (master read access), the transmit data request (TDF = 000_B) is not possible and won't be accepted (leading to a wrong TDF Code indication).

Table 17-15 Valid TDF Codes Overview

Frame Position	Valid TDF Codes
First TDF code (master idle)	Start (100 _B)
Read transfer: second TDF code (after start or repeated start)	Receive with acknowledge (010 _B) or receive with not-acknowledge (011 _B)
Write transfer: second TDF code (after start or repeated start)	Transmit (000 _B), repeated start (101 _B), or stop (110 _B)
Read transfer: third and subsequent TDF code after acknowledge	Receive with acknowledge (010 _B) or receive with not-acknowledge (011 _B)
Read transfer: third and subsequent TDF code after not-acknowledge	Repeated start (101 _B) or stop (110 _B)
Write transfer: third and subsequent TDF code	Transmit (000 _B), repeated start (101 _B), or stop (110 _B)

- First TDF code:
A master transfer starts with the TDF start code (100_B). All other codes are ignored, but no WTDF error will be indicated.
- TDF code after a start (100_B) or repeated start code (101_B) in case of a read access:
If a master-read transfer is started (determined by the LSB of the address byte = 1), the transfer direction of SDA changes and the slave will actively drive the data line. In this case, only the codes 010_B and 011_B are valid. To abort the transfer in case of a wrong code, a dummy read must be performed by the master before the STOP condition can be generated.

Universal Serial Interface Channel (USIC)

- TDF code after a start (100_B) or repeated start code (101_B) in case of a write access: If a master-write transfer is started (determined by the LSB of the address byte = 0), the master still owns the SDA line. In this case, the transmit (000_B), repeated start (101_B) and stop (110_B) codes are valid. The other codes are considered as wrong. To abort the transfer in case of a wrong code, the STOP condition is generated immediately.
- TDF code of the third and subsequent command in case of a read access with acknowledged previous data byte: If a master-read transfer is started (determined by the LSB of the address byte), the transfer direction of SDA changes and the slave will actively drive the data line. To force the slave to release the SDA line, the master has to not-acknowledge a byte transfer. In this case, only the receive codes 010_B and 011_B are valid. To abort the transfer in case of a wrong code, a dummy read must be performed by the master before the STOP condition can be generated.
- TDF code of the third and subsequent command in case of a read access with a not-acknowledged previous data byte: If a master-read transfer is started (determined by the LSB of the address byte), the transfer direction of SDA changes and the slave will actively drive the data line. To force the slave to release the SDA line, the master has to not-acknowledge a byte transfer. In this case, only the restart (101_B) and stop code (110_B) are valid. To abort the transfer in case of a wrong code, the STOP condition is generated immediately.
- TDF code of the third and subsequent command in case of a write access: If a master-write transfer is started (determined by the LSB of the address byte), the master still owns the SDA line. In this case, the transmit (000_B), repeated start (101_B) and stop (110_B) codes are valid. The other codes are considered as wrong. To abort the transfer in case of a wrong code, the STOP condition is generated immediately.
- After a master device has received a non-acknowledge from a slave device, a stop condition will be sent out automatically, except if the following TDF code requests a repeated start condition. In this case, the TDF code is taken into account, whereas all other TDF codes are ignored.

Universal Serial Interface Channel (USIC)

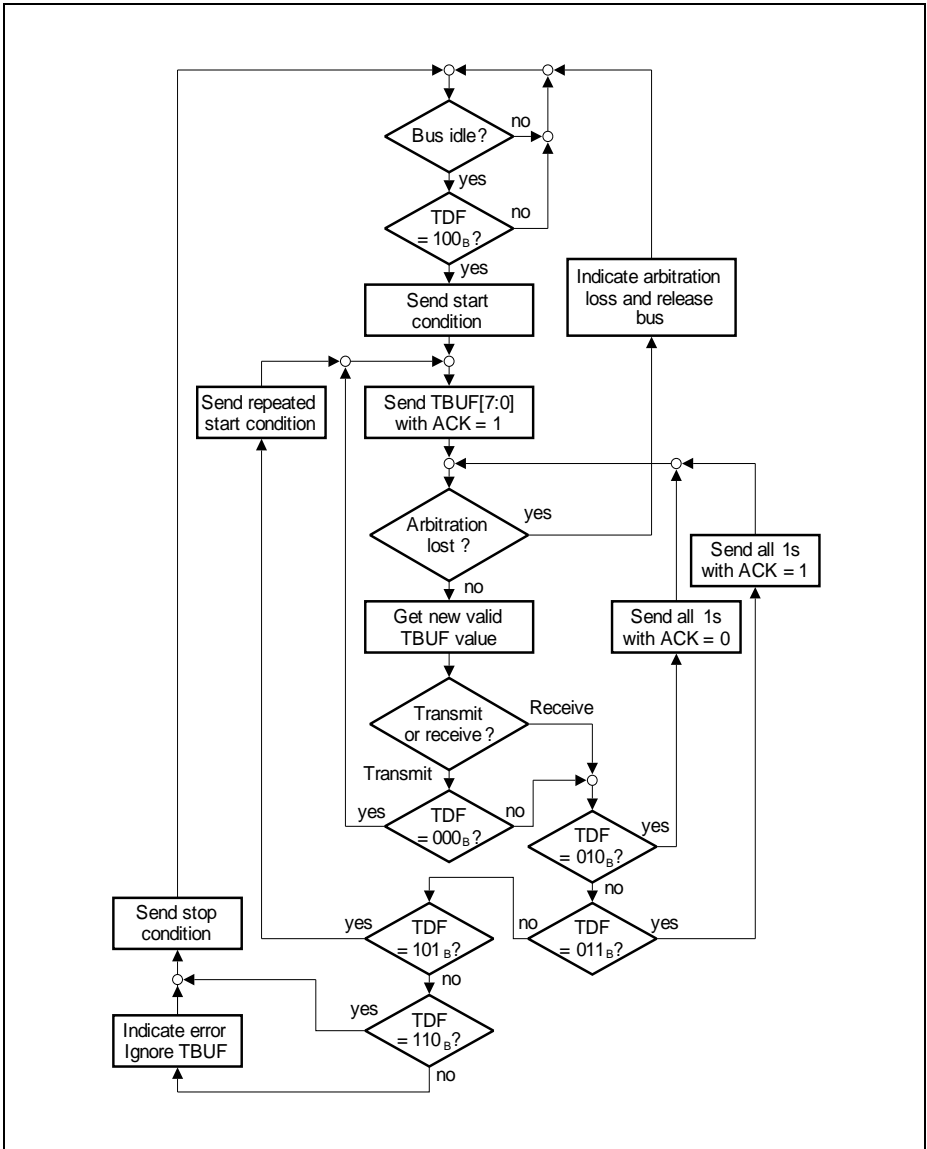


Figure 17-56 IIC Master Transmission

Universal Serial Interface Channel (USIC)

17.5.4.3 Master Transmit/Receive Modes

In master transmit mode, the IIC sends a number of data bytes to a slave receiver. The TDF code sequence for the master transmit mode is shown in [Table 17-16](#).

Table 17-16 TDF Code Sequence for Master Transmit

TDF Code Sequence	TBUF[10:8] (TDF Code)	TBUF[7:0]	IIC Response	Interrupt Events
1st code	100 _B	Slave address + write bit	Send START condition, slave address and write bit	SCR: Indicates a START condition is detected TBIF: Next word can be written to TBUF
2nd code	000 _B	Data or 2nd slave address byte	Send data or 2nd slave address byte	TBIF: Next word can be written to TBUF
Subsequent codes for data transmit	000 _B	Data	Send data	TBIF: Next word can be written to TBUF
Last code	110 _B	Don't care	Send STOP condition	PCR: Indicates a STOP condition is detected

In master receive mode, the IIC receives a number of data bytes from a slave transmitter. The TDF code sequence for the master receive 7-bit and 10-bit addressing modes are shown in [Table 17-17](#) and [Table 17-18](#).

Table 17-17 TDF Code Sequence for Master Receive (7-bit Addressing Mode)

TDF Code Sequence	TBUF[10:8] (TDF Code)	TBUF[7:0]	IIC Response	Interrupt Events
1st code	100 _B	Slave address + read bit	Send START condition, slave address and read bit	SCR: Indicates a START condition is detected TBIF: Next word can be written to TBUF
2nd code	010 _B	Don't care	Receive data and send ACK bit	TBIF: Next word can be written to TBUF AIF: First data received can be read

Universal Serial Interface Channel (USIC)

Table 17-17 TDF Code Sequence for Master Receive (7-bit Addressing Mode)

TDF Code Sequence	TBUF[10:8] (TDF Code)	TBUF[7:0]	IIC Response	Interrupt Events
Subsequent codes for data receive	010 _B	Don't care	Receive data and send ACK bit	TBIF: Next word can be written to TBUF RIF: Subsequent data received can be read
Code for last data to be received	011 _B	Don't care	Receive data and send NACK bit	TBIF: Next word can be written to TBUF RIF: Last data received can be read
Last code	110 _B	Don't care	Send STOP condition	PCR: Indicates a STOP condition is detected

Table 17-18 TDF Code Sequence for Master Receive (10-bit Addressing Mode)

TDF Code Sequence	TBUF[10:8] (TDF Code)	TBUF[7:0]	IIC Response	Interrupt Events
1st code	100 _B	Slave address (1st byte) + write bit	Send START condition, slave address (1st byte) and write bit	SCR: Indicates a START condition is detected TBIF: Next word can be written to TBUF
2nd code	000 _B	Slave address (2nd byte)	Send address (2nd byte)	TBIF: Next word can be written to TBUF
3rd code	101 _B	1st slave address + read bit	Send repeated START condition, slave address (1st byte) and read bit	RSCR: Indicates a repeated START condition is detected TBIF: Next word can be written to TBUF
4th code	010 _B	Don't care	Receive data and send ACK bit	TBIF: Next word can be written to TBUF AIF: First data received can be read
Subsequent codes for data receive	010 _B	Don't care	Receive data and send ACK bit	TBIF: Next word can be written to TBUF RIF: Subsequent data received can be read

Universal Serial Interface Channel (USIC)

Table 17-18 TDF Code Sequence for Master Receive (10-bit Addressing Mode)

TDF Code Sequence	TBUF[10:8] (TDF Code)	TBUF[7:0]	IIC Response	Interrupt Events
Code for last data to be received	011 _B	Don't care	Receive data and send NACK bit	TBIF: Next word can be written to TBUF RIF: Last data received from slave can be read
Last code	110 _B	Don't care	Send STOP condition	PCR: Indicates a STOP condition is detected

Figure 17-57 shows the interrupt events during the master transmit-slave receive and master receive/slave transmit sequences.

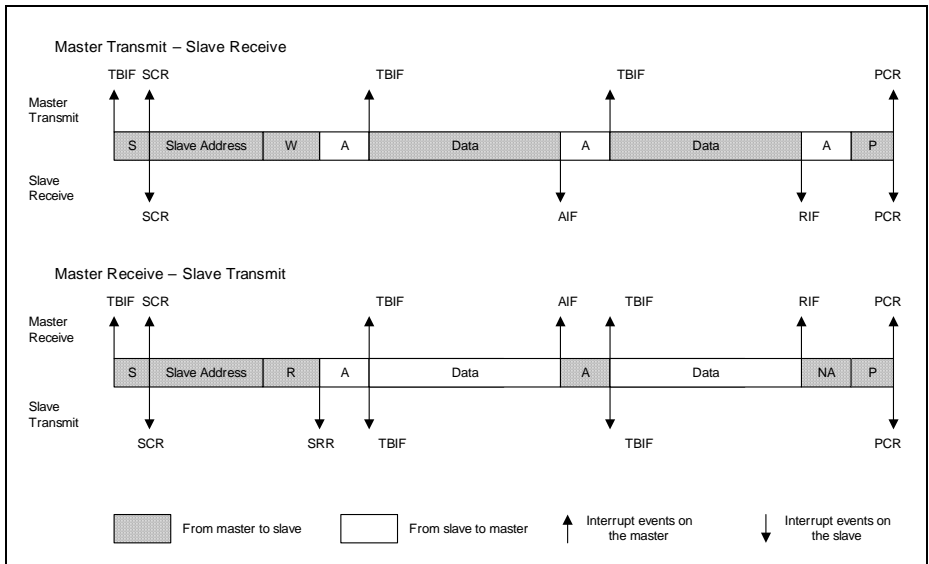


Figure 17-57 Interrupt Events on Data Transfers

17.5.4.4 Slave Transmit/Receive Modes

In slave receive mode, no TDF code needs to be written and data reception is indicated by the alternate receive (AIF) or receive (RIF) events.

In slave transmit mode, upon receiving its own slave address or general call address if this option is enabled, a slave read request event (SRR) will be triggered. The slave IIC then writes the TDF code 001_B and the requested data to TBUF to transmit the data to

Universal Serial Interface Channel (USIC)

the master. The slave does not check if the master reply with an ACK or NACK to the transmitted data.

In both cases, the data transfer is terminated by the master sending a STOP condition, which is indicated by a PCR event. See also [Figure 17-57](#).

17.5.5 IIC Protocol Registers

In IIC mode, the registers PCR and PSR handle IIC related information.

17.5.5.1 IIC Protocol Control Registers

In IIC mode, the PCR register bits or bit fields are defined as described in this section.

PCR

Protocol Control Register [IIC Mode]

(3C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCLK	ACKIEN		HDEL			SACKDIS	ERRIEN	SRRIEN	ARLIEN	NACKIEN	PCRRIEN	RSCRIEN	SCRRIEN	STIM	ACK00
rw	rw		rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLAD															
rw															

Field	Bits	Type	Description
SLAD	[15:0]	rw	Slave Address This bit field contains the programmed slave address. The corresponding bits in the first received address byte are compared to the bits SLAD[15:9] to check for address match. If SLAD[15:11] = 11110 _B , then the second address byte is also compared to SLAD[7:0].
ACK00	16	rw	Acknowledge 00_H This bit defines if a slave device should be sensitive to the slave address 00 _H . 0 _B The slave device is not sensitive to this address. 1 _B The slave device is sensitive to this address.

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
STIM	17	rw	<p>Symbol Timing This bit defines how many time quanta are used in a symbol.</p> <p>0_B A symbol contains 10 time quanta. The timing is adapted for standard mode (100 kBaud).</p> <p>1_B A symbol contains 25 time quanta. The timing is adapted for fast mode (400 kBaud).</p>
SCRIEN	18	rw	<p>Start Condition Received Interrupt Enable This bit enables the generation of a protocol interrupt if a start condition is detected.</p> <p>0_B The start condition interrupt is disabled.</p> <p>1_B The start condition interrupt is enabled.</p>
RSCRIEN	19	rw	<p>Repeated Start Condition Received Interrupt Enable This bit enables the generation of a protocol interrupt if a repeated start condition is detected.</p> <p>0_B The repeated start condition interrupt is disabled.</p> <p>1_B The repeated start condition interrupt is enabled.</p>
PCRIEN	20	rw	<p>Stop Condition Received Interrupt Enable This bit enables the generation of a protocol interrupt if a stop condition is detected.</p> <p>0_B The stop condition interrupt is disabled.</p> <p>1_B The stop condition interrupt is enabled.</p>
NACKIEN	21	rw	<p>Non-Acknowledge Interrupt Enable This bit enables the generation of a protocol interrupt if a non-acknowledge is detected by a master.</p> <p>0_B The non-acknowledge interrupt is disabled.</p> <p>1_B The non-acknowledge interrupt is enabled.</p>
ARLIEN	22	rw	<p>Arbitration Lost Interrupt Enable This bit enables the generation of a protocol interrupt if an arbitration lost event is detected.</p> <p>0_B The arbitration lost interrupt is disabled.</p> <p>1_B The arbitration lost interrupt is enabled.</p>
SRRIEN	23	rw	<p>Slave Read Request Interrupt Enable This bit enables the generation of a protocol interrupt if a slave read request is detected.</p> <p>0_B The slave read request interrupt is disabled.</p> <p>1_B The slave read request interrupt is enabled.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
ERRIEN	24	rw	<p>Error Interrupt Enable</p> <p>This bit enables the generation of a protocol interrupt if an IIC error condition is detected (indicated by PSR.ERR or PSR.WTDF).</p> <p>0_B The error interrupt is disabled. 1_B The error interrupt is enabled.</p>
SACKDIS	25	rw	<p>Slave Acknowledge Disable</p> <p>This bit disables the generation of an active acknowledge signal for a slave device (active acknowledge = 0 level). Once set by software, it is automatically cleared with each (repeated) start condition. If this bit is set after a byte has been received (indicated by an interrupt) but before the next acknowledge bit has started, the next acknowledge bit will be sent with passive level. This would indicate that the receiver does not accept more bytes. As a result, a minimum of 2 bytes will be received if the first receive interrupt is used to set this bit.</p> <p>0_B The generation of an active slave acknowledge is enabled (slave acknowledge with 0 level = more bytes can be received). 1_B The generation of an active slave acknowledge is disabled (slave acknowledge with 1 level = reception stopped).</p>
HDEL	[29:26]	rw	<p>Hardware Delay</p> <p>This bit field defines the delay used to compensate the internal treatment of the SCL signal (see Page 17-118) in order to respect the SDA hold time specified for the IIC protocol.</p>
ACKIEN	30	rw	<p>Acknowledge Interrupt Enable</p> <p>This bit enables the generation of a protocol interrupt if an acknowledge is detected by a master.</p> <p>0_B The acknowledge interrupt is disabled. 1_B The acknowledge interrupt is enabled.</p>
MCLK	31	rw	<p>Master Clock Enable</p> <p>This bit enables generation of the master clock MCLK (not directly used for IIC protocol, can be used as general frequency output).</p> <p>0_B The MCLK generation is disabled and MCLK is 0. 1_B The MCLK generation is enabled.</p>

Universal Serial Interface Channel (USIC)

17.5.5.2 IIC Protocol Status Register

The following PSR status bits or bit fields are available in IIC mode. Please note that the bits in register PSR are not cleared by hardware.

The flags in the PSR register can be cleared by writing a 1 to the corresponding bit position in register PSCR. Writing a 1 to a bit position in PSR sets the corresponding flag, but does not lead to further actions (no interrupt generation). Writing a 0 has no effect. These flags should be cleared by software before enabling a new protocol.

PSR

Protocol Status Register [IIC Mode] (48_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0															BRG IF	
r															rwh	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
AIF	RIF	TBIF	TSIF	DLIF	RSIF	ACK	ERR	SRR	ARL	NAC K	PCR	RSC R	SCR	WTD F	SLS EL	
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
SLSEL	0	rwh	Slave Select This bit indicates that this device has been selected as slave. 0 _B The device is not selected as slave. 1 _B The device is selected as slave.
WTDF	1	rwh	Wrong TDF Code Found¹⁾ This bit indicates that an unexpected/wrong TDF code has been found. A protocol interrupt can be generated if PCR.ERRIEN = 1. 0 _B A wrong TDF code has not been found. 1 _B A wrong TDF code has been found.
SCR	2	rwh	Start Condition Received¹⁾ This bit indicates that a start condition has been detected on the IIC bus lines. A protocol interrupt can be generated if PCR.SCRIEN = 1. 0 _B A start condition has not yet been detected. 1 _B A start condition has been detected.

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
RSCR	3	rwh	<p>Repeated Start Condition Received¹⁾ This bit indicates that a repeated start condition has been detected on the IIC bus lines. A protocol interrupt can be generated if PCR.RSCRIEN = 1.</p> <p>0_B A repeated start condition has not yet been detected. 1_B A repeated start condition has been detected.</p>
PCR	4	rwh	<p>Stop Condition Received¹⁾ This bit indicates that a stop condition has been detected on the IIC bus lines. A protocol interrupt can be generated if PCR.PCRIEN = 1.</p> <p>0_B A stop condition has not yet been detected. 1_B A stop condition has been detected.</p>
NACK	5	rwh	<p>Non-Acknowledge Received¹⁾ This bit indicates that a non-acknowledge has been received in master mode. This bit is not set in slave mode. A protocol interrupt can be generated if PCR.NACKIEN = 1.</p> <p>0_B A non-acknowledge has not been received. 1_B A non-acknowledge has been received.</p>
ARL	6	rwh	<p>Arbitration Lost¹⁾ This bit indicates that an arbitration has been lost. A protocol interrupt can be generated if PCR.ARLIEN = 1.</p> <p>0_B An arbitration has not been lost. 1_B An arbitration has been lost.</p>
SRR	7	rwh	<p>Slave Read Request¹⁾ This bit indicates that a slave read request has been detected. It becomes active to request the first data byte to be made available in the transmit buffer. For further consecutive data bytes, the transmit buffer issues more interrupts. For the end of the transfer, the master transmitter sends a stop condition. A protocol interrupt can be generated if PCR.SRRIEN = 1.</p> <p>0_B A slave read request has not been detected. 1_B A slave read request has been detected.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
ERR	8	rwh	Error¹⁾ This bit indicates that an IIC error (frame format or TDF code) has been detected. A protocol interrupt can be generated if PCR.ERRIEN = 1. 0 _B An IIC error has not been detected. 1 _B An IIC error has been detected.
ACK	9	rwh	Acknowledge Received¹⁾ This bit indicates that an acknowledge has been received in master mode. This bit is not set in slave mode. A protocol interrupt can be generated if PCR.ACKIEN = 1. 0 _B An acknowledge has not been received. 1 _B An acknowledge has been received.
RSIF	10	rwh	Receiver Start Indication Flag 0 _B A receiver start event has not occurred. 1 _B A receiver start event has occurred.
DLIF	11	rwh	Data Lost Indication Flag 0 _B A data lost event has not occurred. 1 _B A data lost event has occurred.
TSIF	12	rwh	Transmit Shift Indication Flag 0 _B A transmit shift event has not occurred. 1 _B A transmit shift event has occurred.
TBIF	13	rwh	Transmit Buffer Indication Flag 0 _B A transmit buffer event has not occurred. 1 _B A transmit buffer event has occurred.
RIF	14	rwh	Receive Indication Flag 0 _B A receive event has not occurred. 1 _B A receive event has occurred.
AIF	15	rwh	Alternative Receive Indication Flag 0 _B An alternative receive event has not occurred. 1 _B An alternative receive event has occurred.
BRGIF	16	rwh	Baud Rate Generator Indication Flag 0 _B A baud rate generator event has not occurred. 1 _B A baud rate generator event has occurred.
0	[31:17]	r	Reserved Returns 0 if read; not modified in IIC mode.

1) This status bit can generate a protocol interrupt (see [Page 17-22](#)). The general interrupt status flags are described in the general interrupt chapter.

17.6 Inter-IC Sound Bus Protocol (IIS)

This chapter describes how the USIC module handles the IIS protocol. This serial protocol can handle reception and transmission of synchronous data frames between a device operating in master mode and a device in slave mode. An IIS connection based on a USIC communication channel supports half-duplex and full-duplex data transfers. The IIS mode is selected by $CCR.MODE = 0011_B$ with $CCFG.IIS = 1$ (IIS mode is available).

17.6.1 Introduction

The IIS protocol is a synchronous serial communication protocol mainly for audio and infotainment applications [21].

17.6.1.1 Signal Description

A connection between an IIS master and an IIS slave is based on the following signals:

- A shift clock signal SCK, generated by the transfer master. It is permanently generated while an IIS connection is established, also while no valid data bits are transferred.
- A word address signal WA (also named WS), generated by the transfer master. It indicates the beginning of a new data word and the targeted audio channel (e.g. left/right). The word address output signal WA is available on all SELOx outputs if the WA generation is enabled (by $PCR.WAGEN = 1$ for the transfer master). The WA signal changes synchronously to the falling edges of the shift clock.
- If the transmitter is the IIS master device, it generates a master transmit slave receive data signal. The data changes synchronously to the falling edges of the shift clock.
- If the transmitter is the IIS slave device, it generates a master receive slave transmit data signal. The data changes synchronously to the falling edges of the shift clock.

The transmitter part and the receiver part of the USIC communication channel can be used together to establish a full-duplex data connection between an IIS master and a slave device.

Table 17-19 IIS IO Signals

IIS Mode	Receive Data	Transmit Data	Shift Clock	Word Address
Master	Input DIN0, handled by DX0	Output DOUT0	Output SCLKOUT	Output(s) SELOx
Slave	Input DIN0, handled by DX0	Output DOUT0	Input SCLKIN, handled by DX1	Input SELIN, handled by DX2

Universal Serial Interface Channel (USIC)

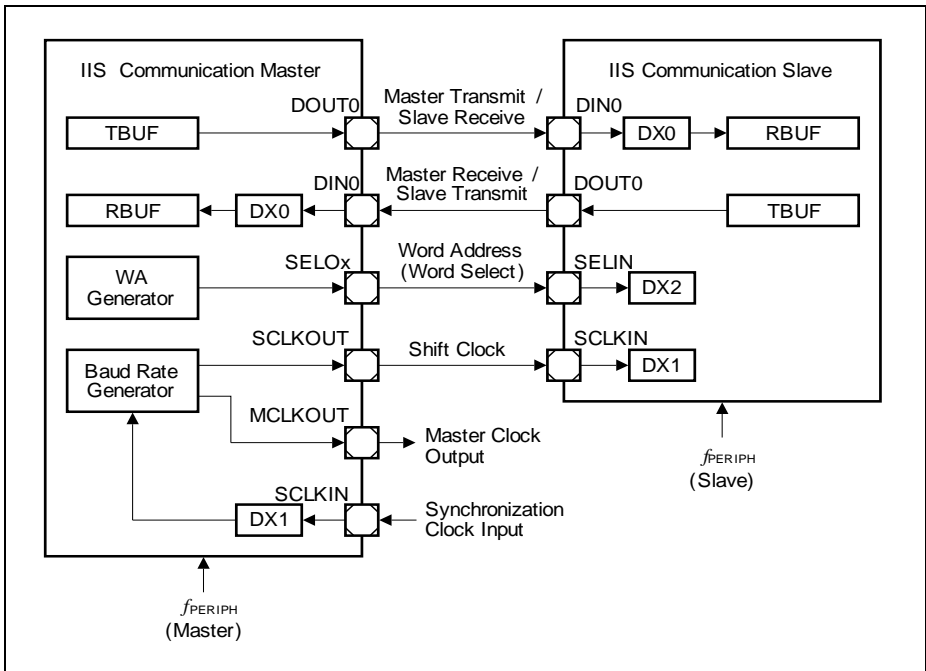


Figure 17-58 IIS Signals

Two additional signals are available for the USIC IIS communication master:

- A master clock output signal MCLKOUT with a fixed phase relation to the shift clock to support oversampling for audio components. It can also be used as master clock output of a communication network with synchronized IIS connections.
- A synchronization clock input SCLKIN for synchronization of the shift clock generation to an external frequency to support audio frequencies that can not be directly derived from the system clock f_{PERIPH} of the communication master. It can be used as master clock input of a communication network with synchronized IIS connections.

17.6.1.2 Protocol Overview

An IIS connection supports transfers for two different data frames via the same data line, e.g. a data frames for the left audio channel and a data frame for the right audio channel. The word address signal WA is used to distinguish between the different data frames. Each data frame can consist of several data words.

Universal Serial Interface Channel (USIC)

In a USIC communication channel, data words are tagged for being transmitted for the left or for the right channel. Also the received data words contain a tag identifying the WA state when the data has been received.

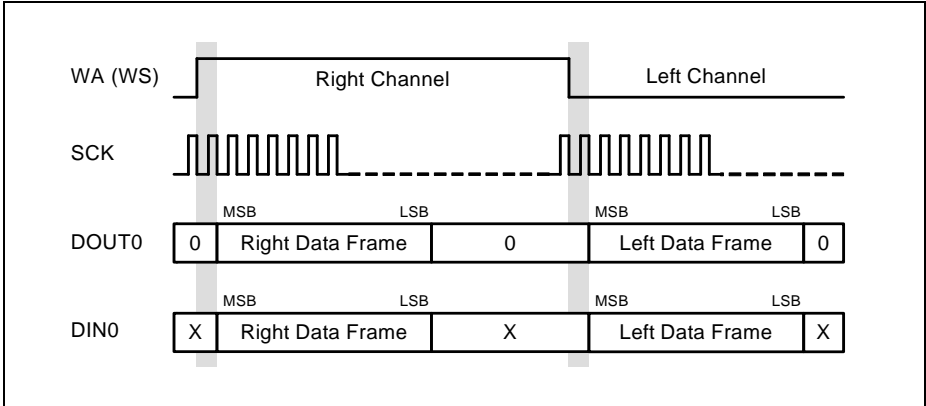


Figure 17-59 Protocol Overview

17.6.1.3 Transfer Delay

The transfer delay feature allows the transfer of data (transmission and reception) with a programmable delay (counted in shift clock periods).

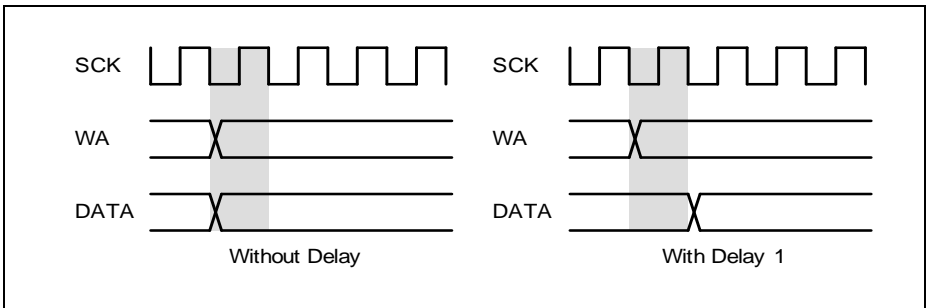


Figure 17-60 Transfer Delay for IIS

17.6.1.4 Connection of External Audio Components

The IIS signals can be used to communicate with external audio devices (such as Codecs) or other audio data sources/destinations.

Universal Serial Interface Channel (USIC)

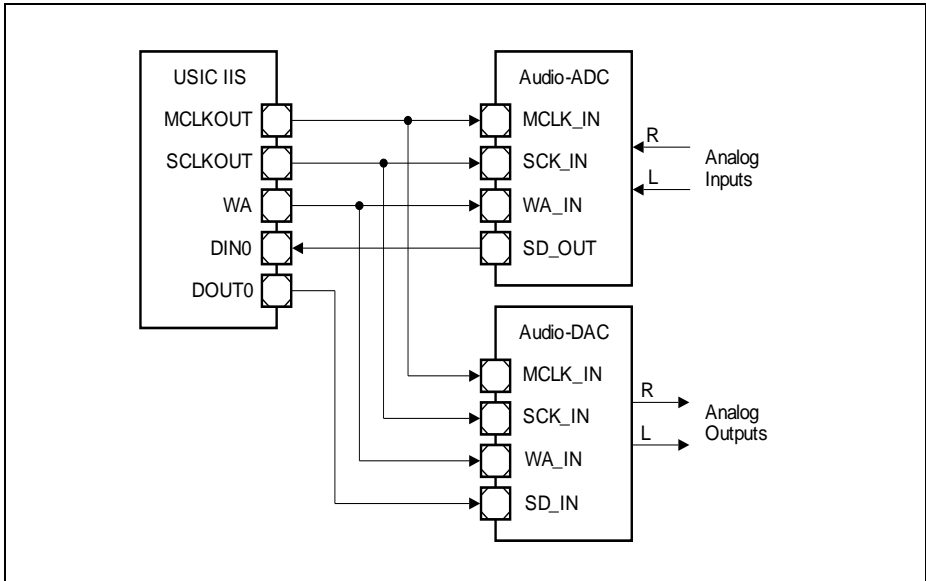


Figure 17-61 Connection of External Audio Devices

In some applications, especially for Audio-ADCs or Audio-DACs, a master clock signal is required with a fixed phase relation to the shift clock signal. The frequency of MCLKOUT is a multiple of the shift frequency SCLKOUT. This factor defines the oversampling factor of the external device (commonly used values: 256 or 384).

17.6.2 Operating the IIS

This chapter contains IIS issues, that are of general interest and not directly linked to master mode or slave mode.

17.6.2.1 Frame Length and Word Length Configuration

After each change of the WA signal, a complete data frame is intended to be transferred (frame length \leq system word length). The number of data bits transferred after a change of signal WA is defined by SCTR.FLE. A data frame can consist of several data words with a data word length defined by SCTR.WLE. The changes of signal WA define the system word length as the number of SCLK cycles between two changes of WA (number of bits available for the right channel and same number available for the left channel).

If the system word length is longer than the frame length defined by SCTR.FLE, the additional bits are transmitted with passive data level (SCTR.PDL). If the system word

Universal Serial Interface Channel (USIC)

length is smaller than the device frame length, not all LSBs of the transmit data can be transferred.

It is recommended to program bits WLEMD, FLEMD and SELMD in register TCSR to 0.

17.6.2.2 Automatic Shadow Mechanism

The baud rate and shift control setting are internally kept constant while a data frame is transferred by an automatic shadow mechanism. The registers can be programmed all the time with new settings that are taken into account for the next data frame. During a data frame, the applied (shadowed) setting is not changed, although new values have been written after the start of the data frame. The setting is internally “frozen” with the start of each data frame.

Although this shadow mechanism being implemented, it is recommended to change the baud rate and shift control setting only while the IIS protocol is switched off.

17.6.2.3 Mode Control Behavior

In IIS mode, the following kernel modes are supported:

- Run Mode 0/1:
Behavior as programmed, no impact on data transfers.
- Stop Mode 0/1:
Bit PCR.WAGEN is internally considered as 0 (the bit itself is not changed). If WAGEN = 1, the WA generation is stopped, but PSR.END is not set. The complete data frame is finished before entering stop mode, including a possible delay due to PCR.TDEL.
When leaving a stop mode with WAGEN = 1, the WA generation starts from the beginning.

17.6.2.4 Transfer Delay

The transfer delay can be used to synchronize a data transfer to an event (e.g. a change of the WA signal). This event has to be synchronously generated to the falling edge of the shift clock SCK (like the change of the transmit data), because the input signal for the event is directly sampled in the receiver (as a result, the transmitter can use the detection information with its next edge).

Event signals that are asynchronous to the shift clock while the shift clock is running must not be used. In the example in [Figure 17-60](#), the event (change of signal WA) is generated by the transfer master and as a result, is synchronous to the shift clock SCK. With the rising edge of SCK, signal WA is sampled and checked for a change. If a change is detected, a transfer delay counter TDC is automatically loaded with its programmable reload value (PCR.TDEL), otherwise it is decremented with each rising edge of SCK until it reaches 0, where it stops. The transfer itself is started if the value of TDC has become 0. This can happen under two conditions:

Universal Serial Interface Channel (USIC)

- TDC is reloaded with a PCR.TDEL = 0 when the event is detected
- TDC has reached 0 while counting down

The transfer delay counter is internal to the IIS protocol pre-processor and can not be observed by software. The transfer delay in SCK cycles is given by PCR.TDEL+1.

In the example in [Figure 17-62](#), the reload value PCR.TDEL for TDC is 0. When the samples taken on receiver side show the change of the WA signal, the counter TDC is reloaded. If the reload value is 0, the data transfer starts with 1 shift clock cycle delay compared to the change of WA.

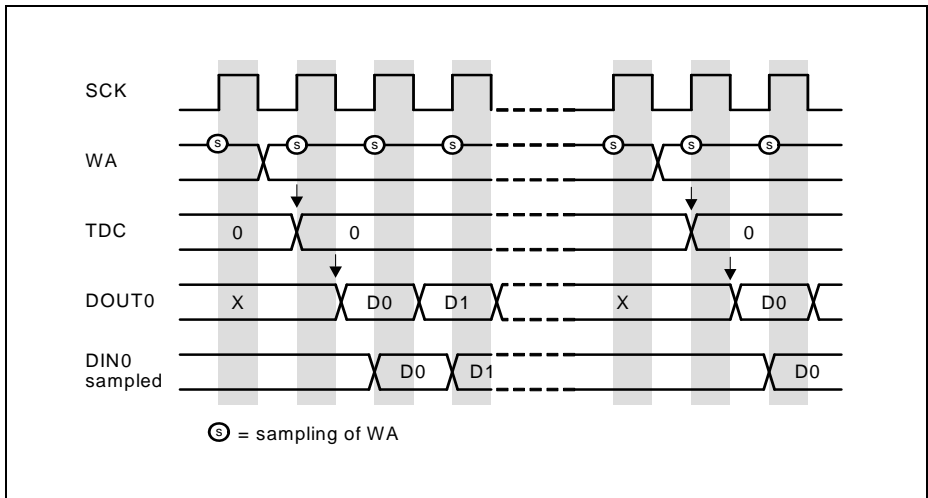


Figure 17-62 Transfer Delay with Delay 1

The ideal case without any transfer delay is shown in [Figure 17-63](#). The WA signal changes and the data output value become valid at the same time. This implies that the transmitter “knows” in advance that the event signal will change with the next rising edge of TCLK. This is achieved by delaying the data transmission after the previously detected WA change the system word length minus 1.

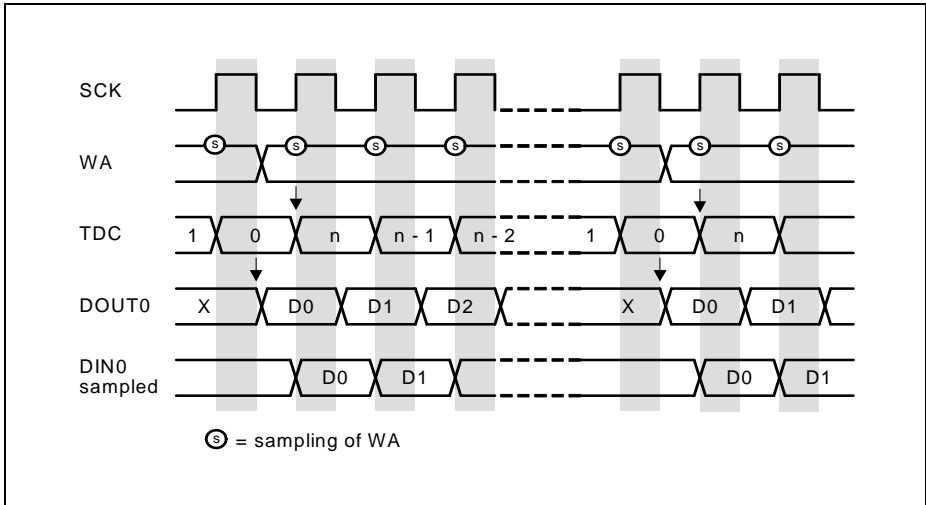


Figure 17-63 No Transfer Delay

If the end of the transfer delay is detected simultaneously to change of WA, the transfer is started and the delay counter is reloaded with PCR.TDEL. This allows to run the USIC as IIS device without any delay. In this case, internally the delay from the previous event elapses just at the moment when a new event occurs. If PCR.TDEL is set to a value bigger than the system word length, no transfer takes place.

17.6.2.5 Parity Mode

Parity generation is not supported in IIS mode and bit field CCR.PM = 00_B has to be programmed.

17.6.2.6 Transfer Mode

In IIS mode, bit field SCTR.TRM = 11_B has to be programmed to allow data transfers. Setting SCTR.TRM = 00_B disables and stops the data transfer immediately.

17.6.2.7 Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to IIS frame handling.

- Transmit buffer interrupt TBI:
Bit PSR.TBIF is set after the start of first data bit of a data word.
- Transmit shift interrupt TSI:
Bit PSR.TSIF is set after the start of the last data bit of a data word.

Universal Serial Interface Channel (USIC)

- Receiver start interrupt RS1:
Bit PSR.RSIF is set after the reception of the first data bit of a data word.
With this event, bit TCSR.TDV is cleared and new data can be loaded to the transmit buffer.
- Receiver interrupt RI and alternative interrupt AI:
Bit PSR.RIF is set at after the reception of the last data bit of a data word with WA = 0.
Bit RBUFSR.SOF indicates whether the received data word has been the first data word of a new data frame.
Bit PSR.AIF is set at after the reception of the last data bit of a data word with WA = 1.
Bit RBUFSR.SOF indicates whether the received data word has been the first data word of a new data frame.

17.6.2.8 Baud Rate Generator Interrupt Handling

The baud rate generator interrupt indicate that the capture mode timer has reached its maximum value. With this event, the bit PSR.BRGIF is set.

17.6.2.9 Protocol-Related Argument and Error

In order to distinguish between data words received for the left or the right channel, the IIS protocol pre-processor samples the level of the WA input (just after the WA transition) and propagates it as protocol-related error (although it is not an error, but an indication) to the receive buffer status register at the bit position RBUFSR[9]. This bit position defines if either a standard receive interrupt (if RBUFSR[9] = 0) or an alternative receive interrupt (if RBUFSR[9] = 1) becomes activated when a new data word has been received. Incoming data can be handled by different interrupts or DMA mechanisms for the left and the right channel if the corresponding events are directed to different interrupt nodes. Flag PAR is always 0.

17.6.2.10 Transmit Data Handling

The IIS protocol pre-processor allows to distinguish between the left and the right channel for data transmission. Therefore, bit TCSR.WA indicates on which channel the data in the buffer will be transmitted. If TCSR.WA = 0, the data will be transmitted after a falling edge of WA. If TCSR.WA = 1, the data will be transmitted after a rising edge of WA. The WA value sampled after the WA transition is considered to distinguish between both channels (referring to PSR.WA).

Bit TCSR.WA can be automatically updated by the transmit control information TC[4] for each data word if TCSR.WAMD = 1. In this case, data written to TBUF[15:0] (or IN[15:0] if a FIFO data buffer is used) is considered as left channel data, whereas data written to TBUF[31:16] (or IN[31:16] if a FIFO data buffer is used) is considered as right channel data.

17.6.2.11 Receive Buffer Handling

If a receive FIFO buffer is available ($CCFG.RB = 1$) and enabled for data handling ($RBCTR.SIZE > 0$), it is recommended to set $RBCTR.RCIM = 11_B$ in IIS mode. This leads to an indication that the data word has been the first data word of a new data frame if bit $OUTR.RCI[0] = 1$, and the channel indication by the sampled WA value is given by $OUTR.RCI[4]$.

The standard receive buffer event and the alternative receive buffer event can be used for the following operation in RCI mode ($RBCTR.RNM = 1$):

- A standard receive buffer event indicates that a data word can be read from OUTR that belongs to a data frame started when $WA = 0$.
- An alternative receive buffer event indicates that a data word can be read from OUTR that belongs to a data frame started when $WA = 1$.

17.6.2.12 Loop-Delay Compensation

The synchronous signaling mechanism of the IIS protocol being similar to the one of the SSC protocol, the closed-loop delay has to be taken into account for the application setup. In IIS mode, loop-delay compensation in master mode is also possible to achieve higher baud rates.

Please refer to the more detailed description in the SSC chapter.

17.6.3 Operating the IIS in Master Mode

In order to operate the IIS in master mode, the following issues have to be considered:

- Select IIS mode:
It is recommended to configure all parameters of the IIS that do not change during run time while $CCR.MODE = 0000_B$. Bit field $SCTR.TRM = 11_B$ has to be programmed. The configuration of the input stages has to be done while $CCR.MODE = 0000_B$ to avoid unintended edges of the input signals and the IIS mode can be enabled by $CCR.MODE = 0011_B$ afterwards.
- Pin connection for data transfer:
Establish a connection of input stage DX0 with the selected receive data input pin ($DIN0$) with $DX0CR.INSW = 1$. Configure a transmit data output pin ($DOUT0$) for a transmitter.
The data shift unit allowing full-duplex data transfers based on the same WA signal, the values delivered by the DX0 stage are considered as data bits (receive function can not be disabled independently from the transmitter). To receive IIS data, the transmitter does not necessarily need to be configured (no assignment of DOUT0 signal to a pin).
- Baud rate generation:
The desired baud rate setting has to be selected, comprising the fractional divider

Universal Serial Interface Channel (USIC)

and the baud rate generator. Bit DX1CR.INSW = 0 has to be programmed to use the baud rate generator output SCLK directly as input for the data shift unit.

- **Word address WA generation:**
The WA generation has to be enabled by setting PCR.WAGEN = 1 and the programming of the number of shift clock cycles between the changes of WA. Bit DX2CR.INSW = 0 has to be programmed to use the WA generator as input for the data shift unit. Configure WA output pin for signal SELOx if needed.
- **Data format configuration:**
The word length, the frame length, and the shift direction have to be set up according to the application requirements by programming the register SCTR. Generally, the MSB is shifted first (SCTR.SDIR = 1).
Bit TCSR.WAMD can be set to use the transmit control information TCI[4] to distinguish the data words for transmission while WA = 0 or while WA = 1.

Note: The step to enable the alternate output port functions should only be done after the IIS mode is enabled, to avoid unintended spikes on the output.

17.6.3.1 Baud Rate Generation

The baud rate is defined by the frequency of the SCLK signal (one period of f_{SCLK} represents one data bit).

If the fractional divider mode is used to generate f_{PIN} , there can be an uncertainty of one period of f_{PERIPH} for f_{PIN} . This uncertainty does not accumulate over several SCLK cycles. As a consequence, the average frequency is reached, whereas the duty cycle of 50% of the SCLK and MCLK signals can vary by one period of f_{PERIPH} .

In IIS applications, where the phase relation between the optional MCLK output signal and SCLK is not relevant, SCLK can be based on the frequency f_{PIN} (BRG.PPPEN = 0). In the case that a fixed phase relation between the MCLK signal and SCLK is required (e.g. when using MCLK as clock reference for external devices), the additional divider by 2 stage has to be taken into account (BRG.PPPEN = 1). This division is due to the fact that signal MCLK toggles with each cycle of f_{PIN} . Signal SCLK is then based on signal MCLK, see [Figure 17-64](#).

The adjustable integer divider factor is defined by bit field BRG.PDIV.

$$\begin{aligned}
 f_{SCLK} &= \frac{f_{PIN}}{2} \times \frac{1}{PDIV + 1} && \text{if PPPEN} = 0 \\
 f_{SCLK} &= \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} && \text{if PPPEN} = 1
 \end{aligned}
 \tag{17.12}$$

Note: In the IIS protocol, the master (unit generating the shift clock and the WA signal) changes the status of its data and WA output line with the falling edge of SCK. The slave transmitter also has to transmit on falling edges. The sampling of the received data is done with the rising edges of SCLK. The input stage DX1 and the

Universal Serial Interface Channel (USIC)

SCLKOUT have to be programmed to invert the shift clock signal to fit to the internal signals.

17.6.3.2 WA Generation

The word address (or word select) line WA regularly toggles after N cycles of signal SCLK. The time between the changes of WA is called system word length and can be programmed by using the following bit fields.

In IIS master mode, the system word length is defined by:

- BRG.CTQSEL = 10_B
to base the WA toggling on SCLK
- BRG.PCTQ
to define the number N of SCLK cycles per system word length
- BRG.DCTQ
to define the number N of SCLK cycles per system word length

$$N = (PCTQ + 1) \times (DCTQ + 1) \tag{17.13}$$

17.6.3.3 Master Clock Output

The master clock signal MCLK can be generated by the master of the IIS transfer (BRG.PPPEN = 1). It is used especially to connect external Codec devices. It can be configured by bit BRG.MCLKCFG in its polarity to become the output signal MCLKOUT.

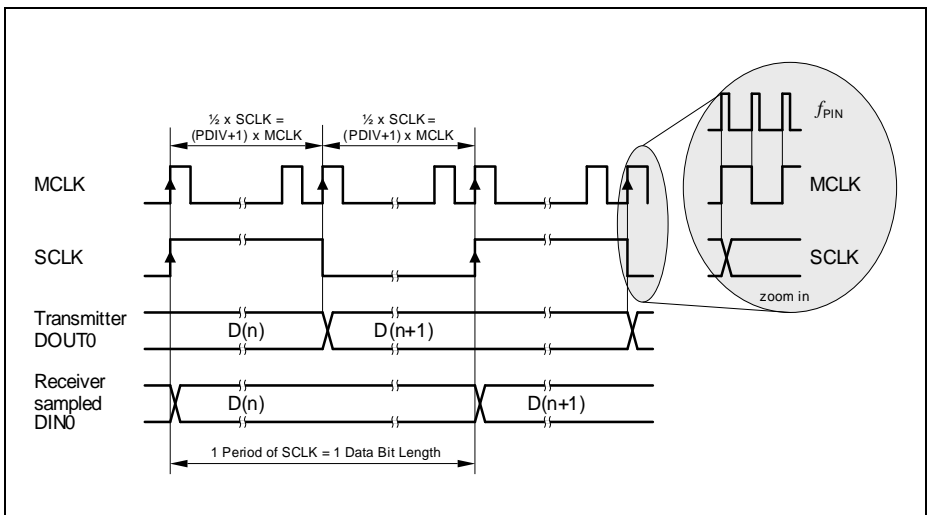


Figure 17-64 MCLK and SCLK for IIS

17.6.3.4 Protocol Interrupt Events

The following protocol-related events are generated in IIS mode and can lead to a protocol interrupt.

Please note that the bits in register PSR are not all automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- **WA rising/falling edge events:**
The WA generation block indicates two events that are monitored in register PSR. Flag PSR.WAFE is set with the falling edge, flag PSR.WARE with the rising edge of the WA signal. A protocol interrupt can be generated if PCR.WAFEIEN = 1 for the falling edge, similar for PCR.WAREIEN = 1 for a rising edge.
- **WA end event:**
The WA generation block also indicates when it has stopped the WA generation after it has been disabled by writing PCR.WAGEN = 0. A protocol interrupt can be generated if PCR.ENDIEN = 1.
- **DX2T event:**
An activation of the trigger signal DX2T is indicated by PSR.DX2TEV = 1 and can generate a protocol interrupt if PCR.DX2TIEN = 1. This event can be evaluated instead of the WA rising/falling events if a delay compensation like in SSC mode (for details, refer to corresponding SSC section) is used.

17.6.4 Operating the IIS in Slave Mode

In order to operate the IIS in slave mode, the following issues have to be considered:

- **Select IIS mode:**
It is recommended to configure all parameters of the IIS that do not change during run time while CCR.MODE = 0000_B. Bit field SCTR.TRM = 11_B has to be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the IIS mode can be enabled by CCR.MODE = 0011_B afterwards.
- **Pin connection for data transfer:**
Establish a connection of input stage DX0 with the selected receive data input pin (DIN0) with DX0CR.INSW = 1. Configure a transmit data output pin (DOUT0) for a transmitter.
The data shift unit allowing full-duplex data transfers based on the same WA signal, the values delivered by the DX0 stage are considered as data bits (receive function can not be disabled independently from the transmitter). To receive IIS data, the transmitter does not necessarily need to be configured (no assignment of DOUT0 signal to a pin).
Note that the step to enable the alternate output port functions should only be done after the IIS mode is enabled, to avoid unintended spikes on the output.

Universal Serial Interface Channel (USIC)

- Pin connection for shift clock:
Establish a connection of input stage DX1 with the selected shift clock input pin (SCLKIN) with DX1CR.INSW = 1 and with inverted polarity (DX1CR.DPOL = 1).
- Pin connection for WA input:
Establish a connection of input stage DX2 with the WA input pin (SELIN) with DX2CR.INSW = 1.
- Baud rate generation:
The baud rate generator is not needed and can be switched off by the fractional divider.
- WA generation:
The WA generation is not needed and can be switched off (PCR.WAGEN = 0).

17.6.4.1 Protocol Events and Interrupts

The following protocol-related event is generated in IIS mode and can lead to a protocol interrupt.

Please note that the bits in register PSR are not all automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- WA rising/falling/end events:
The WA generation being switched off, these events are not available.
- DX2T event:
An activation of the trigger signal DX2T is indicated by PSR.DX2TEV = 1 and can generate a protocol interrupt if PCR.DX2TIEN = 1.

17.6.5 IIS Protocol Registers

In IIS mode, the registers PCR and PSR handle IIS related information.

17.6.5.1 IIS Protocol Control Registers

In IIS mode, the PCR register bits or bit fields are defined as described in this section.

Universal Serial Interface Channel (USIC)

PCR

Protocol Control Register [IIS Mode]

(3C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCLK		0						TDEL							
rw		rw						rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DX2 TIEN		0						ENDI EN	WAR EIEN	WAF EIEN	0	SELINV	DTE N	WAGEN	
rw		rw						rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
WAGEN	0	rw	<p>WA Generation Enable</p> <p>This bit enables/disables the generation of word address control output signal WA.</p> <p>0_B The IIS can be used as slave. The generation of the word address signal is disabled. The output signal WA is 0. The MCLKO signal generation depends on PCR.MCLK.</p> <p>1_B The IIS can be used as master. The generation of the word address signal is enabled. The signal starts with a 0 after being enabled. The generation of MCLK is enabled, independent of PCR.MCLK. After clearing WAGEN, the USIC module stops the generation of the WA signal within the next 4 WA periods.</p>
DTE N	1	rw	<p>Data Transfers Enable</p> <p>This bit enables/disables the transfer of IIS frames as a reaction to changes of the input word address control line WA.</p> <p>0_B The changes of the WA input signal are ignored and no transfers take place.</p> <p>1_B Transfers are enabled.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
SELINV	2	rw	<p>Select Inversion</p> <p>This bit defines if the polarity of the SELOx outputs in relation to the internally generated word address signal WA.</p> <p>0_B The SELOx outputs have the same polarity as the WA signal.</p> <p>1_B The SELOx outputs have the inverted polarity to the WA signal.</p>
WAFEIEN	4	rw	<p>WA Falling Edge Interrupt Enable</p> <p>This bit enables/disables the activation of a protocol interrupt when a falling edge of WA has been generated.</p> <p>0_B A protocol interrupt is not activated if a falling edge of WA is generated.</p> <p>1_B A protocol interrupt is activated if a falling edge of WA is generated.</p>
WAREIEN	5	rw	<p>WA Rising Edge Interrupt Enable</p> <p>This bit enables/disables the activation of a protocol interrupt when a rising edge of WA has been generated.</p> <p>0_B A protocol interrupt is not activated if a rising edge of WA is generated.</p> <p>1_B A protocol interrupt is activated if a rising edge of WA is generated.</p>
ENDIEN	6	rw	<p>END Interrupt Enable</p> <p>This bit enables/disables the activation of a protocol interrupt when the WA generation stops after clearing PCR.WAGEN (complete system word length is processed before stopping).</p> <p>0_B A protocol interrupt is not activated.</p> <p>1_B A protocol interrupt is activated.</p>
DX2TIEN	15	rw	<p>DX2T Interrupt Enable</p> <p>This bit enables/disables the generation of a protocol interrupt if the DX2T signal becomes activated (indicated by PSR.DX2TEV = 1).</p> <p>0_B A protocol interrupt is not generated if DX2T is active.</p> <p>1_B A protocol interrupt is generated if DX2T is active.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
TDEL	[21:16]	rw	Transfer Delay This bit field defines the transfer delay when an event is detected. If bit field TDEL = 0, the additional delay functionality is switched off and a delay of one shift clock cycle is introduced.
MCLK	31	rw	Master Clock Enable This bit enables generation of the master clock MCLK (not directly used for IIC protocol, can be used as general frequency output). 0 _B The MCLK generation is disabled and MCLK is 0. 1 _B The MCLK generation is enabled.
0	3, [14:7], [30:22]	rw	Reserved Returns 0 if read; should be written with 0;

17.6.5.2 IIS Protocol Status Register

The following PSR status bits or bit fields are available in IIS mode. Please note that the bits in register PSR are not cleared by hardware.

The flags in the PSR register can be cleared by writing a 1 to the corresponding bit position in register PSCR. Writing a 1 to a bit position in PSR sets the corresponding flag, but does not lead to further actions (no interrupt generation). Writing a 0 has no effect. These flags should be cleared by software before enabling a new protocol.

PSR

Protocol Status Register [IIS Mode] (48_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
															BRG IF
															0
															r
															rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	RIF	TBIF	TSIF	DLIF	RSIF	0		END	WAR E	WAF E	DX2 TEV	0	DX2 S	WA	
rwh	rwh	rwh	rwh	rwh	rwh	r		rwh	rwh	rwh	rwh	r	rwh	rwh	

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
WA	0	rwh	<p>Word Address</p> <p>This bit indicates the status of the WA input signal, sampled after a transition of WA has been detected. This information is forwarded to the corresponding bit position RBUFSR[9] to distinguish between data received for the right and the left channel.</p> <p>0_B WA has been sampled 0. 1_B WA has been sampled 1.</p>
DX2S	1	rwh	<p>DX2S Status</p> <p>This bit indicates the current status of the DX2S signal, which is used as word address signal WA.</p> <p>0_B DX2S is 0. 1_B DX2S is 1.</p>
DX2TEV	3	rwh	<p>DX2T Event Detected¹⁾</p> <p>This bit indicates that the DX2T signal has been activated. In IIS slave mode, an activation of DX2T generates a protocol interrupt if PCR.DX2TIEN = 1.</p> <p>0_B The DX2T signal has not been activated. 1_B The DX2T signal has been activated.</p>
WAFE	4	rwh	<p>WA Falling Edge Event¹⁾</p> <p>This bit indicates that a falling edge of the WA output signal has been generated. This event generates a protocol interrupt if PCR.WAFEIEN = 1.</p> <p>0_B A WA falling edge has not been generated. 1_B A WA falling edge has been generated.</p>
WARE	5	rwh	<p>WA Rising Edge Event¹⁾</p> <p>This bit indicates that a rising edge of the WA output signal has been generated. This event generates a protocol interrupt if PCR.WAREIEN = 1.</p> <p>0_B A WA rising edge has not been generated. 1_B A WA rising edge has been generated.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
END	6	rwh	<p>WA Generation End¹⁾ This bit indicates that the WA generation has ended after clearing PCR.WAGEN. This bit should be cleared by software before clearing WAGEN.</p> <p>0_B The WA generation has not yet ended (if it is running and WAGEN has been cleared). 1_B The WA generation has ended (if it has been running).</p>
RSIF	10	rwh	<p>Receiver Start Indication Flag 0_B A receiver start event has not occurred. 1_B A receiver start event has occurred.</p>
DLIF	11	rwh	<p>Data Lost Indication Flag 0_B A data lost event has not occurred. 1_B A data lost event has occurred.</p>
TSIF	12	rwh	<p>Transmit Shift Indication Flag 0_B A transmit shift event has not occurred. 1_B A transmit shift event has occurred.</p>
TBIF	13	rwh	<p>Transmit Buffer Indication Flag 0_B A transmit buffer event has not occurred. 1_B A transmit buffer event has occurred.</p>
RIF	14	rwh	<p>Receive Indication Flag 0_B A receive event has not occurred. 1_B A receive event has occurred.</p>
AIF	15	rwh	<p>Alternative Receive Indication Flag 0_B An alternative receive event has not occurred. 1_B An alternative receive event has occurred.</p>
BRGIF	16	rwh	<p>Baud Rate Generator Indication Flag 0_B A baud rate generator event has not occurred. 1_B A baud rate generator event has occurred.</p>
0	2, [9:7], [31:17]	r	<p>Reserved Returns 0 if read; not modified in IIS mode.</p>

1) This status bit can generate a protocol interrupt (see [Page 17-22](#)). The general interrupt status flags are described in the general interrupt chapter.

17.7 Service Request Generation

The USIC module provides 6 service request outputs SR[5:0] to be shared between two channels. The service request outputs SR[5:0] are connected to interrupt nodes in the Nested Vectored Interrupt Controller (NVIC). Additionally, the first 2 outputs, SR[1:0], are also connected to the General Purpose DMA (GPDMA) via the DMA Line Router (DLR).

Each USIC communication channel can be connected to up to 6 service request handlers (connected to USICx.SR[5:0], though 3 or 4 are normally used, e.g. one for transmission, one for reception, one or two for protocol or error handling, or for the alternative receive events).

17.8 Debug Behaviour

Each USIC communication channel can be pre-configured to enter one of four kernel modes, when the program execution of the CPU is halted by the debugger.

Refer to [Section 17.2.2.2](#) for details.

17.9 Power, Reset and Clock

The USIC module is located in the core power domain. The module, including all registers other than the bit field KSCFG.SUMCFG, can be reset to its default state by a system reset or a software reset triggered through the setting of corresponding bits in PRSETx registers. The bit field KSCFG.SUMCFG is reset to its default value only by a debug reset.

The USIC module is clocked by the Peripheral Bus clock (f_{PERIPH}). If the module clock is disabled by KSCFG.MODEN = 0, the module cannot be accessed by read or write operations (except register KSCFG that can always be accessed).

17.10 Initialization and System Dependencies

The USIC module is held in reset after a start-up from a system or software reset. Therefore, the application has to apply the following initialization sequence before operating the USIC module:

- Release reset of USIC module by writing a 1 to the USICxRS bit in SCU_PRCLR0 or SCU_PRCLR1 registers
- Enable the module by writing 1s to the MODEN and BPMODEN bits in KSCFG register.

17.11 Registers

[Table 17-20](#) shows all registers which are required for programming a USIC channel, as well as the FIFO buffer. It summarizes the USIC communication channel registers and defines the relative addresses and the reset values.

Universal Serial Interface Channel (USIC)

Please note that all registers can be accessed with any access width (8-bit, 16-bit, 32-bit), independent of the described width.

All USIC registers (except bit field KSCFG.SUMCFG) are always reset by a system reset. Bit field KSCFG.SUMCFG is reset by a debug reset.

Note: The register bits marked “w” always deliver 0 when read. They are used to modify flip-flops in other registers or to trigger internal actions.

Figure 17-65 shows the register types of the USIC module registers and channel registers. In a specific microcontroller, module registers of USIC module “x” are marked by the module prefix “USICx_”. Channel registers of USIC module “x” are marked by the channel prefix “USICx_CH0_” and “USICx_CH1_”.

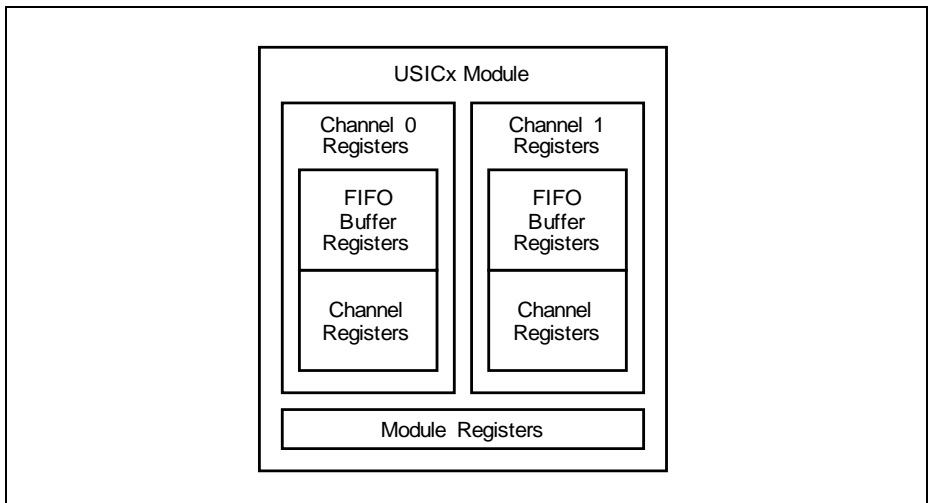


Figure 17-65 USIC Module and Channel Registers

Table 17-20 USIC Kernel-Related and Kernel Registers

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Description see
			Read	Write	
Module Registers¹⁾					
ID	Module Identification Register	008 _H	U, PV	U, PV	Page 17-158
Channel Registers					
–	reserved	000 _H	BE	BE	–
CCFG	Channel Configuration Register	004 _H	U, PV	U, PV	Page 17-163

Universal Serial Interface Channel (USIC)

Table 17-20 USIC Kernel-Related and Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Description see
			Read	Write	
KSCFG	Kernel State Configuration Register	00C _H	U, PV	U, PV	Page 17-164
FDR	Fractional Divider Register	010 _H	U, PV	PV	Page 17-177
BRG	Baud Rate Generator Register	014 _H	U, PV	PV	Page 17-178
INPR	Interrupt Node Pointer Register	018 _H	U, PV	U, PV	Page 17-167
DX0CR	Input Control Register 0	01C _H	U, PV	U, PV	Page 17-172
DX1CR	Input Control Register 1	020 _H	U, PV	U, PV	Page 17-174
DX2CR	Input Control Register 2	024 _H	U, PV	U, PV	Page 17-172
DX3CR	Input Control Register 3	028 _H	U, PV	U, PV	
DX4CR	Input Control Register 4	02C _H	U, PV	U, PV	
DX5CR	Input Control Register 5	030 _H	U, PV	U, PV	
SCTR	Shift Control Register	034 _H	U, PV	U, PV	
TCSR	Transmit Control/Status Register	038 _H	U, PV	U, PV	Page 17-185
PCR	Protocol Control Register	03C _H	U, PV	U, PV	Page 17-168 ²⁾
			U, PV	U, PV	Page 17-66 ³⁾
			U, PV	U, PV	Page 17-98 ⁴⁾
			U, PV	U, PV	Page 17-129 ⁵⁾
			U, PV	U, PV	Page 17-148 ⁶⁾
CCR	Channel Control Register	040 _H	U, PV	PV	Page 17-159
CMTR	Capture Mode Timer Register	044 _H	U, PV	U, PV	Page 17-181

Universal Serial Interface Channel (USIC)

Table 17-20 USIC Kernel-Related and Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Description see
			Read	Write	
PSR	Protocol Status Register	048 _H	U, PV	U, PV	Page 17-169 ²⁾
			U, PV	U, PV	Page 17-70 ³⁾
			U, PV	U, PV	Page 17-102 ⁴⁾
			U, PV	U, PV	Page 17-132 ⁵⁾
			U, PV	U, PV	Page 17-150 ⁶⁾
PSCR	Protocol Status Clear Register	04C _H	U, PV	U, PV	Page 17-170
RBUFSR	Receiver Buffer Status Register	050 _H	U, PV	U, PV	Page 17-203
RBUF	Receiver Buffer Register	054 _H	U, PV	U, PV	Page 17-201
RBUFD	Receiver Buffer Register for Debugger	058 _H	U, PV	U, PV	Page 17-202
RBUF0	Receiver Buffer Register 0	05C _H	U, PV	U, PV	Page 17-194
RBUF1	Receiver Buffer Register 1	060 _H	U, PV	U, PV	Page 17-195
RBUF01SR	Receiver Buffer 01 Status Register	064 _H	U, PV	U, PV	Page 17-196
FMR	Flag Modification Register	068 _H	U, PV	U, PV	Page 17-192
–	reserved; do not access this location	06C _H	U, PV	BE	–
–	reserved	070 _H - 07C _H	BE	BE	–
TBUFx	Transmit Buffer Input Location x (x = 00-31)	080 _H + x*4	U, PV	U, PV	Page 17-194

FIFO Buffer Registers

BYP	Bypass Data Register	100 _H	U, PV	U, PV	Page 17-204
BYPCTR	Bypass Control Register	104 _H	U, PV	U, PV	Page 17-205
TBCTR	Transmit Buffer Control Register	108 _H	U, PV	U, PV	Page 17-213
RBCTR	Receive Buffer Control Register	10C _H	U, PV	U, PV	Page 17-217
TRBPTR	Transmit/Receive Buffer Pointer Register	110 _H	U, PV	U, PV	Page 17-225

Universal Serial Interface Channel (USIC)

Table 17-20 USIC Kernel-Related and Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Description see
			Read	Write	
TRBSR	Transmit/Receive Buffer Status Register	114 _H	U, PV	U, PV	Page 17-208
TRBSCR	Transmit/Receive Buffer Status Clear Register	118 _H	U, PV	U, PV	Page 17-212
OUTR	Receive Buffer Output Register	11C _H	U, PV	U, PV	Page 17-223
OUTDR	Receive Buffer Output Register for Debugger	120 _H	U, PV	U, PV	Page 17-224
–	reserved	124 _H - 17C _H	BE	BE	–
INx	Transmit FIFO Buffer Input Location x (x = 00-31)	180 _H + x*4	U, PV	U, PV	Page 17-222

- 1) Details of the module identification registers are described in the implementation section (see [Page 17-158](#)).
- 2) This page shows the general register layout.
- 3) This page shows the register layout in ASC mode.
- 4) This page shows the register layout in SSC mode.
- 5) This page shows the register layout in IIC mode.
- 6) This page shows the register layout in IIS mode.

17.11.1 Address Map

The registers of the USIC communication channel are available at the following base addresses. The exact register address is given by the relative address of the register (given in [Table 17-20](#)) plus the channel base address (given in [Table 17-21](#)).

Table 17-21 Registers Address Space

Module	Base Address	End Address	Note
USIC0_CH0	40030000 _H	400301FF _H	–
USIC0_CH1	40030200 _H	400303FF _H	–
USIC1_CH0	48020000 _H	480201FF _H	–
USIC1_CH1	48020200 _H	480203FF _H	–

Table 17-22 FIFO and Reserved Address Space

Module	Base Address	End Address	Note
USIC0	40030400 _H	400307FF _H	USIC0 RAM area, shared between USIC0_CH0 and USIC0_CH1
reserved	40030800 _H	40033FFF _H	This address range is reserved
USIC1	48020400 _H	480207FF _H	USIC1 RAM area, shared between USIC1_CH0 and USIC1_CH1
reserved	48020800 _H	48027FFF _H	This address range is reserved

17.11.2 Module Identification Registers

The module identification registers indicate the function and the design step of the USIC modules.

USIC0_ID

Module Identification Register

(4003 0008_H)

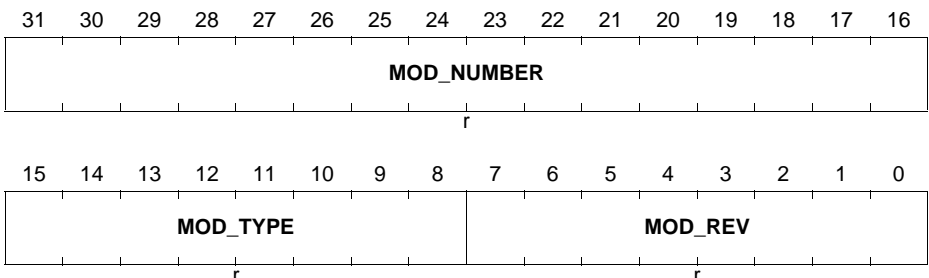
Reset Value: 00AA C0XX_H

USIC1_ID

Module Identification Register

(4802 0008_H)

Reset Value: 00AA C0XX_H



Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
MOD_REV	[7:0]	r	Module Revision Number MOD_REV defines the revision number. The value of a module revision starts with 01 _H (first revision).
MOD_TYPE	[15:8]	r	Module Type This bit field is C0 _H . It defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	Module Number Value This bit field defines the USIC module identification number (00AA _H = USIC).

17.11.3 Channel Control and Configuration Registers

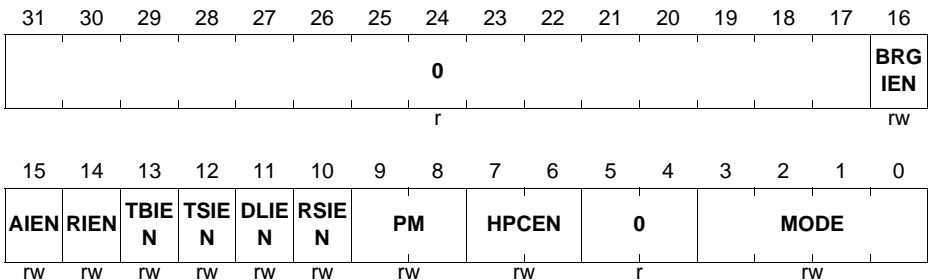
17.11.3.1 Channel Control Register

The channel control register contains the enable/disable bits for hardware port control and interrupt generation on channel events, the control of the parity generation and the protocol selection of a USIC channel.

FDR can be written only with a privilege mode access.

CCR

Channel Control Register (40_H) **Reset Value: 0000 0000_H**



Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
MODE	[3:0]	rw	<p>Operating Mode</p> <p>This bit field selects the protocol for this USIC channel. Selecting a protocol that is not available (see register CCFG) or a reserved combination disables the USIC channel. When switching between two protocols, the USIC channel has to be disabled before selecting a new protocol. In this case, registers PCR and PSR have to be cleared or updated by software.</p> <p>0_H The USIC channel is disabled. All protocol-related state machines are set to an idle state.</p> <p>1_H The SSC (SPI) protocol is selected.</p> <p>2_H The ASC (SCI, UART) protocol is selected.</p> <p>3_H The IIS protocol is selected.</p> <p>4_H The IIC protocol is selected.</p> <p>Other bit combinations are reserved.</p>
HPCEN	[7:6]	rw	<p>Hardware Port Control Enable</p> <p>This bit enables the hardware port control for the specified set of DX[3:0] and DOUT[3:0] pins.</p> <p>00_B The hardware port control is disabled.</p> <p>01_B The hardware port control is enabled for DX0 and DOUT0.</p> <p>10_B The hardware port control is enabled for DX3, DX0 and DOUT[1:0].</p> <p>11_B The hardware port control is enabled for DX0, DX[5:3] and DOUT[3:0].</p> <p><i>Note: The hardware port control feature is useful only for SSC protocols in half-duplex configurations, such as dual- and quad-SSC. For all other protocols HPCEN must always be written with 00_B.</i></p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
PM	[9:8]	rw	<p>Parity Mode</p> <p>This bit field defines the parity generation of the sampled input values.</p> <p>00_B The parity generation is disabled.</p> <p>01_B Reserved</p> <p>10_B Even parity is selected (parity bit = 1 on odd number of 1s in data, parity bit = 0 on even number of 1s in data).</p> <p>11_B Odd parity is selected (parity bit = 0 on odd number of 1s in data, parity bit = 1 on even number of 1s in data).</p>
RSIEN	10	rw	<p>Receiver Start Interrupt Enable</p> <p>This bit enables the interrupt generation in case of a receiver start event.</p> <p>0_B The receiver start interrupt is disabled.</p> <p>1_B The receiver start interrupt is enabled. In case of a receiver start event, the service request output SRx indicated by INPR.TBINP is activated.</p>
DLIEN	11	rw	<p>Data Lost Interrupt Enable</p> <p>This bit enables the interrupt generation in case of a data lost event (data received in RBUFx while RDVx = 1).</p> <p>0_B The data lost interrupt is disabled.</p> <p>1_B The data lost interrupt is enabled. In case of a data lost event, the service request output SRx indicated by INPR.PINP is activated.</p>
TSIEN	12	rw	<p>Transmit Shift Interrupt Enable</p> <p>This bit enables the interrupt generation in case of a transmit shift event.</p> <p>0_B The transmit shift interrupt is disabled.</p> <p>1_B The transmit shift interrupt is enabled. In case of a transmit shift interrupt event, the service request output SRx indicated by INPR.TSINP is activated.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
TBIEN	13	rw	<p>Transmit Buffer Interrupt Enable</p> <p>This bit enables the interrupt generation in case of a transmit buffer event.</p> <p>0_B The transmit buffer interrupt is disabled.</p> <p>1_B The transmit buffer interrupt is enabled. In case of a transmit buffer event, the service request output SRx indicated by INPR.TBINP is activated.</p>
RIEN	14	rw	<p>Receive Interrupt Enable</p> <p>This bit enables the interrupt generation in case of a receive event.</p> <p>0_B The receive interrupt is disabled.</p> <p>1_B The receive interrupt is enabled. In case of a receive event, the service request output SRx indicated by INPR.RINP is activated.</p>
AIEN	15	rw	<p>Alternative Receive Interrupt Enable</p> <p>This bit enables the interrupt generation in case of an alternative receive event.</p> <p>0_B The alternative receive interrupt is disabled.</p> <p>1_B The alternative receive interrupt is enabled. In case of an alternative receive event, the service request output SRx indicated by INPR.AINP is activated.</p>
BRGIEN	16	rw	<p>Baud Rate Generator Interrupt Enable</p> <p>This bit enables the interrupt generation in case of a baud rate generator event.</p> <p>0_B The baud rate generator interrupt is disabled.</p> <p>1_B The baud rate generator interrupt is enabled. In case of a baud rate generator event, the service request output SRx indicated by INPR.PINP is activated.</p>
0	[5:4], [31:17]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Universal Serial Interface Channel (USIC)

17.11.3.2 Channel Configuration Register

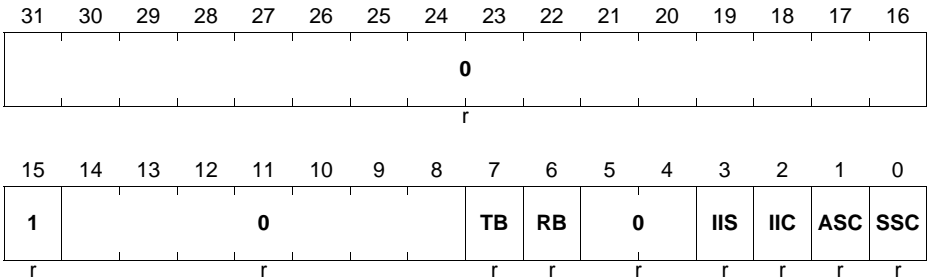
The channel configuration register contains indicates the functionality that is available in the USIC channel.

CCFG

Channel Configuration Register

(04_H)

Reset Value: 0000 80CF_H



Field	Bits	Type	Description
SSC	0	r	SSC Protocol Available This bit indicates if the SSC protocol is available. 0 _B The SSC protocol is not available. 1 _B The SSC protocol is available.
ASC	1	r	ASC Protocol Available This bit indicates if the ASC protocol is available. 0 _B The ASC protocol is not available. 1 _B The ASC protocol is available.
IIC	2	r	IIC Protocol Available This bit indicates if the IIC functionality is available. 0 _B The IIC protocol is not available. 1 _B The IIC protocol is available.
IIS	3	r	IIS Protocol Available This bit indicates if the IIS protocol is available. 0 _B The IIS protocol is not available. 1 _B The IIS protocol is available.
RB	6	r	Receive FIFO Buffer Available This bit indicates if an additional receive FIFO buffer is available. 0 _B A receive FIFO buffer is not available. 1 _B A receive FIFO buffer is available.

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
TB	7	r	Transmit FIFO Buffer Available This bit indicates if an additional transmit FIFO buffer is available. 0 _B A transmit FIFO buffer is not available. 1 _B A transmit FIFO buffer is available.
1	15	r	Reserved Read as 1; should be written with 1.
0	[5:4], [14:8], [31:16]	r	Reserved Read as 0; should be written with 0.

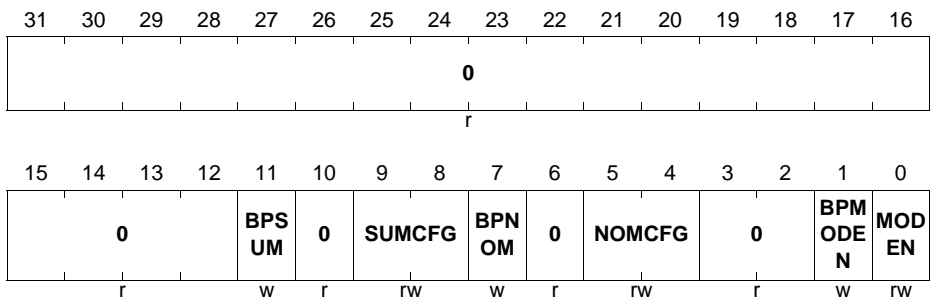
17.11.3.3 Kernel State Configuration Register

The kernel state configuration register KSCCFG allows the selection of the desired kernel modes for the different device operating modes.

KSCCFG

Kernel State Configuration Register (0C_H)

Reset Value: 0000 0000_H



Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
MODEN	0	rw	<p>Module Enable This bit enables the module kernel clock and the module functionality.</p> <p>0_B The module is switched off immediately (without respecting a stop condition). It does not react on mode control actions and the module clock is switched off. The module does not react on read accesses and ignores write accesses (except to KSCFG).</p> <p>1_B The module is switched on and can operate. After writing 1 to MODEN, it is recommended to read register KSCFG to avoid pipeline effects in the control block before accessing other USIC registers.</p>
BPMODEN	1	w	<p>Bit Protection for MODEN This bit enables the write access to the bit MODEN. It always reads 0.</p> <p>0_B MODEN is not changed. 1_B MODEN is updated with the written value.</p>
NOMCFG	[5:4]	rw	<p>Normal Operation Mode Configuration This bit field defines the kernel mode applied in normal operation mode.</p> <p>00_B Run mode 0 is selected. 01_B Run mode 1 is selected. 10_B Stop mode 0 is selected. 11_B Stop mode 1 is selected.</p>
BPNOM	7	w	<p>Bit Protection for NOMCFG This bit enables the write access to the bit field NOMCFG. It always reads 0.</p> <p>0_B NOMCFG is not changed. 1_B NOMCFG is updated with the written value.</p>
SUMCFG	[9:8]	rw	<p>Suspend Mode Configuration This bit field defines the kernel mode applied in suspend mode. Coding like NOMCFG.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
BPSUM	11	w	<p>Bit Protection for SUMCFG</p> <p>This bit enables the write access to the bit field SUMCFG. It always reads 0.</p> <p>0_B SUMCFG is not changed.</p> <p>1_B SUMCFG is updated with the written value.</p>
0	[3:2], 6, 10, [31:12]	r	<p>Reserved</p> <p>Read as 0; should be written with 0. Bit 2 can read as 1 after BootROM exit (but can be ignored).</p>

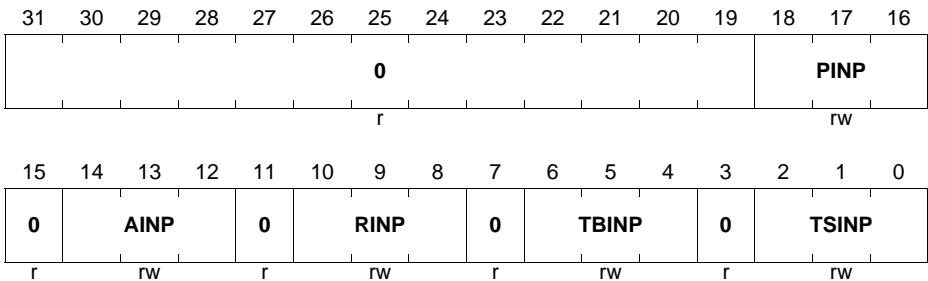
Universal Serial Interface Channel (USIC)

17.11.3.4 Interrupt Node Pointer Register

The interrupt node pointer register defines the service request output SR_x that is activated if the corresponding event occurs and interrupt generation is enabled.

INPR

Interrupt Node Pointer Register (18_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TSINP	[2:0]	rw	<p>Transmit Shift Interrupt Node Pointer</p> <p>This bit field defines which service request output SR_x becomes activated in case of a transmit shift interrupt.</p> <p>000_B Output SR0 becomes activated. 001_B Output SR1 becomes activated. 010_B Output SR2 becomes activated. 011_B Output SR3 becomes activated. 100_B Output SR4 becomes activated. 101_B Output SR5 becomes activated.</p> <p><i>Note: All other settings of the bit field are reserved.</i></p>
TBINP	[6:4]	rw	<p>Transmit Buffer Interrupt Node Pointer</p> <p>This bit field defines which service request output SR_x will be activated in case of a transmit buffer interrupt or a receive start interrupt. Coding like TSINP.</p>
RINP	[10:8]	rw	<p>Receive Interrupt Node Pointer</p> <p>This bit field defines which service request output SR_x will be activated in case of a receive interrupt. Coding like TSINP.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
AINP	[14:12]	rw	Alternative Receive Interrupt Node Pointer This bit field defines which service request output SRx will be activated in case of a alternative receive interrupt. Coding like TSINP.
PINP	[18:16]	rw	Protocol Interrupt Node Pointer This bit field defines which service request output SRx becomes activated in case of a protocol interrupt. Coding like TSINP.
0	3, 7, 11, 15, [31:19]	r	Reserved Read as 0; should be written with 0.

17.11.4 Protocol Related Registers

17.11.4.1 Protocol Control Registers

The bits in the protocol control register define protocol-specific functions. They have to be configured by software before enabling a new protocol. Only the bits used for the selected protocol are taken into account, whereas the other bit positions always read as 0. The protocol-specific meaning is described in the related protocol section.

PCR

Protocol Control Register (3C_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTR 31	CTR 30	CTR 29	CTR 28	CTR 27	CTR 26	CTR 25	CTR 24	CTR 23	CTR 22	CTR 21	CTR 20	CTR 19	CTR 18	CTR 17	CTR 16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR 15	CTR 14	CTR 13	CTR 12	CTR 11	CTR 10	CTR 9	CTR 8	CTR 7	CTR 6	CTR 5	CTR 4	CTR 3	CTR 2	CTR 1	CTR 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CTR_x (x = 0-31)	x	rw	Protocol Control Bit x This bit is a protocol control bit.

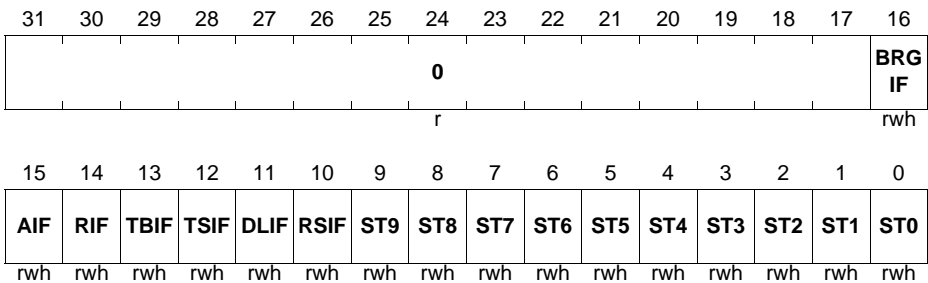
Universal Serial Interface Channel (USIC)

17.11.4.2 Protocol Status Register

The flags in the protocol status register can be cleared by writing a 1 to the corresponding bit position in register PSCR. Writing a 1 to a bit position in PSR sets the corresponding flag, but does not lead to further actions (no interrupt generation). Writing a 0 has no effect. These flags should be cleared by software before enabling a new protocol. The protocol-specific meaning is described in the related protocol section.

PSR

Protocol Status Register (48_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
STx (x = 0-9)	x	rwh	Protocol Status Flag x See protocol specific description.
RSIF	10	rwh	Receiver Start Indication Flag 0 _B A receiver start event has not occurred. 1 _B A receiver start event has occurred.
DLIF	11	rwh	Data Lost Indication Flag 0 _B A data lost event has not occurred. 1 _B A data lost event has occurred.
TSIF	12	rwh	Transmit Shift Indication Flag 0 _B A transmit shift event has not occurred. 1 _B A transmit shift event has occurred.
TBIF	13	rwh	Transmit Buffer Indication Flag 0 _B A transmit buffer event has not occurred. 1 _B A transmit buffer event has occurred.
RIF	14	rwh	Receive Indication Flag 0 _B A receive event has not occurred. 1 _B A receive event has occurred.

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
AIF	15	rwh	Alternative Receive Indication Flag 0 _B An alternative receive event has not occurred. 1 _B An alternative receive event has occurred.
BRGIF	16	rwh	Baud Rate Generator Indication Flag 0 _B A baud rate generator event has not occurred. 1 _B A baud rate generator event has occurred.
0	[31:17]	r	Reserved; read as 0; should be written with 0;

17.11.4.3 Protocol Status Clear Register

Read accesses to this register always deliver 0 at all bit positions.

PSCR

Protocol Status Clear Register (4C_H) **Reset Value: 0000 0000_H**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAIF	CRIF	CTBI	CTSI	CDLI	CRSI	CST	CST	CST	CST	CST	CST	CST	CST	CST	CST
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
		F	F	F	F	9	8	7	6	5	4	3	2	1	0

Field	Bits	Type	Description
CSTx (x = 0-9)	x	w	Clear Status Flag x in PSR 0 _B No action 1 _B Flag PSR.STx is cleared.
CRSIF	10	w	Clear Receiver Start Indication Flag 0 _B No action 1 _B Flag PSR.RSIF is cleared.
CDLIF	11	w	Clear Data Lost Indication Flag 0 _B No action 1 _B Flag PSR.DLIF is cleared.

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
CTSIF	12	w	Clear Transmit Shift Indication Flag 0 _B No action 1 _B Flag PSR.TSIF is cleared.
CTBIF	13	w	Clear Transmit Buffer Indication Flag 0 _B No action 1 _B Flag PSR.TBIF is cleared.
CRIF	14	w	Clear Receive Indication Flag 0 _B No action 1 _B Flag PSR.RIF is cleared.
CAIF	15	w	Clear Alternative Receive Indication Flag 0 _B No action 1 _B Flag PSR.AIF is cleared.
CBRGIF	16	w	Clear Baud Rate Generator Indication Flag 0 _B No action 1 _B Flag PSR.BRGIF is cleared.
0	[31:17]	r	Reserved; read as 0; should be written with 0;

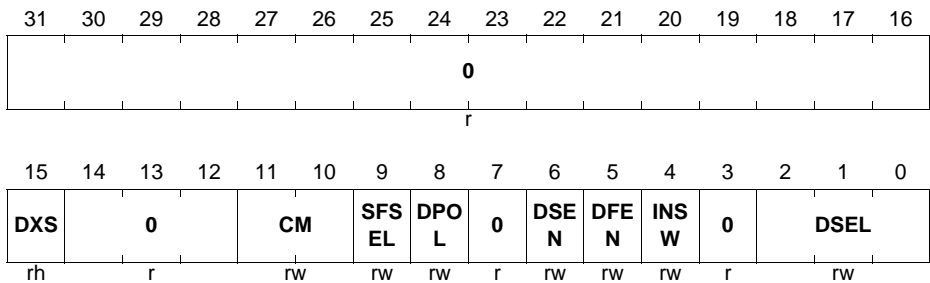
17.11.5 Input Stage Register

17.11.5.1 Input Control Registers

The input control registers contain the bits to define the characteristics of the input stages (input stage DX0 is controlled by register DX0CR, etc.).

Universal Serial Interface Channel (USIC)

DX0CR		
Input Control Register 0	(1C_H)	Reset Value: 0000 0000_H
DX2CR		
Input Control Register 2	(24_H)	Reset Value: 0000 0000_H
DX3CR		
Input Control Register 3	(28_H)	Reset Value: 0000 0000_H
DX4CR		
Input Control Register 4	(2C_H)	Reset Value: 0000 0000_H
DX5CR		
Input Control Register 5	(30_H)	Reset Value: 0000 0000_H



Field	Bits	Type	Description
DSEL	[2:0]	rw	<p>Data Selection for Input Signal</p> <p>This bit field defines the input data signal for the corresponding input line for protocol pre-processor. The selection can be made from the input vector DXn[G:A].</p> <p>000_B The data input DXnA is selected. 001_B The data input DXnB is selected. 010_B The data input DXnC is selected. 011_B The data input DXnD is selected. 100_B The data input DXnE is selected. 101_B The data input DXnF is selected. 110_B The data input DXnG is selected. 111_B The data input is always 1.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
INSW	4	rw	<p>Input Switch</p> <p>This bit defines if the data shift unit input is derived from the input data path DXn or from the selected protocol pre-processors.</p> <p>0_B The input of the data shift unit is controlled by the protocol pre-processor.</p> <p>1_B The input of the data shift unit is connected to the selected data input line. This setting is used if the signals are directly derived from an input pin without treatment by the protocol pre-processor.</p>
DFEN	5	rw	<p>Digital Filter Enable</p> <p>This bit enables/disables the digital filter for signal DXnS.</p> <p>0_B The input signal is not digitally filtered.</p> <p>1_B The input signal is digitally filtered.</p>
DSEN	6	rw	<p>Data Synchronization Enable</p> <p>This bit selects if the asynchronous input signal or the synchronized (and optionally filtered) signal DXnS can be used as input for the data shift unit.</p> <p>0_B The un-synchronized signal can be taken as input for the data shift unit.</p> <p>1_B The synchronized signal can be taken as input for the data shift unit.</p>
DPOL	8	rw	<p>Data Polarity for DXn</p> <p>This bit defines the signal polarity of the input signal.</p> <p>0_B The input signal is not inverted.</p> <p>1_B The input signal is inverted.</p>
SFSEL	9	rw	<p>Sampling Frequency Selection</p> <p>This bit defines the sampling frequency of the digital filter for the synchronized signal DXnS.</p> <p>0_B The sampling frequency is f_{PERIPH}.</p> <p>1_B The sampling frequency is f_{FD}.</p>

Universal Serial Interface Channel (USIC)

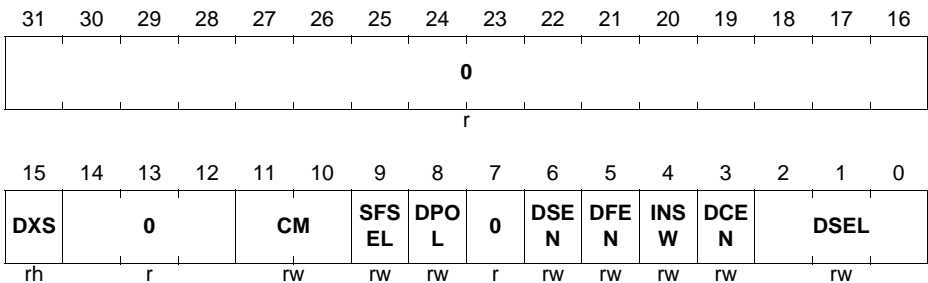
Field	Bits	Type	Description
CM	[11:10]	rw	Combination Mode This bit field selects which edge of the synchronized (and optionally filtered) signal DXnS activates the trigger output DXnT of the input stage. 00 _B The trigger activation is disabled. 01 _B A rising edge activates DXnT. 10 _B A falling edge activates DXnT. 11 _B Both edges activate DXnT.
DXS	15	rh	Synchronized Data Value This bit indicates the value of the synchronized (and optionally filtered) input signal. 0 _B The current value of DXnS is 0. 1 _B The current value of DXnS is 1.
0	3, 7, [14:12] , [31:16]	r	Reserved Read as 0; should be written with 0.

DX1CR

Input Control Register 1

(20_H)

Reset Value: 0000 0000_H



Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
DSEL	[2:0]	rw	<p>Data Selection for Input Signal This bit field defines the input data signal for the corresponding input line for protocol pre-processor. The selection can be made from the input vector DX1[G:A].</p> <p>000_B The data input DX1A is selected. 001_B The data input DX1B is selected. 010_B The data input DX1C is selected. 011_B The data input DX1D is selected. 100_B The data input DX1E is selected. 101_B The data input DX1F is selected. 110_B The data input DX1G is selected. 111_B The data input is always 1.</p>
DCEN	3	rw	<p>Delay Compensation Enable This bit selects if the receive shift clock is controlled by INSW or derived from the input data path DX1.</p> <p>0_B The receive shift clock is dependent on INSW selection. 1_B The receive shift clock is connected to the selected data input line. This setting is used if delay compensation is required in SSC and IIS protocols, else DCEN should always be 0.</p>
INSW	4	rw	<p>Input Switch This bit defines if the data shift unit input is derived from the input data path DX1 or from the selected protocol pre-processors.</p> <p>0_B The input of the data shift unit is controlled by the protocol pre-processor. 1_B The input of the data shift unit is connected to the selected data input line. This setting is used if the signals are directly derived from an input pin without treatment by the protocol pre-processor.</p>
DFEN	5	rw	<p>Digital Filter Enable This bit enables/disables the digital filter for signal DX1S.</p> <p>0_B The input signal is not digitally filtered. 1_B The input signal is digitally filtered.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
DSEN	6	rw	<p>Data Synchronization Enable</p> <p>This bit selects if the asynchronous input signal or the synchronized (and optionally filtered) signal DX1S can be used as input for the data shift unit.</p> <p>0_B The un-synchronized signal can be taken as input for the data shift unit.</p> <p>1_B The synchronized signal can be taken as input for the data shift unit.</p>
DPOL	8	rw	<p>Data Polarity for DXn</p> <p>This bit defines the signal polarity of the input signal.</p> <p>0_B The input signal is not inverted.</p> <p>1_B The input signal is inverted.</p>
SFSEL	9	rw	<p>Sampling Frequency Selection</p> <p>This bit defines the sampling frequency of the digital filter for the synchronized signal DX1S.</p> <p>0_B The sampling frequency is f_{PERIPH}.</p> <p>1_B The sampling frequency is f_{FD}.</p>
CM	[11:10]	rw	<p>Combination Mode</p> <p>This bit field selects which edge of the synchronized (and optionally filtered) signal DX1S activates the trigger output DX1T of the input stage.</p> <p>00_B The trigger activation is disabled.</p> <p>01_B A rising edge activates DX1T.</p> <p>10_B A falling edge activates DX1T.</p> <p>11_B Both edges activate DX1T.</p>
DXS	15	rh	<p>Synchronized Data Value</p> <p>This bit indicates the value of the synchronized (and optionally filtered) input signal.</p> <p>0_B The current value of DX1S is 0.</p> <p>1_B The current value of DX1S is 1.</p>
0	7, [14:12], , [31:16]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

17.11.6 Baud Rate Generator Registers

17.11.6.1 Fractional Divider Register

The fractional divider register FDR allows the generation of the internal frequency f_{FD} , that is derived from the system clock f_{PERIPH} .

FDR can be written only with a privilege mode access.

FDR

Fractional Divider Register

(10_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		0				RESULT									
rw		r				rh									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DM		0				STEP									
rw		r				rw									

Field	Bits	Type	Description
STEP	[9:0]	rw	Step Value In normal divider mode STEP contains the reload value for RESULT after RESULT has reached 3FF _H . In fractional divider mode STEP defines the value added to RESULT with each input clock cycle.
DM	[15:14]	rw	Divider Mode This bit fields defines the functionality of the fractional divider block. 00 _B The divider is switched off, $f_{FD} = 0$. 01 _B Normal divider mode selected. 10 _B Fractional divider mode selected. 11 _B The divider is switched off, $f_{FD} = 0$.

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
RESULT	[25:16]	rh	Result Value In normal divider mode this bit field is updated with f_{PERIPH} according to: $\text{RESULT} = \text{RESULT} + 1$ In fractional divider mode this bit field is updated with f_{PERIPH} according to: $\text{RESULT} = \text{RESULT} + \text{STEP}$ If bit field DM is written with 01_{B} or 10_{B} , RESULT is loaded with a start value of 3FF_{H} .
0	[31:30]	rw	Reserved for Future Use Must be written with 0 to allow correct fractional divider operation.
0	[13:10], [29:26]	r	Reserved Read as 0; should be written with 0.

17.11.6.2 Baud Rate Generator Register

The protocol-related counters for baud rate generation and timing measurement are controlled by the register BRG.

FDR can be written only with a privilege mode access.

BRG

Baud Rate Generator Register (14_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCLKCFG		MCL KCF G	SCL KOS EL	0		PDIV									
rw		rw	rw	r		rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		DCTQ				PCTQ		CTQSEL		0	PPP EN	TME N	0	CLKSEL	
r		rw				rw		rw		r	rw	rw	r	rw	

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
CLKSEL	[1:0]	rw	<p>Clock Selection</p> <p>This bit field defines the input frequency f_{PIN}</p> <p>00_B The fractional divider frequency f_{FD} is selected.</p> <p>01_B Reserved, no action</p> <p>10_B The trigger signal DX1T defines f_{PIN}. Signal MCLK toggles with f_{PIN}.</p> <p>11_B Signal MCLK corresponds to the DX1S signal and the frequency f_{PIN} is derived from the rising edges of DX1S.</p>
TMEN	3	rw	<p>Timing Measurement Enable</p> <p>This bit enables the timing measurement of the capture mode timer.</p> <p>0_B Timing measurement is disabled: The trigger signals DX0T and DX1T are ignored.</p> <p>1_B Timing measurement is enabled: The 10-bit counter is incremented by 1 with f_{PPP} and stops counting when reaching its maximum value. If one of the trigger signals DX0T or DX1T become active, the counter value is captured into bit field CTV, the counter is cleared and a transmit shift event is generated.</p>
PPPEN	4	rw	<p>Enable 2:1 Divider for f_{PPP}</p> <p>This bit defines the input frequency f_{PPP}.</p> <p>0_B The 2:1 divider for f_{PPP} is disabled. $f_{PPP} = f_{PIN}$</p> <p>1_B The 2:1 divider for f_{PPP} is enabled. $f_{PPP} = f_{MCLK} = f_{PIN} / 2$.</p>
CTQSEL	[7:6]	rw	<p>Input Selection for CTQ</p> <p>This bit defines the length of a time quantum for the protocol pre-processor.</p> <p>00_B $f_{CTQIN} = f_{PDIV}$</p> <p>01_B $f_{CTQIN} = f_{PPP}$</p> <p>10_B $f_{CTQIN} = f_{SCLK}$</p> <p>11_B $f_{CTQIN} = f_{MCLK}$</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
PCTQ	[9:8]	rw	Pre-Divider for Time Quanta Counter This bit field defines length of a time quantum t_q for the time quanta counter in the protocol pre-processor. $t_q = (PCTQ + 1) / f_{CTQIN}$
DCTQ	[14:10]	rw	Denominator for Time Quanta Counter This bit field defines the number of time quanta t_q taken into account by the time quanta counter in the protocol pre-processor.
PDIV	[25:16]	rw	Divider Mode: Divider Factor to Generate f_{PDIV} This bit field defines the ratio between the input frequency f_{PPP} and the divider frequency f_{PDIV} .
SCLKOSEL	28	rw	Shift Clock Output Select This bit field selects the input source for the SCLKOUT signal. 0_B SCLK from the baud rate generator is selected as the SCLKOUT input source. 1_B The transmit shift clock from DX1 input stage is selected as the SCLKOUT input source. <i>Note: The setting SCLKOSEL = 1 is used only when complete closed loop delay compensation is required for a slave SSC/IIS. The default setting of SCLKOSEL = 0 should be always used for all other cases.</i>
MCLKCFG	29	rw	Master Clock Configuration This bit field defines the level of the passive phase of the MCLKOUT signal. 0_B The passive level is 0. 1_B The passive level is 1.
SCLKCFG	[31:30]	rw	Shift Clock Output Configuration This bit field defines the level of the passive phase of the SCLKOUT signal and enables/disables a delay of half of a SCLK period. 00_B The passive level is 0 and the delay is disabled. 01_B The passive level is 1 and the delay is disabled. 10_B The passive level is 0 and the delay is enabled. 11_B The passive level is 1 and the delay is enabled.
0	2, 5, 15, [27:26]	r	Reserved Read as 0; should be written with 0.

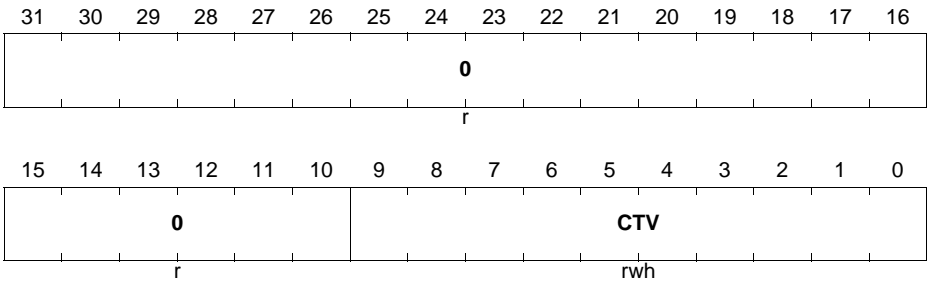
Universal Serial Interface Channel (USIC)

17.11.6.3 Capture Mode Timer Register

The captured timer value is provided by the register CMTR.

CMTR

Capture Mode Timer Register (44_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
CTV	[9:0]	rwh	Captured Timer Value The value of the counter is captured into this bit field if one of the trigger signals DX0T or DX1T are activated by the corresponding input stage.
0	[31:10]	r	Reserved Read as 0; should be written with 0.

17.11.7 Transfer Control and Status Registers

17.11.7.1 Shift Control Register

The data shift unit is controlled by the register SCTR. The values in this register are applied for data transmission and reception.

Please note that the shift control settings SDIR, WLE, FLE, DSM and HPCDIR are shared between transmitter and receiver. They are internally “frozen” for a each data word transfer in the transmitter with the first transmit shift clock edge and with the first receive shift clock edge in the receiver. The software has to take care that updates of these bit fields by software are done coherently (e.g. refer to the receiver start event indication PSR.RSIF).

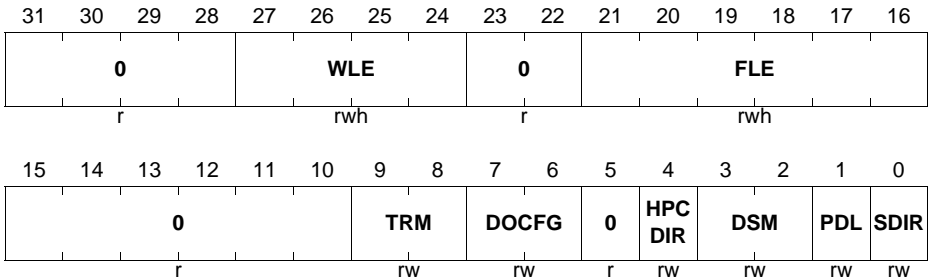
Universal Serial Interface Channel (USIC)

SCTR

Shift Control Register

(34_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SDIR	0	rw	<p>Shift Direction</p> <p>This bit defines the shift direction of the data words for transmission and reception.</p> <p>0_B Shift LSB first. The first data bit of a data word is located at bit position 0.</p> <p>1_B Shift MSB first. The first data bit of a data word is located at the bit position given by bit field SCTR.WLE.</p>
PDL	1	rw	<p>Passive Data Level</p> <p>This bit defines the output level at the shift data output signal when no data is available for transmission. The PDL level is output with the first relevant transmit shift clock edge of a data word.</p> <p>0_B The passive data level is 0.</p> <p>1_B The passive data level is 1.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
DSM	[3:2]	rw	<p>Data Shift Mode</p> <p>This bit field describes how the receive and transmit data is shifted in and out.</p> <p>00_B Receive and transmit data is shifted in and out one bit at a time through DX0 and DOUT0.</p> <p>01_B Reserved.</p> <p>10_B Receive and transmit data is shifted in and out two bits at a time through two input stages (DX0 and DX3) and DOUT[1:0] respectively.</p> <p>11_B Receive and transmit data is shifted in and out four bits at a time through four input stages (DX0, DX[5:3]) and DOUT[3:0] respectively.</p> <p><i>Note: Dual- and Quad-output modes are used only by the SSC protocol. For all other protocols DSM must always be written with 00_B.</i></p>
HPCDIR	4	rw	<p>Port Control Direction</p> <p>This bit defines the direction of the port pin(s) which allows hardware pin control (CCR.PCEN = 1).</p> <p>0_B The pin(s) with hardware pin control enabled are selected to be in input mode.</p> <p>1_B The pin(s) with hardware pin control enabled are selected to be in output mode.</p>
DOCFG	[7:6]	rw	<p>Data Output Configuration</p> <p>This bit defines the relation between the internal shift data value and the data output signal DOUTx.</p> <p>X0_B DOUTx = shift data value</p> <p>X1_B DOUTx = inverted shift data value</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
TRM	[9:8]	rw	<p>Transmission Mode</p> <p>This bit field describes how the shift control signal is interpreted by the DSU. Data transfers are only possible while the shift control signal is active.</p> <p>00_B The shift control signal is considered as inactive and data frame transfers are not possible.</p> <p>01_B The shift control signal is considered active if it is at 1-level. This is the setting to be programmed to allow data transfers.</p> <p>10_B The shift control signal is considered active if it is at 0-level. It is recommended to avoid this setting and to use the inversion in the DX2 stage in case of a low-active signal.</p> <p>11_B The shift control signal is considered active without referring to the actual signal level. Data frame transfer is possible after each edge of the signal.</p>
FLE	[21:16]	rwh	<p>Frame Length</p> <p>This bit field defines how many bits are transferred within a data frame. A data frame can consist of several concatenated data words.</p> <p>If TCSR.FLEMD = 1, the value can be updated automatically by the data handler.</p>
WLE	[27:24]	rwh	<p>Word Length</p> <p>This bit field defines the data word length (amount of bits that are transferred in each data word) for reception and transmission. The data word is always right-aligned in the data buffer at the bit positions [WLE down to 0].</p> <p>If TCSR.WLEMD = 1, the value can be updated automatically by the data handler.</p> <p>0_H The data word contains 1 data bit located at bit position 0.</p> <p>1_H The data word contains 2 data bits located at bit positions [1:0].</p> <p>...</p> <p>E_H The data word contains 15 data bits located at bit positions [14:0].</p> <p>F_H The data word contains 16 data bits located at bit positions [15:0].</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
0	5, [15:10], [23:22], [31:28]	r	Reserved Read as 0; should be written with 0.

17.11.7.2 Transmission Control and Status Register

The data transmission is controlled and monitored by register TCSR.

TCSR

Transmit Control/Status Register (38_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			TE	TVC	TV	0	TSO F				0				
r			rh	rh	rh	r	rh				r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	WA	TDV TR		TDEN		0	TDS SM	TDV	EOF	SOE	HPC MD	WA MD	FLE MD	SEL MD	WLE MD
r	rwh	rw		rw		r	rw	rh	rwh	rwh	rw	rw	rw	rw	rw

Field	Bits	Type	Description
WLEMD	0	rw	WLE Mode This bit enables the data handler to automatically update the bit field SCTR.WLE by the transmit control information TCI[3:0] and bit TCSR.EOF by TCI[4] (see Page 17-33). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUF _x or by an optional data buffer. 0 _B The automatic update of SCTR.WLE and TCSR.EOF is disabled. 1 _B The automatic update of SCTR.WLE and TCSR.EOF is enabled.

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
SELMD	1	rw	<p>Select Mode</p> <p>This bit can be used mainly for the SSC protocol. It enables the data handler to automatically update bit field PCR.CTR[20:16] by the transmit control information TCI[4:0] and clear bit field PCR.CTR[23:21] (see Page 17-33). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer.</p> <p>0_B The automatic update of PCR.CTR[23:16] is disabled.</p> <p>1_B The automatic update of PCR.CTR[23:16] is disabled.</p>
FLEMD	2	rw	<p>FLE Mode</p> <p>This bit enables the data handler to automatically update bits SCTR.FLE[4:0] by the transmit control information TCI[4:0] and to clear bit SCTR.FLE[5] (see Page 17-33). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer.</p> <p>0_B The automatic update of FLE is disabled.</p> <p>1_B The automatic update of FLE is enabled.</p>
WAMD	3	rw	<p>WA Mode</p> <p>This bit can be used mainly for the IIS protocol. It enables the data handler to automatically update bit TCSR.WA by the transmit control information TCI[4] (see Page 17-33). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer.</p> <p>0_B The automatic update of bit WA is disabled.</p> <p>1_B The automatic update of bit WA is enabled.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
HPCMD	4	rw	<p>Hardware Port Control Mode</p> <p>This bit can be used mainly for the dual and quad SSC protocol. It enables the data handler to automatically update bit SCTR.DSM by the transmit control information TCI[1:0] and bit SCTR.HPCDIR by TCI[2] (see Page 17-33). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer.</p> <p>0_B The automatic update of bits SCTR.DSM and SCTR.HPCDIR is disabled.</p> <p>1_B The automatic update of bits SCTR.DSM and SCTR.HPCDIR is enabled.</p>
SOF	5	rwh	<p>Start Of Frame</p> <p>This bit is only taken into account for the SSC protocol, otherwise it is ignored.</p> <p>It indicates that the data word in TBUF is considered as the first word of a new SSC frame if it is valid for transmission (TCSR.TDV = 1). This bit becomes cleared when the TBUF data word is transferred to the transmit shift register.</p> <p>0_B The data word in TBUF is not considered as first word of a frame.</p> <p>1_B The data word in TBUF is considered as first word of a frame. A currently running frame is finished and MSLs becomes deactivated (respecting the programmed delays).</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
EOF	6	rwh	<p>End Of Frame</p> <p>This bit is only taken into account for the SSC protocol, otherwise it is ignored. It can be modified automatically by the data handler if bit WLEMD = 1. It indicates that the data word in TBUF is considered as the last word of an SSC frame. If it is the last word, the MSLS signal becomes inactive after the transfer, respecting the programmed delays. This bit becomes cleared when the TBUF data word is transferred to the transmit shift register.</p> <p>0_B The data word in TBUF is not considered as last word of an SSC frame.</p> <p>1_B The data word in TBUF is considered as last word of an SSC frame.</p>
TDV	7	rh	<p>Transmit Data Valid</p> <p>This bit indicates that the data word in the transmit buffer TBUF can be considered as valid for transmission. The TBUF data word can only be sent out if TDV = 1. It is automatically set when data is moved to TBUF (by writing to one of the transmit buffer input locations TBUFx, or optionally, by the bypass or FIFO mechanism).</p> <p>0_B The data word in TBUF is not valid for transmission.</p> <p>1_B The data word in TBUF is valid for transmission and a transmission start is possible. New data should not be written to a TBUFx input location while TDV = 1.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
TDSSM	8	rw	<p>TBUF Data Single Shot Mode</p> <p>This bit defines if the data word TBUF data is considered as permanently valid or if the data should only be transferred once.</p> <p>0_B The data word in TBUF is not considered as invalid after it has been loaded into the transmit shift register. The loading of the TBUF data into the shift register does not clear TDV.</p> <p>1_B The data word in TBUF is considered as invalid after it has been loaded into the shift register. In ASC and IIC mode, TDV is cleared with the TBI event, whereas in SSC and IIS mode, it is cleared with the RSI event.</p> <p>TDSSM = 1 has to be programmed if an optional data buffer is used.</p>
TDEN	[11:10]	rw	<p>TBUF Data Enable</p> <p>This bit field controls the gating of the transmission start of the data word in the transmit buffer TBUF.</p> <p>00_B A transmission start of the data word in TBUF is disabled. If a transmission is started, the passive data level is sent out.</p> <p>01_B A transmission of the data word in TBUF can be started if TDV = 1.</p> <p>10_B A transmission of the data word in TBUF can be started if TDV = 1 while DX2S = 0.</p> <p>11_B A transmission of the data word in TBUF can be started if TDV = 1 while DX2S = 1.</p>
TDVTR	12	rw	<p>TBUF Data Valid Trigger</p> <p>This bit enables the transfer trigger unit to set bit TCSR.TE if the trigger signal DX2T becomes active for event driven transfer starts, e.g. timer-based or depending on an event at an input pin. Bit TDVTR has to be 0 for protocols where the input stage DX2 is used for data shifting.</p> <p>0_B Bit TCSR.TE is permanently set.</p> <p>1_B Bit TCSR.TE is set if DX2T becomes active while TDV = 1.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
WA	13	rwh	<p>Word Address</p> <p>This bit is only taken into account for the IIS protocol, otherwise it is ignored. It can be modified automatically by the data handler if bit WAMD = 1. Bit WA defines for which channel the data stored in TBUF will be transmitted.</p> <p>0_B The data word in TBUF will be transmitted after a falling edge of WA has been detected (referring to PSR.WA).</p> <p>1_B The data word in TBUF will be transmitted after a rising edge of WA has been detected (referring to PSR.WA).</p>
TSOF	24	rh	<p>Transmitted Start Of Frame</p> <p>This bit indicates if the latest start of a data word transmission has taken place for the first data word of a new data frame. This bit is updated with the transmission start of each data word.</p> <p>0_B The latest data word transmission has not been started for the first word of a data frame.</p> <p>1_B The latest data word transmission has been started for the first word of a data frame.</p>
TV	26	rh	<p>Transmission Valid</p> <p>This bit represents the transmit buffer underflow and indicates if the latest start of a data word transmission has taken place with a valid data word from the transmit buffer TBUF. This bit is updated with the transmission start of each data word.</p> <p>0_B The latest start of a data word transmission has taken place while no valid data was available. As a result, the transmission of a data words with passive level (SCTR.PDL) has been started.</p> <p>1_B The latest start of a data word transmission has taken place with valid data from TBUF.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
TVC	27	rh	<p>Transmission Valid Cumulated</p> <p>This bit cumulates the transmit buffer underflow indication TV. It is cleared automatically together with bit TV and has to be set by writing FMR.ATVC = 1.</p> <p>0_B Since TVC has been set, at least one data buffer underflow condition has occurred.</p> <p>1_B Since TVC has been set, no data buffer underflow condition has occurred.</p>
TE	28	rh	<p>Trigger Event</p> <p>If the transfer trigger mechanism is enabled, this bit indicates that a trigger event has been detected (DX2T = 1) while TCSR.TDV = 1. If the event trigger mechanism is disabled, the bit TE is permanently set. It is cleared by writing FMR.MTDV = 10_B or when the data word located in TBUF is loaded into the shift register.</p> <p>0_B The trigger event has not yet been detected. A transmission of the data word in TBUF can not be started.</p> <p>1_B The trigger event has been detected (or the trigger mechanism is switched off) and a transmission of the data word in TBUF can be started.</p>
0	9, [23:14], 25, [31:29]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

17.11.7.3 Flag Modification Registers

The flag modification register FMR allows the modification of control and status flags related to data handling by using only write accesses. Read accesses to FMR always deliver 0 at all bit positions.

Additionally, the service request outputs of this USIC channel can be activated by software (the activation is triggered by the write access and is deactivated automatically).

Universal Serial Interface Channel (USIC)

FMR

Flag Modification Register

(68_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0										SIO5	SIO4	SIO3	SIO2	SIO1	SIO0
r										w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRD V1	CRD V0	0								ATV C	0		MTDV		
w	w	r								w	r		w		

Field	Bits	Type	Description
MTDV	[1:0]	w	<p>Modify Transmit Data Valid</p> <p>Writing to this bit field can modify bits TCSR.TDV and TCSR.TE to control the start of a data word transmission by software.</p> <p>00_B No action. 01_B Bit TDV is set, TE is unchanged. 10_B Bits TDV and TE are cleared. 11_B Reserved</p>
ATVC	4	w	<p>Activate Bit TVC</p> <p>Writing to this bit can set bit TCSR.TVC to start a new cumulation of the transmit buffer underflow condition.</p> <p>0_B No action. 1_B Bit TCSR.TVC is set.</p>
CRDV0	14	w	<p>Clear Bits RDV for RBUF0</p> <p>Writing 1 to this bit clears bits RBUF01SR.RDV00 and RBUF01SR.RDV10 to declare the received data in RBUF0 as no longer valid (to emulate a read action).</p> <p>0_B No action. 1_B Bits RBUF01SR.RDV00 and RBUF01SR.RDV10 are cleared.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
CRDV1	15	w	<p>Clear Bit RDV for RBUF1</p> <p>Writing 1 to this bit clears bits RBUF01SR.RDV01 and RBUF01SR.RDV11 to declare the received data in RBUF1 as no longer valid (to emulate a read action).</p> <p>0_B No action. 1_B Bits RBUF01SR.RDV01 and RBUF01SR.RDV11 are cleared.</p>
SIO0, SIO1, SIO2, SIO3, SIO4, SIO5	16, 17, 18, 19, 20, 21	w	<p>Set Interrupt Output SRx</p> <p>Writing a 1 to this bit field activates the service request output SRx of this USIC channel. It has no impact on service request outputs of other USIC channels.</p> <p>0_B No action. 1_B The service request output SRx is activated.</p>
0	[3:2], [13:5], [31:22]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

17.11.8 Data Buffer Registers

17.11.8.1 Transmit Buffer Locations

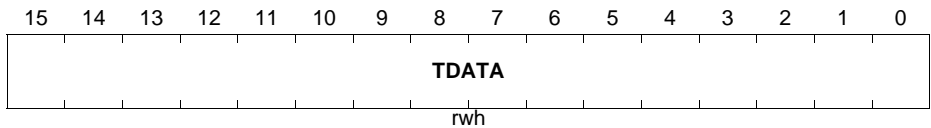
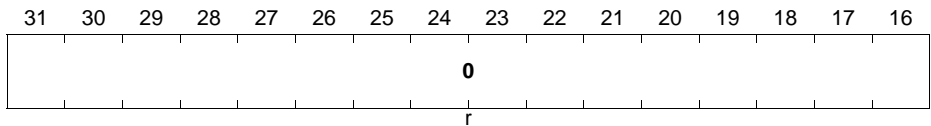
The 32 independent data input locations TBUF00 to TBUF31 are address locations that can be used as data entry locations for the transmit buffer. Data written to one of these locations will appear in a common register TBUF. Additionally, the 5 bit coding of the number [31:0] of the addressed data input location represents the transmit control information TCI (please refer to the protocol sections for more details).

The internal transmit buffer register TBUF contains the data that will be loaded to the transmit shift register for the next transmission of a data word. It can be read out at all TBUF00 to TBUF31 addresses.

Universal Serial Interface Channel (USIC)

TBUF_x (x = 00-31)

Transmit Buffer Input Location x (80_H + x*4) Reset Value: 0000 0000_H



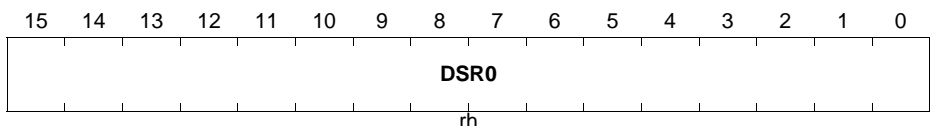
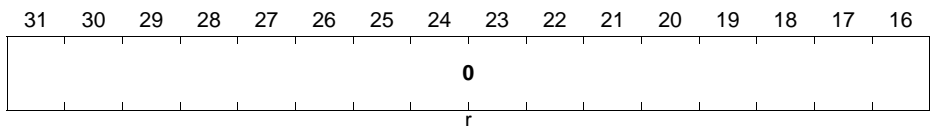
Field	Bits	Type	Description
TDATA	[15:0]	rwh	Transmit Data This bit field contains the data to be transmitted (read view). A data write action to at least the low byte of TDATA sets TCSR.TDV.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

17.11.8.2 Receive Buffer Registers RBUF0, RBUF1

The receive buffer register RBUF0 contains the data received from RSR0[3:0]. A read action does not change the status of the receive data from “not yet read = valid” to “already read = not valid”.

RBUF0

Receiver Buffer Register 0 (5C_H) Reset Value: 0000 0000_H



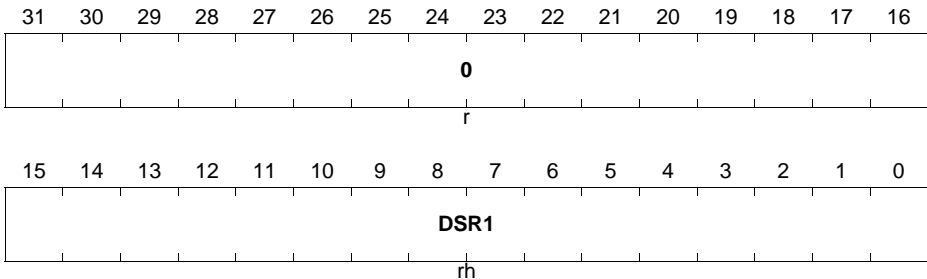
Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
DSR0	[15:0]	rh	Data of Shift Registers 0[3:0]
0	[31:16]	r	Reserved Read as 0; should be written with 0.

The receive buffer register RBUF1 contains the data received from RSR1[3:0]. A read action does not change the status of the receive data from “not yet read = valid” to “already read = not valid”.

RBUF1

Receiver Buffer Register 1 (60_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DSR1	[15:0]	rh	Data of Shift Registers 1[3:0]
0	[31:16]	r	Reserved Read as 0; should be written with 0.

The receive buffer status register RBUF01SR provides the status of the data in receive buffers RBUF0 and RBUF1.

Universal Serial Interface Channel (USIC)

RBUF01SR

Receiver Buffer 01 Status Register (64_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DS1	RDV 11	RDV 10	0			PER R1	PAR 1	0	SOF 1	0	WLEN1				
rh	rh	rh	r			rh	rh	r	rh	r	rh				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DS0	RDV 01	RDV 00	0			PER R0	PAR 0	0	SOF 0	0	WLEN0				
rh	rh	rh	r			rh	rh	r	rh	r	rh				

Field	Bits	Type	Description
WLEN0	[3:0]	rh	<p>Received Data Word Length in RBUF0</p> <p>This bit field indicates how many bits have been received within the last data word stored in RBUF0. This number indicates how many data bits have to be considered as receive data, whereas the other bits in RBUF0 have been cleared automatically. The received bits are always right-aligned.</p> <p>For all protocol modes besides dual and quad SSC, Received data word length = WLEN0 + 1</p> <p>For dual SSC mode, Received data word length = WLEN0 + 2</p> <p>For quad SSC mode, Received data word length = WLEN0 + 4</p>
SOF0	6	rh	<p>Start of Frame in RBUF0</p> <p>This bit indicates whether the data word in RBUF0 has been the first data word of a data frame.</p> <p>0_B The data in RBUF0 has not been the first data word of a data frame.</p> <p>1_B The data in RBUF0 has been the first data word of a data frame.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
PAR0	8	rh	<p>Protocol-Related Argument in RBUF0</p> <p>This bit indicates the value of the protocol-related argument. This value is elaborated depending on the selected protocol and adds additional information to the data word in RBUF0.</p> <p>The meaning of this bit is described in the corresponding protocol chapter.</p>
PERR0	9	rh	<p>Protocol-related Error in RBUF0</p> <p>This bit indicates if the value of the protocol-related argument meets an expected value. This value is elaborated depending on the selected protocol and adds additional information to the data word in RBUF0.</p> <p>The meaning of this bit is described in the corresponding protocol chapter.</p> <p>0_B The received protocol-related argument PAR matches the expected value. The reception of the data word sets bit PSR.RIF and can generate a receive interrupt.</p> <p>1_B The received protocol-related argument PAR does not match the expected value. The reception of the data word sets bit PSR.AIF and can generate an alternative receive interrupt.</p>
RDV00	13	rh	<p>Receive Data Valid in RBUF0</p> <p>This bit indicates the status of the data content of register RBUF0. This bit is identical to bit RBUF01SR.RDV10 and allows consisting reading of information for the receive buffer registers. It is set when a new data word is stored in RBUF0 and automatically cleared if it is read out via RBUF.</p> <p>0_B Register RBUF0 does not contain data that has not yet been read out.</p> <p>1_B Register RBUF0 contains data that has not yet been read out.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
RDV01	14	rh	<p>Receive Data Valid in RBUF1</p> <p>This bit indicates the status of the data content of register RBUF1. This bit is identical to bit RBUF01SR.RDV11 and allows consisting reading of information for the receive buffer registers. It is set when a new data word is stored in RBUF1 and automatically cleared if it is read out via RBUF.</p> <p>0_B Register RBUF1 does not contain data that has not yet been read out.</p> <p>1_B Register RBUF1 contains data that has not yet been read out.</p>
DS0	15	rh	<p>Data Source</p> <p>This bit indicates which receive buffer register (RBUF0 or RBUF1) is currently visible in registers RBUF(D) and in RBUFSR for the associated status information. It indicates which buffer contains the oldest data (the data that has been received first). This bit is identical to bit RBUF01SR.DS1 and allows consisting reading of information for the receive buffer registers.</p> <p>0_B The register RBUF contains the data of RBUF0 (same for associated status information).</p> <p>1_B The register RBUF contains the data of RBUF1 (same for associated status information).</p>
WLEN1	[19:16]	rh	<p>Received Data Word Length in RBUF1</p> <p>This bit field indicates how many bits have been received within the last data word stored in RBUF1. This number indicates how many data bits have to be considered as receive data, whereas the other bits in RBUF1 have been cleared automatically. The received bits are always right-aligned.</p> <p>For all protocol modes besides dual and quad SSC, Received data word length = WLEN1 + 1</p> <p>For dual SSC mode, Received data word length = WLEN1 + 2</p> <p>For quad SSC mode, Received data word length = WLEN1 + 4</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
SOF1	22	rh	<p>Start of Frame in RBUF1</p> <p>This bit indicates whether the data word in RBUF1 has been the first data word of a data frame.</p> <p>0_B The data in RBUF1 has not been the first data word of a data frame.</p> <p>1_B The data in RBUF1 has been the first data word of a data frame.</p>
PAR1	24	rh	<p>Protocol-Related Argument in RBUF1</p> <p>This bit indicates the value of the protocol-related argument. This value is elaborated depending on the selected protocol and adds additional information to the data word in RBUF1.</p> <p>The meaning of this bit is described in the corresponding protocol chapter.</p>
PERR1	25	rh	<p>Protocol-related Error in RBUF1</p> <p>This bit indicates if the value of the protocol-related argument meets an expected value. This value is elaborated depending on the selected protocol and adds additional information to the data word in RBUF1.</p> <p>The meaning of this bit is described in the corresponding protocol chapter.</p> <p>0_B The received protocol-related argument PAR matches the expected value. The reception of the data word sets bit PSR.RIF and can generate a receive interrupt.</p> <p>1_B The received protocol-related argument PAR does not match the expected value. The reception of the data word sets bit PSR.AIF and can generate an alternative receive interrupt.</p>
RDV10	29	rh	<p>Receive Data Valid in RBUF0</p> <p>This bit indicates the status of the data content of register RBUF0. This bit is identical to bit RBUF01SR.RDV00 and allows consistent reading of information for the receive buffer registers.</p> <p>0_B Register RBUF0 does not contain data that has not yet been read out.</p> <p>1_B Register RBUF0 contains data that has not yet been read out.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
RDV11	30	rh	<p>Receive Data Valid in RBUF1</p> <p>This bit indicates the status of the data content of register RBUF1. This bit is identical to bit RBUF01SR.RDV01 and allows consisting reading of information for the receive buffer registers.</p> <p>0_B Register RBUF1 does not contain data that has not yet been read out.</p> <p>1_B Register RBUF1 contains data that has not yet been read out.</p>
DS1	31	rh	<p>Data Source</p> <p>This bit indicates which receive buffer register (RBUF0 or RBUF1) is currently visible in registers RBUF(D) and in RBUFSR for the associated status information. It indicates which buffer contains the oldest data (the data that has been received first). This bit is identical to bit RBUF01SR.DS0 and allows consisting reading of information for the receive buffer registers.</p> <p>0_B The register RBUF contains the data of RBUF0 (same for associated status information).</p> <p>1_B The register RBUF contains the data of RBUF1 (same for associated status information).</p>
0	[5:4], 7, [12:10], [21:20], 23, [28:26]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

17.11.8.3 Receive Buffer Registers RBUF, RBUFD, RBUF SR

The receiver buffer register RBUF shows the content of the either RBUF0 or RBUF1, depending on the order of reception. Always the oldest data (the data word that has been received first) from both receive buffers can be read from RBUF. It is recommended to read out the received data from RBUF instead of RBUF0/1. With a read access of at least the low byte of RBUF, the status of the receive data is automatically changed from “not yet read = valid” to “already read = not valid”, the content of RBUF becomes updated, and the next received data word becomes visible in RBUF.

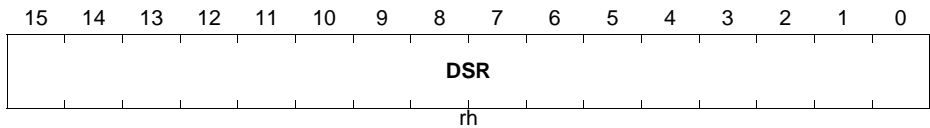
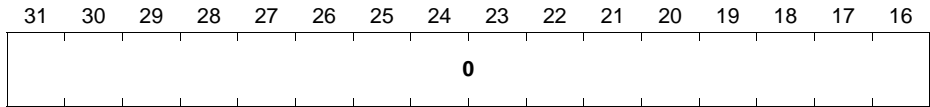
Universal Serial Interface Channel (USIC)

RBUF

Receiver Buffer Register

(54_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DSR	[15:0]	rh	Received Data This bit field monitors the content of either RBUF0 or RBUF1, depending on the reception sequence.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

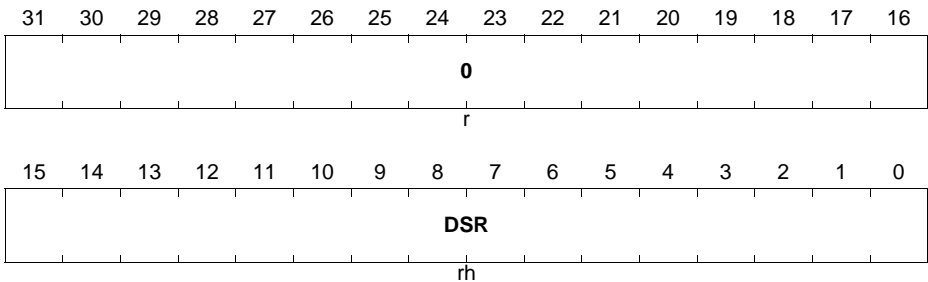
Universal Serial Interface Channel (USIC)

If a debugger should be used to monitor the received data, the automatic update mechanism has to be de-activated to guaranty data consistency. Therefore, the receiver buffer register for debugging RBUFD is available. It is similar to RBUF, but without the automatic update mechanism by a read action. So a debugger (or other monitoring function) can read RBUFD without disturbing the receive sequence.

RBUFD

Receiver Buffer Register for Debugger(58_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DSR	[15:0]	rh	Data from Shift Register Same as RBUF.DSR, but without releasing the buffer after a read action.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Universal Serial Interface Channel (USIC)

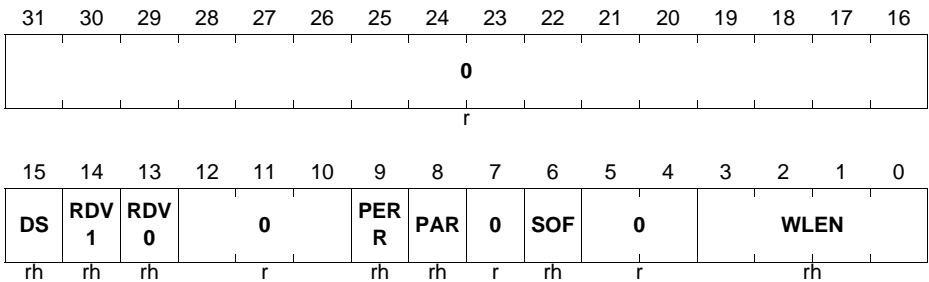
The receive buffer status register RBUF_{SR} provides the status of the data in receive buffers RBUF and RBUFD. If bits RBUF01SR.DS0 (or RBUF01SR.DS1) are 0, the lower 16-bit content of RBUF01SR is monitored in RBUF_{SR}, otherwise the upper 16-bit content of RBUF01SR is shown.

RBUF_{SR}

Receiver Buffer Status Register

(50_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
WLEN	[3:0]	rh	Received Data Word Length in RBUF or RBUFD Description see RBUF01SR.WLEN0 or RBUF01SR.WLEN1.
SOF	6	rh	Start of Frame in RBUF or RBUFD Description see RBUF01SR.SOF0 or RBUF01SR.SOF1.
PAR	8	rh	Protocol-Related Argument in RBUF or RBUFD Description see RBUF01SR.PAR0 or RBUF01SR.PAR1.
PERR	9	rh	Protocol-related Error in RBUF or RBUFD Description see RBUF01SR.PERR0 or RBUF01SR.PERR1.
RDV0	13	rh	Receive Data Valid in RBUF or RBUFD Description see RBUF01SR.RDV00 or RBUF01SR.RDV10.
RDV1	14	rh	Receive Data Valid in RBUF or RBUFD Description see RBUF01SR.RDV01 or RBUF01SR.RDV11.

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
DS	15	rh	Data Source of RBUF or RBUFD Description see RBUF01SR.DS0 or RBUF01SR.DS1.
0	[5:4], 7, [12:10], [31:16]	r	Reserved Read as 0; should be written with 0.

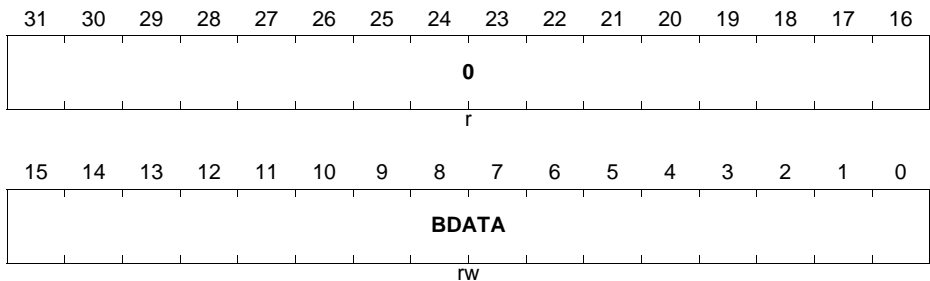
17.11.9 FIFO Buffer and Bypass Registers

17.11.9.1 Bypass Registers

A write action to at least the low byte of the bypass data register sets BYPCR.BDV = 1 (bypass data tagged valid).

BYP

Bypass Data Register (100_H) **Reset Value: 0000 0000_H**



Bit (Field)	Width	Type	Description
BDATA	[15:0]	rw	Bypass Data This bit field contains the bypass data.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

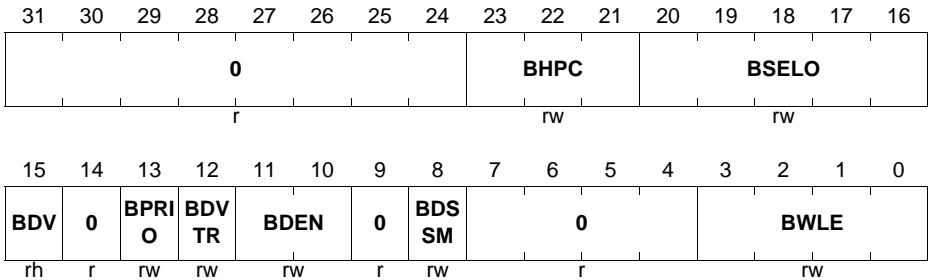
Universal Serial Interface Channel (USIC)

BYPCCR

Bypass Control Register

(104_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
BWLE	[3:0]	rw	<p>Bypass Word Length</p> <p>This bit field defines the word length of the bypass data. The word length is given by BWLE + 1 with the data word being right-aligned in the data buffer at the bit positions [BWLE down to 0].</p> <p>The bypass data word is always considered as an own frame with the length of BWLE.</p> <p>Same coding as SCTR.WLE.</p>
BDSSM	8	rw	<p>Bypass Data Single Shot Mode</p> <p>This bit defines if the bypass data is considered as permanently valid or if the bypass data is only transferred once (single shot mode).</p> <p>0_B The bypass data is still considered as valid after it has been loaded into TBUF. The loading of the data into TBUF does not clear BDV.</p> <p>1_B The bypass data is considered as invalid after it has been loaded into TBUF. The loading of the data into TBUF clears BDV.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
BDEN	[11:10]	rw	<p>Bypass Data Enable</p> <p>This bit field defines if and how the transfer of bypass data to TBUF is enabled.</p> <p>00_B The transfer of bypass data is disabled.</p> <p>01_B The transfer of bypass data to TBUF is possible. Bypass data will be transferred to TBUF according to its priority if BDV = 1.</p> <p>10_B Gated bypass data transfer is enabled. Bypass data will be transferred to TBUF according to its priority if BDV = 1 and while DX2S = 0.</p> <p>11_B Gated bypass data transfer is enabled. Bypass data will be transferred to TBUF according to its priority if BDV = 1 and while DX2S = 1.</p>
BDVTR	12	rw	<p>Bypass Data Valid Trigger</p> <p>This bit enables the bypass data for being tagged valid when DX2T is active (for time framing or time-out purposes).</p> <p>0_B Bit BDV is not influenced by DX2T.</p> <p>1_B Bit BDV is set if DX2T is active.</p>
BPRIO	13	rw	<p>Bypass Priority</p> <p>This bit defines the priority between the bypass data and the transmit FIFO data.</p> <p>0_B The transmit FIFO data has a higher priority than the bypass data.</p> <p>1_B The bypass data has a higher priority than the transmit FIFO data.</p>
BDV	15	rh	<p>Bypass Data Valid</p> <p>This bit defines if the bypass data is valid for a transfer to TBUF. This bit is set automatically by a write access to at least the low-byte of register BYP. It can be cleared by software by writing TRBSCR.CBDV.</p> <p>0_B The bypass data is not valid.</p> <p>1_B The bypass data is valid.</p>
BSELO	[20:16]	rw	<p>Bypass Select Outputs</p> <p>This bit field contains the value that is written to PCR.CTR[20:16] if bypass data is transferred to TBUF while TCSR.SELMD = 1.</p> <p>In the SSC protocol, this bit field can be used to define which SELOx output line will be activated when bypass data is transmitted.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
BHPC	[23:21]	rw	<p>Bypass Hardware Port Control</p> <p>This bit field contains the value that is written to SCTR[4:2] if bypass data is transferred to TBUF while TCSR.HPCMD = 1.</p> <p>In the SSC protocol, this bit field can be used to define the data shift mode and if hardware port control is enabled through CCR.HPCEN = 1, the pin direction when bypass data is transmitted.</p>
0	[7:4], 9, 14, [31:24]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

17.11.9.2 General FIFO Buffer Control Registers

The transmit and receive FIFO status information of USICx_CHy is given in registers USICx_CHy.TRBSR.

The bits related to the transmitter buffer in this register can only be written if the transmit buffer functionality is enabled by CCFG.TB = 1, otherwise write accesses are ignored. A similar behavior applies for the bits related to the receive buffer referring to CCFG.RB = 1.

The interrupt flags (event flags) in the transmit and receive FIFO status register TRBSR can be cleared by writing a 1 to the corresponding bit position in register TRBSCR, whereas writing a 0 has no effect on these bits. Writing a 1 by software to SRBI, RBERI, ARBI, STBI, or TBERI sets the corresponding bit to simulate the detection of a transmit/receive buffer event, but without activating any service request output (therefore, see FMR.SIOx).

Bits TBUS and RBUS have been implemented for testing purposes. They can be ignored by data handling software. Please note that a read action can deliver either a 0 or a 1 for these bits. It is recommended to treat them as "don't care".

Universal Serial Interface Channel (USIC)

TRBSR

Transmit/Receive Buffer Status Register

(114_H)

Reset Value: 0000 0808_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	TBFLVL							0	RBFLVL						
r	rh							r	rh						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	STB T	TBU S	TFU LL	TEM PTY	0	TBE RI	STBI	0	SRB T	RBU S	RFU LL	REM PTY	ARBI	RBE RI	SRBI
r	rh	rh	rh	rh	r	rwh	rwh	r	rh	rh	rh	rh	rwh	rwh	rwh

Field	Bits	Type	Description
SRBI	0	rwh	<p>Standard Receive Buffer Event</p> <p>This bit indicates that a standard receive buffer event has been detected. It is cleared by writing TRBSCR.CSRBI = 1.</p> <p>If enabled by RBCTR.SRBIEN, the service request output SRx selected by RBCTR.SRBINP becomes activated if a standard receive buffer event is detected.</p> <p>0_B A standard receive buffer event has not been detected.</p> <p>1_B A standard receive buffer event has been detected.</p>
RBERI	1	rwh	<p>Receive Buffer Error Event</p> <p>This bit indicates that a receive buffer error event has been detected. It is cleared by writing TRBSCR.CRBERI = 1.</p> <p>If enabled by RBCTR.RBERIEN, the service request output SRx selected by RBCTR.ARBINP becomes activated if a receive buffer error event is detected.</p> <p>0_B A receive buffer error event has not been detected.</p> <p>1_B A receive buffer error event has been detected.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
ARBI	2	rwh	<p>Alternative Receive Buffer Event</p> <p>This bit indicates that an alternative receive buffer event has been detected. It is cleared by writing <code>TRBSCR.CARBI = 1</code>.</p> <p>If enabled by <code>RBCTR.ARBIEN</code>, the service request output <code>SRx</code> selected by <code>RBCTR.ARBINP</code> becomes activated if an alternative receive buffer event is detected.</p> <p>0_B An alternative receive buffer event has not been detected.</p> <p>1_B An alternative receive buffer event has been detected.</p>
REMPY	3	rh	<p>Receive Buffer Empty</p> <p>This bit indicates whether the receive buffer is empty.</p> <p>0_B The receive buffer is not empty.</p> <p>1_B The receive buffer is empty.</p>
RFULL	4	rh	<p>Receive Buffer Full</p> <p>This bit indicates whether the receive buffer is full.</p> <p>0_B The receive buffer is not full.</p> <p>1_B The receive buffer is full.</p>
RBUS	5	rh	<p>Receive Buffer Busy</p> <p>This bit indicates whether the receive buffer is currently updated by the FIFO handler.</p> <p>0_B The receive buffer information has been completely updated.</p> <p>1_B The <code>OUTR</code> update from the FIFO memory is ongoing. A read from <code>OUTR</code> will be delayed. FIFO pointers from the previous read are not yet updated.</p>
SRBT	6	rh	<p>Standard Receive Buffer Event Trigger</p> <p>This bit triggers a standard receive buffer event when set.</p> <p>If enabled by <code>RBCTR.SRBIEN</code>, the service request output <code>SRx</code> selected by <code>RBCTR.SRBINP</code> becomes activated until the bit is cleared.</p> <p>0_B A standard receive buffer event is not triggered using this bit.</p> <p>1_B A standard receive buffer event is triggered using this bit.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
STBI	8	rwh	<p>Standard Transmit Buffer Event</p> <p>This bit indicates that a standard transmit buffer event has been detected. It is cleared by writing $TRBSCR.CSTBI = 1$.</p> <p>If enabled by $TBCTR.STBIEN$, the service request output SRx selected by $TBCTR.STBINP$ becomes activated if a standard transmit buffer event is detected.</p> <p>0_B A standard transmit buffer event has not been detected.</p> <p>1_B A standard transmit buffer event has been detected.</p>
TBERI	9	rwh	<p>Transmit Buffer Error Event</p> <p>This bit indicates that a transmit buffer error event has been detected. It is cleared by writing $TRBSCR.CTBERI = 1$.</p> <p>If enabled by $TBCTR.TBERIEN$, the service request output SRx selected by $TBCTR.ATBINP$ becomes activated if a transmit buffer error event is detected.</p> <p>0_B A transmit buffer error event has not been detected.</p> <p>1_B A transmit buffer error event has been detected.</p>
TEMPY	11	rh	<p>Transmit Buffer Empty</p> <p>This bit indicates whether the transmit buffer is empty.</p> <p>0_B The transmit buffer is not empty.</p> <p>1_B The transmit buffer is empty.</p>
TFULL	12	rh	<p>Transmit Buffer Full</p> <p>This bit indicates whether the transmit buffer is full.</p> <p>0_B The transmit buffer is not full.</p> <p>1_B The transmit buffer is full.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
TBUS	13	rh	<p>Transmit Buffer Busy</p> <p>This bit indicates whether the transmit buffer is currently updated by the FIFO handler.</p> <p>0_B The transmit buffer information has been completely updated.</p> <p>1_B The FIFO memory update after write to INx is ongoing. A write to INx will be delayed. FIFO pointers from the previous INx write are not yet updated.</p>
STBT	14	rh	<p>Standard Transmit Buffer Event Trigger</p> <p>This bit triggers a standard transmit buffer event when set.</p> <p>If enabled by TBCTR.STBIEN, the service request output SRx selected by TBCTR.STBINP becomes activated until the bit is cleared.</p> <p>0_B A standard transmit buffer event is not triggered using this bit.</p> <p>1_B A standard transmit buffer event is triggered using this bit.</p>
RBFLVL	[22:16]	rh	<p>Receive Buffer Filling Level</p> <p>This bit field indicates the filling level of the receive buffer, starting with 0 for an empty buffer.</p>
TBFLVL	[30:24]	rh	<p>Transmit Buffer Filling Level</p> <p>This bit field indicates the filling level of the transmit buffer, starting with 0 for an empty buffer.</p> <p><i>Note: The first data word written to Transmit FIFO will be loaded immediately to TBUF and removed from FIFO.</i></p>
0	7, 10, 15, 23, 31	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Universal Serial Interface Channel (USIC)

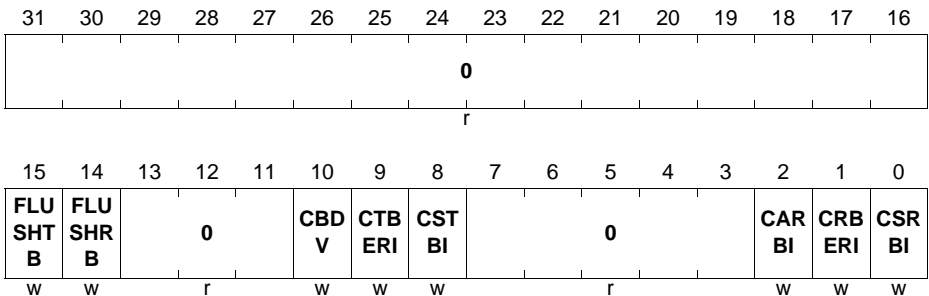
The bits in register TRBSCR are used to clear the notification bits in register TRBSR or to clear the FIFO mechanism for the transmit or receive buffer. A read action always delivers 0.

TRBSCR

Transmit/Receive Buffer Status Clear Register

(118_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CSRBI	0	w	Clear Standard Receive Buffer Event 0 _B No effect. 1 _B Clear TRBSR.SRBI.
CRBERI	1	w	Clear Receive Buffer Error Event 0 _B No effect. 1 _B Clear TRBSR.RBERI.
CARBI	2	w	Clear Alternative Receive Buffer Event 0 _B No effect. 1 _B Clear TRBSR.ARBI.
CSTBI	8	w	Clear Standard Transmit Buffer Event 0 _B No effect. 1 _B Clear TRBSR.STBI.
CTBERI	9	w	Clear Transmit Buffer Error Event 0 _B No effect. 1 _B Clear TRBSR.TBERI.
CBDV	10	w	Clear Bypass Data Valid 0 _B No effect. 1 _B Clear BYPCR.BDV.

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
FLUSHRB	14	w	Flush Receive Buffer 0 _B No effect. 1 _B The receive FIFO buffer is cleared (filling level is cleared and output pointer is set to input pointer value). Should only be used while the FIFO buffer is not taking part in data traffic.
FLUSHTB	15	w	Flush Transmit Buffer 0 _B No effect. 1 _B The transmit FIFO buffer is cleared (filling level is cleared and output pointer is set to input pointer value). Should only be used while the FIFO buffer is not taking part in data traffic.
0	[7:3], [13:11], [31:16]	r	Reserved Read as 0; should be written with 0.

17.11.9.3 Transmit FIFO Buffer Control Registers

The transmit FIFO buffer is controlled by register TBCTR. TBCTR can only be written if the transmit buffer functionality is enabled by CCFG.TB = 1, otherwise write accesses are ignored.

TBCTR

Transmitter Buffer Control Register (108_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBE RIEN	STBI EN	0	LOF	0	SIZE			0	ATBINP			STBINP			
r/w	r/w	r	r/w	r	r/w			r	r/w			r/w			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STB TEN	STB TM	LIMIT					0	DPTR							
r/w	r/w	r/w					r	w							

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
DPTR	[5:0]	w	<p>Data Pointer</p> <p>This bit field defines the start value for the transmit buffer pointers when assigning the FIFO entries to the transmit FIFO buffer. A read always delivers 0. When writing DPTR while SIZE = 0, both transmitter pointers TDIPTR and RTDOPTR in register TRBPTR are updated with the written value and the buffer is considered as empty. A write access to DPTR while SIZE > 0 is ignored and does not modify the pointers.</p>
LIMIT	[13:8]	rw	<p>Limit For Interrupt Generation</p> <p>This bit field defines the target filling level of the transmit FIFO buffer that is used for the standard transmit buffer event detection.</p>
STBTM	14	rw	<p>Standard Transmit Buffer Trigger Mode</p> <p>This bit selects the standard transmit buffer event trigger mode.</p> <p>0_B Trigger mode 0: While TRBSR.STBT=1, a standard buffer event will be generated whenever there is a data transfer to TBUF or data write to INx (depending on TBCTR.LOF setting). STBT is cleared when TRBSR.TBFLVL=TBCTR.LIMIT.</p> <p>1_B Trigger mode 1: While TRBSR.STBT=1, a standard buffer event will be generated whenever there is a data transfer to TBUF or data write to INx (depending on TBCTR.LOF setting). STBT is cleared when TRBSR.TBFLVL=TBCTR.SIZE.</p>
STBTEN	15	rw	<p>Standard Transmit Buffer Trigger Enable</p> <p>This bit enables/disables triggering of the standard transmit buffer event through bit TRBSR.STBT.</p> <p>0_B The standard transmit buffer event trigger through bit TRBSR.STBT is disabled.</p> <p>1_B The standard transmit buffer event trigger through bit TRBSR.STBT is enabled.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
STBINP	[18:16]	rw	<p>Standard Transmit Buffer Interrupt Node Pointer</p> <p>This bit field defines which service request output SRx becomes activated in case of a standard transmit buffer event.</p> <p>000_B Output SR0 becomes activated. 001_B Output SR1 becomes activated. 010_B Output SR2 becomes activated. 011_B Output SR3 becomes activated. 100_B Output SR4 becomes activated. 101_B Output SR5 becomes activated.</p> <p><i>Note: All other settings of the bit field are reserved.</i></p>
ATBINP	[21:19]	rw	<p>Alternative Transmit Buffer Interrupt Node Pointer</p> <p>This bit field define which service request output SRx will be activated in case of a transmit buffer error event.</p> <p>000_B Output SR0 becomes activated. 001_B Output SR1 becomes activated. 010_B Output SR2 becomes activated. 011_B Output SR3 becomes activated. 100_B Output SR4 becomes activated. 101_B Output SR5 becomes activated.</p> <p><i>Note: All other settings of the bit field are reserved.</i></p>
SIZE	[26:24]	rw	<p>Buffer Size</p> <p>This bit field defines the number of FIFO entries assigned to the transmit FIFO buffer.</p> <p>000_B The FIFO mechanism is disabled. The buffer does not accept any request for data. 001_B The FIFO buffer contains 2 entries. 010_B The FIFO buffer contains 4 entries. 011_B The FIFO buffer contains 8 entries. 100_B The FIFO buffer contains 16 entries. 101_B The FIFO buffer contains 32 entries. 110_B The FIFO buffer contains 64 entries. 111_B Reserved</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
LOF	28	rw	<p>Buffer Event on Limit Overflow</p> <p>This bit defines which relation between filling level and programmed limit leads to a standard transmit buffer event.</p> <p>0_B A standard transmit buffer event occurs when the filling level equals the limit value and gets lower due to transmission of a data word.</p> <p>1_B A standard transmit buffer interrupt event occurs when the filling level equals the limit value and gets bigger due to a write access to a data input location INx.</p>
STBIEN	30	rw	<p>Standard Transmit Buffer Interrupt Enable</p> <p>This bit enables/disables the generation of a standard transmit buffer interrupt in case of a standard transmit buffer event.</p> <p>0_B The standard transmit buffer interrupt generation is disabled.</p> <p>1_B The standard transmit buffer interrupt generation is enabled.</p>
TBERIEN	31	rw	<p>Transmit Buffer Error Interrupt Enable</p> <p>This bit enables/disables the generation of a transmit buffer error interrupt in case of a transmit buffer error event (software writes to a full transmit buffer).</p> <p>0_B The transmit buffer error interrupt generation is disabled.</p> <p>1_B The transmit buffer error interrupt generation is enabled.</p>
0	[7:6], [23:22], 27, 29	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Universal Serial Interface Channel (USIC)

17.11.9.4 Receive FIFO Buffer Control Registers

The receive FIFO buffer is controlled by register RBCTR. This register can only be written if the receive buffer functionality is enabled by CCFG.RB = 1, otherwise write accesses are ignored.

RBCTR

Receiver Buffer Control Register (10C_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
RBE RIEN	SRBI EN	ARBI EN	LOF	RNM	SIZE			RCIM			ARBINP			SRBINP		
rw	rw	rw	rw	rw	rw			rw			rw			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
SRB TEN	SRB TM	LIMIT				0			DPTR							
rw	rw	rw				r			w							

Field	Bits	Type	Description
DPTR	[5:0]	w	Data Pointer This bit field defines the start value for the receive buffer pointers when assigning the FIFO entries to the receive FIFO buffer. A read always delivers 0. When writing DPTR while SIZE = 0, both receiver pointers RDIPTR and RDOPTR in register TRBPTR are updated with the written value and the buffer is considered as empty. A write access to DPTR while SIZE > 0 is ignored and does not modify the pointers.
LIMIT	[13:8]	rw	Limit For Interrupt Generation This bit field defines the target filling level of the receive FIFO buffer that is used for the standard receive buffer event detection.

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
SRBTM	14	rw	<p>Standard Receive Buffer Trigger Mode</p> <p>This bit selects the standard receive buffer event trigger mode.</p> <p>0_B Trigger mode 0: While TRBSR.SRBT=1, a standard receive buffer event will be generated whenever there is a new data received or data read out (depending on RBCTR.LOF setting). SRBT is cleared when TRBSR.RBFLVL=RBCTR.LIMIT.</p> <p>1_B Trigger mode 1: While TRBSR.SRBT=1, a standard receive buffer event will be generated whenever there is a new data received or data read out (depending on RBCTR.LOF setting). SRBT is cleared when TRBSR.RBFLVL=0.</p>
SRBTEN	15	rw	<p>Standard Receive Buffer Trigger Enable</p> <p>This bit enables/disables triggering of the standard receive buffer event through bit TRBSR.SRBT.</p> <p>0_B The standard receive buffer event trigger through bit TRBSR.SRBT is disabled.</p> <p>1_B The standard receive buffer event trigger through bit TRBSR.SRBT is enabled.</p>
SRBINP	[18:16]	rw	<p>Standard Receive Buffer Interrupt Node Pointer</p> <p>This bit field defines which service request output SRx becomes activated in case of a standard receive buffer event.</p> <p>000_B Output SR0 becomes activated. 001_B Output SR1 becomes activated. 010_B Output SR2 becomes activated. 011_B Output SR3 becomes activated. 100_B Output SR4 becomes activated. 101_B Output SR5 becomes activated.</p> <p><i>Note: All other settings of the bit field are reserved.</i></p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
ARBINP	[21:19]	rw	<p>Alternative Receive Buffer Interrupt Node Pointer</p> <p>This bit field defines which service request output SRx becomes activated in case of an alternative receive buffer event or a receive buffer error event.</p> <p>000_B Output SR0 becomes activated. 001_B Output SR1 becomes activated. 010_B Output SR2 becomes activated. 011_B Output SR3 becomes activated. 100_B Output SR4 becomes activated. 101_B Output SR5 becomes activated.</p> <p><i>Note: All other settings of the bit field are reserved.</i></p>
RCIM	[23:22]	rw	<p>Receiver Control Information Mode</p> <p>This bit field defines which information from the receiver status register RBUF SR is propagated as 5 bit receiver control information RCI[4:0] to the receive FIFO buffer and can be read out in registers OUT(D)R.</p> <p>00_B RCI[4] = PERR, RCI[3:0] = WLEN 01_B RCI[4] = SOF, RCI[3:0] = WLEN 10_B RCI[4] = 0, RCI[3:0] = WLEN 11_B RCI[4] = PERR, RCI[3] = PAR, RCI[2:1] = 00_B, RCI[0] = SOF</p>
SIZE	[26:24]	rw	<p>Buffer Size</p> <p>This bit field defines the number of FIFO entries assigned to the receive FIFO buffer.</p> <p>000_B The FIFO mechanism is disabled. The buffer does not accept any request for data. 001_B The FIFO buffer contains 2 entries. 010_B The FIFO buffer contains 4 entries. 011_B The FIFO buffer contains 8 entries. 100_B The FIFO buffer contains 16 entries. 101_B The FIFO buffer contains 32 entries. 110_B The FIFO buffer contains 64 entries. 111_B Reserved</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
RNM	27	rw	<p>Receiver Notification Mode</p> <p>This bit defines the receive buffer event mode. The receive buffer error event is not affected by RNM.</p> <p>0_B Filling level mode: A standard receive buffer event occurs when the filling level equals the limit value and changes, either due to a read access from OUTR (LOF = 0) or due to a new received data word (LOF = 1).</p> <p>1_B RCI mode: A standard receive buffer event occurs when register OUTR is updated with a new value if the corresponding value in OUTR.RCI[4] = 0. If OUTR.RCI[4] = 1, an alternative receive buffer event occurs instead of the standard receive buffer event.</p>
LOF	28	rw	<p>Buffer Event on Limit Overflow</p> <p>This bit defines which relation between filling level and programmed limit leads to a standard receive buffer event in filling level mode (RNM = 0). In RCI mode (RNM = 1), bit fields LIMIT and LOF are ignored.</p> <p>0_B A standard receive buffer event occurs when the filling level equals the limit value and gets lower due to a read access from OUTR.</p> <p>1_B A standard receive buffer event occurs when the filling level equals the limit value and gets bigger due to the reception of a new data word.</p>
ARBIEN	29	rw	<p>Alternative Receive Buffer Interrupt Enable</p> <p>This bit enables/disables the generation of an alternative receive buffer interrupt in case of an alternative receive buffer event.</p> <p>0_B The alternative receive buffer interrupt generation is disabled.</p> <p>1_B The alternative receive buffer interrupt generation is enabled.</p>

Universal Serial Interface Channel (USIC)

Field	Bits	Type	Description
SRBIEN	30	rw	<p>Standard Receive Buffer Interrupt Enable This bit enables/disables the generation of a standard receive buffer interrupt in case of a standard receive buffer event.</p> <p>0_B The standard receive buffer interrupt generation is disabled.</p> <p>1_B The standard receive buffer interrupt generation is enabled.</p>
RBERIEN	31	rw	<p>Receive Buffer Error Interrupt Enable This bit enables/disables the generation of a receive buffer error interrupt in case of a receive buffer error event (the software reads from an empty receive buffer).</p> <p>0_B The receive buffer error interrupt generation is disabled.</p> <p>1_B The receive buffer error interrupt generation is enabled.</p>
0	[7:6]	r	<p>Reserved Read as 0; should be written with 0.</p>

Universal Serial Interface Channel (USIC)

17.11.9.5 FIFO Buffer Data Registers

The 32 independent data input locations IN00 to IN31 are addresses that can be used as data entry locations for the transmit FIFO buffer. Data written to one of these locations will be stored in the transmit buffer FIFO. Additionally, the 5-bit coding of the number [31:0] of the addressed data input location represents the transmit control information TCI.

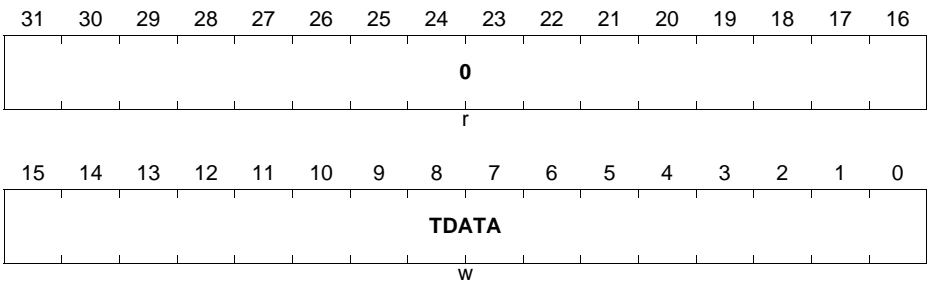
If the FIFO is already full and new data is written to it, the write access is ignored and a transmit buffer error event is signaled.

INx (x = 00-31)

Transmit FIFO Buffer Input Location x

$$(180_H + x * 4)$$

Reset Value: 0000 0000_H



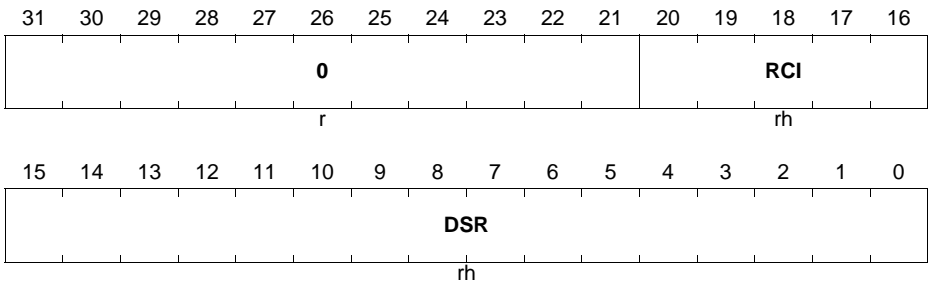
Field	Bits	Type	Description
TDATA	[15:0]	w	Transmit Data This bit field contains the data to be transmitted (write view), read actions deliver 0. A write action to at least the low byte of TDATA triggers the data storage in the FIFO.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Universal Serial Interface Channel (USIC)

The receiver FIFO buffer output register OUTR shows the oldest received data word in the FIFO buffer and contains the receiver control information RCI containing the information selected by RBCTR.RCIM. A read action from this address location delivers the received data. With a read access of at least the low byte, the data is declared to be read and the next entry becomes visible. Write accesses to OUTR are ignored.

OUTR

Receiver Buffer Output Register (11C_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DSR	[15:0]	rh	Received Data This bit field monitors the content of the oldest data word in the receive FIFO. Reading at least the low byte releases the buffer entry currently shown in DSR.
RCI	[20:16]	rh	Receiver Control Information This bit field monitors the receiver control information associated to DSR. The bit structure of RCI depends on bit field RBCTR.RCIM.
0	[31:21]	r	Reserved Read as 0; should be written with 0.

Universal Serial Interface Channel (USIC)

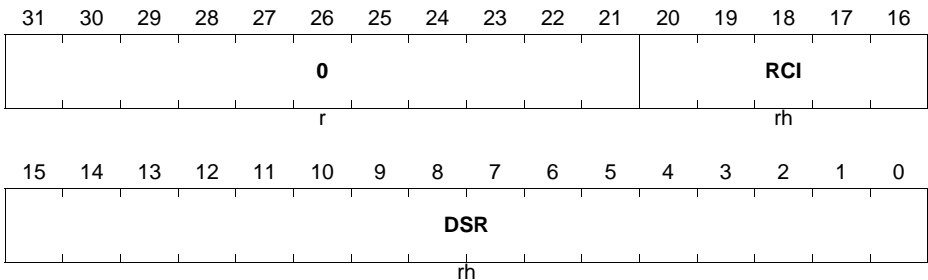
If a debugger should be used to monitor the received data in the FIFO buffer, the FIFO mechanism must not be activated in order to guaranty data consistency. Therefore, a second address set is available, named OUTDR (D like debugger), having the same bit fields like the original buffer output register OUTR, but without the FIFO mechanism. A debugger can read here (in order to monitor the receive data flow) without the risk of data corruption. Write accesses to OUTDR are ignored.

OUTDR

Receiver Buffer Output Register L for Debugger

(120_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DSR	[15:0]	rh	Data from Shift Register Same as OUTR.DSR, but without releasing the buffer after a read action.
RCI	[20:16]	rh	Receive Control Information from Shift Register Same as OUTR.RCI.
0	[31:21]	r	Reserved Read as 0; should be written with 0.

Universal Serial Interface Channel (USIC)

17.11.9.6 FIFO Buffer Pointer Registers

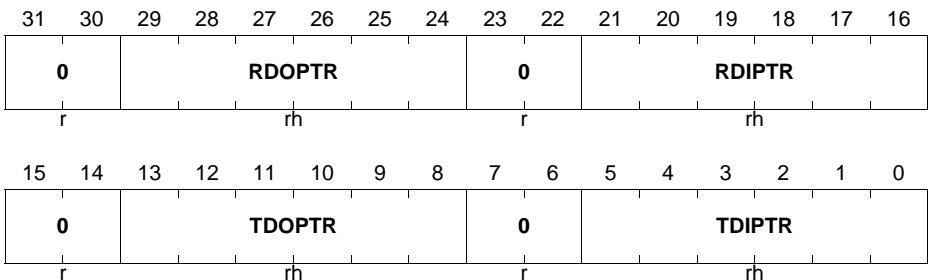
The pointers for FIFO handling of the transmit and receive FIFO buffers are located in register TRBPTR. The pointers are automatically handled by the FIFO buffer mechanism and do not need to be modified by software. As a consequence, these registers can only be read by software (e.g. for verification purposes), whereas write accesses are ignored.

TRBPTR

Transmit/Receive Buffer Pointer Register

(110_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
TDIPTTR	[5:0]	rh	Transmitter Data Input Pointer This bit field indicates the buffer entry that will be used for the next transmit data coming from the INx addresses.
TDOPTR	[13:8]	rh	Transmitter Data Output Pointer This bit field indicates the buffer entry that will be used for the next transmit data to be output to TBUF.
RDIPTTR	[21:16]	rh	Receiver Data Input Pointer This bit field indicates the buffer entry that will be used for the next receive data coming from RBUF.
RDOPTR	[29:24]	rh	Receiver Data Output Pointer This bit field indicates the buffer entry that will be used for the next receive data to be output at the OUT(D)R addresses.
0	[7:6], [15:14], [23:22], [31:30]	r	Reserved Read as 0; should be written with 0.

Universal Serial Interface Channel (USIC)

17.12 Interconnects

The XMC4300 device contains two USIC modules (USIC0 and USIC1) with 2 communication channels each.

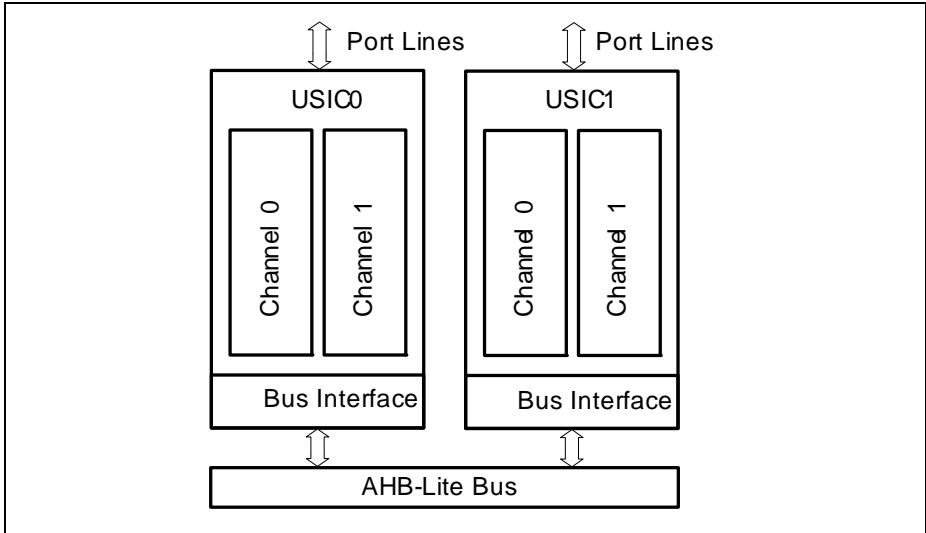


Figure 17-66 USIC Module Structure in XMC4300

Figure 17-67 shows the I/O lines of one USIC channel. The tables in this section define the pin assignments and internal connections of the USIC channels I/O lines in the XMC4300 device. Naming convention: USIC_x_CH_y refers to USIC module x channel y. The meaning of these pins with respect to each protocol is described in the protocols' respective chapters and summarized in [Table 17-2](#) and [Table 17-3](#).

Universal Serial Interface Channel (USIC)

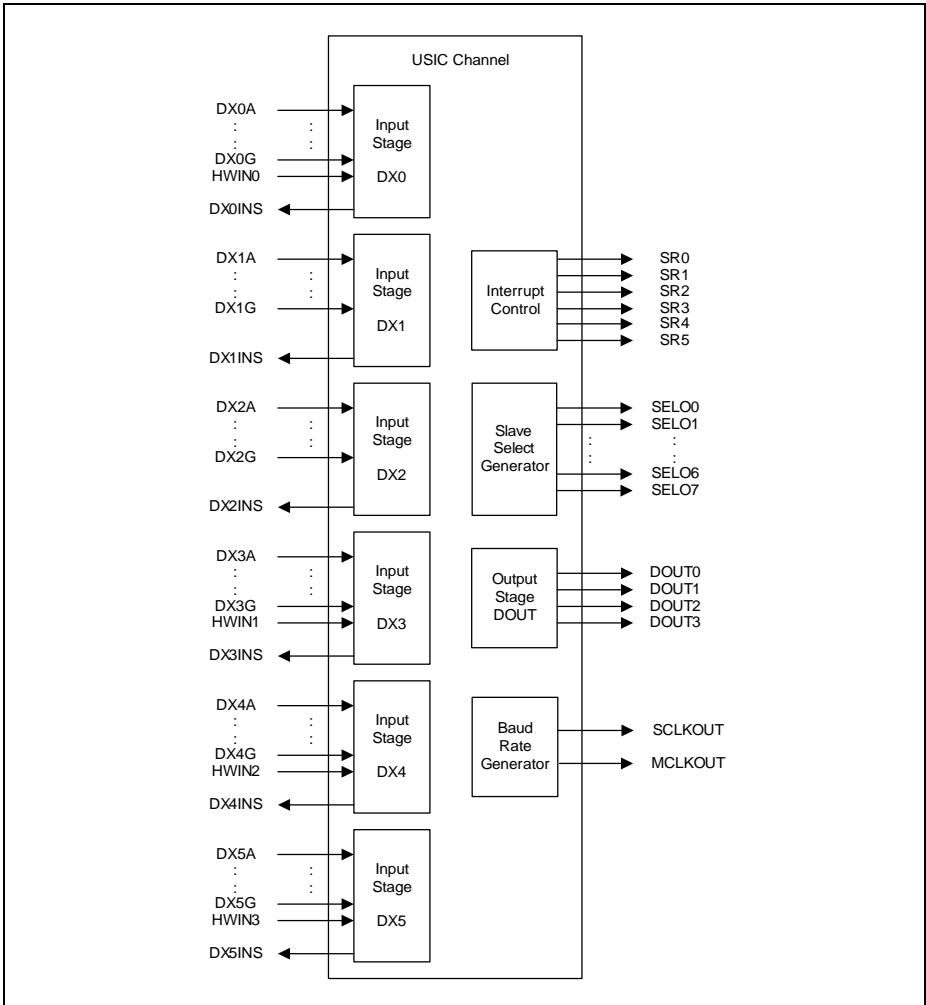


Figure 17-67 USIC Channel I/O Lines

The service request outputs SR[5:0] of one USIC channel is combined with those of the other channel with the module. Therefore, only 6 service request outputs are available per module.

17.12.1 USIC Module 0 Interconnects

The interconnects of USIC module 0 is grouped into the following categories:

Universal Serial Interface Channel (USIC)

- **USIC Module 0 Channel 0 Interconnects (Table 17-23)**
- **USIC Module 0 Channel 1 Interconnects (Table 17-24)**
- **USIC Module 0 Module Interconnects (Table 17-25)**

Table 17-23 USIC Module 0 Channel 0 Interconnects

Input/Output	I/O	Connected To	Description
Data Inputs (DX0)			
USIC0_CH0.DX0A	I	P1.5	Shift data input; used for: <ul style="list-style-type: none"> • ASC RXD • SSC MTSR/MRST • IIC SDA • IIS DIN
USIC0_CH0.DX0B	I	P1.4	
USIC0_CH0.DX0C	I	P4.7	
USIC0_CH0.DX0D	I	P5.0	
USIC0_CH0.DX0E	I	0	
USIC0_CH0.DX0F	I	XTAL1	
USIC0_CH0.DX0G	I	USIC0_CH0.DOUT0	Loop back shift data input
USIC0_CH0.HWIN0	I	P1.5	HW controlled shift data input
Clock Inputs			
USIC0_CH0.DX1A	I	P1.1	Shift clock input; used for: <ul style="list-style-type: none"> • SSC Slave SCLKIN • IIC SCL • IIS Slave SCLKIN • Optional for ASC, SSC Master and IIS Master
USIC0_CH0.DX1B	I	P0.8	
USIC0_CH0.DX1C	I	0	
USIC0_CH0.DX1D	I	0	
USIC0_CH0.DX1E	I	0	
USIC0_CH0.DX1F	I	USIC0_CH0.DX0INS	
USIC0_CH0.DX1G	I	USIC0_CH0.SCLKOUT	
Control Inputs			
USIC0_CH0.DX2A	I	P1.0	Shift control input; used for: <ul style="list-style-type: none"> • SSC Slave SELIN • IIS Slave WAIN • Optional for ASC, SSC Master, IIC and IIS Master
USIC0_CH0.DX2B	I	P0.7	
USIC0_CH0.DX2C	I	0	
USIC0_CH0.DX2D	I	0	
USIC0_CH0.DX2E	I	CCU40.SR1	
USIC0_CH0.DX2F	I	CCU80.SR1	
USIC0_CH0.DX2G	I	USIC0_CH0.SELO0	Loop back shift control input
Data Inputs (DX3)			

Universal Serial Interface Channel (USIC)

Table 17-23 USIC Module 0 Channel 0 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC0_CH0.DX3A	I	0	Shift data input; used for: <ul style="list-style-type: none"> • Dual and Quad SSC MTSR1/MRST1 • Can be ignored for all other protocols
USIC0_CH0.DX3B	I	0	
USIC0_CH0.DX3C	I	0	
USIC0_CH0.DX3D	I	0	
USIC0_CH0.DX3E	I	0	
USIC0_CH0.DX3F	I	0	
USIC0_CH0.DX3G	I	USIC0_CH0.DOUT1	Loop back shift data input
USIC0_CH0.HWIN1	I	P1.4	HW controlled shift data input
Data Inputs (DX4)			
USIC0_CH0.DX4A	I	0	Shift data input; used for: <ul style="list-style-type: none"> • Quad SSC MTSR2/MRST2 • Can be ignored for all other protocols
USIC0_CH0.DX4B	I	0	
USIC0_CH0.DX4C	I	0	
USIC0_CH0.DX4D	I	0	
USIC0_CH0.DX4E	I	0	
USIC0_CH0.DX4F	I	0	
USIC0_CH0.DX4G	I	USIC0_CH0.DOUT2	Loop back shift data input
USIC0_CH0.HWIN2	I	P1.3	HW controlled shift data input
Data Inputs (DX5)			
USIC0_CH0.DX5A	I	0	Shift data input; used for: <ul style="list-style-type: none"> • Quad SSC MTSR3/MRST3 • Can be ignored for all other protocols
USIC0_CH0.DX5B	I	0	
USIC0_CH0.DX5C	I	0	
USIC0_CH0.DX5D	I	0	
USIC0_CH0.DX5E	I	0	
USIC0_CH0.DX5F	I	0	
USIC0_CH0.DX5G	I	USIC0_CH0.DOUT3	Loop back shift data input
USIC0_CH0.HWIN3	I	P1.2	HW controlled shift data input
Data Outputs			

Universal Serial Interface Channel (USIC)

Table 17-23 USIC Module 0 Channel 0 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC0_CH0.DOUT0	O	P1.5 P1.7 P5.1 P1.5.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • ASC TXD • SSC MTSR/MRST • IIC SDA • IIS DOUT
USIC0_CH0.DOUT1	O	P1.4.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • Dual and Quad SSC MTSR1/MRST1 • Can be ignored for all other protocols
USIC0_CH0.DOUT2	O	P1.3.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • Quad SSC MTSR2/MRST2 • Can be ignored for all other protocols
USIC0_CH0.DOUT3	O	P1.2.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • Quad SSC MTSR3/MRST3 • Can be ignored for all other protocols

Clock Outputs

USIC0_CH0.MCLKOUT	O	P1.3	Master clock output (optional for all protocols)
USIC0_CH0.SCLKOUT	O	P0.8 P1.1 P1.6 P1.10	Shift clock output; used for: <ul style="list-style-type: none"> • Master SCLKOUT in SSC and IIS • IIC SCL • Can be ignored in ASC

Control Outputs

Universal Serial Interface Channel (USIC)

Table 17-23 USIC Module 0 Channel 0 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC0_CH0.SELO0	O	P0.7 P1.0 P1.11	Shift control output; used for: <ul style="list-style-type: none"> • SSC Master SELO • IIS WA • Can be ignored for all other protocols
USIC0_CH0.SELO1	O	P1.8	
USIC0_CH0.SELO2	O	P4.6	
USIC0_CH0.SELO3	O	P4.5	
USIC0_CH0.SELO4	O	P4.4	
USIC0_CH0.SELO5	O	P4.3	
USIC0_CH0.SELO6	O	not connected	
USIC0_CH0.SELO7	O	not connected	

System Related Outputs

USIC0_CH0.DX0INS	O	USIC0_CH0.DX1F	Selected DX0 input signal
USIC0_CH0.DX1INS	O	DAC.TRIGGER[6]	Selected DX1 input signal
USIC0_CH0.DX2INS	O	CCU40.IN0L CCU42.IN0L CCU43.IN0L	Selected DX2 input signal
USIC0_CH0.DX3INS	O	not connected	Selected DX3 input signal
USIC0_CH0.DX4INS	O	not connected	Selected DX4 input signal
USIC0_CH0.DX5INS	O	not connected	Selected DX5 input signal

Table 17-24 USIC Module 0 Channel 1 Interconnects

Input/Output	I/O	Connected To	Description
Data Inputs (DX0)			
USIC0_CH1.DX0A	I	P2.2	Shift data input; used for: <ul style="list-style-type: none"> • ASC RXD • SSC MTSR/MRST • IIC SDA • IIS DIN
USIC0_CH1.DX0B	I	P2.5	
USIC0_CH1.DX0C	I	P6.3	
USIC0_CH1.DX0D	I	P3.13	
USIC0_CH1.DX0E	I	P4.0	
USIC0_CH1.DX0F	I	XTAL1	
USIC0_CH1.DX0G	I	USIC0_CH1.DOUT0	Loop back shift data input
USIC0_CH1.HWIN0	I	P3.13	HW controlled shift data input

Universal Serial Interface Channel (USIC)

Table 17-24 USIC Module 0 Channel 1 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
Clock Inputs			
USIC0_CH1.DX1A	I	P2.4	Shift clock input; used for: <ul style="list-style-type: none"> • SSC Slave SCLKIN • IIC SCL • IIS Slave SCLKIN • Optional for ASC, SSC Master and IIS Master
USIC0_CH1.DX1B	I	P3.0	
USIC0_CH1.DX1C	I	P6.2	
USIC0_CH1.DX1D	I	0	
USIC0_CH1.DX1E	I	0	
USIC0_CH1.DX1F	I	USIC0_CH1.DX0INS	
USIC0_CH1.DX1G	I	USIC0_CH1.SCLKOUT	Loop back shift clock input
Control Inputs			
USIC0_CH1.DX2A	I	P2.3	Shift control input; used for: <ul style="list-style-type: none"> • SSC Slave SELIN • IIS Slave WAIN • Optional for ASC, SSC Master, IIC and IIS Master
USIC0_CH1.DX2B	I	P3.1	
USIC0_CH1.DX2C	I	P6.1	
USIC0_CH1.DX2D	I	0	
USIC0_CH1.DX2E	I	CCU42.SR1	
USIC0_CH1.DX2F	I	CCU80.SR1	
USIC0_CH1.DX2G	I	USIC0_CH1.SELO0	Loop back shift control input
Data Inputs (DX3)			
USIC0_CH1.DX3A	I	0	Shift data input; used for: <ul style="list-style-type: none"> • Dual and Quad SSC MTSR1/MRST1 • Can be ignored for all other protocols
USIC0_CH1.DX3B	I	0	
USIC0_CH1.DX3C	I	0	
USIC0_CH1.DX3D	I	0	
USIC0_CH1.DX3E	I	0	
USIC0_CH1.DX3F	I	0	
USIC0_CH1.DX3G	I	USIC0_CH1.DOUT1	Loop back shift data input
USIC0_CH1.HWIN1	I	P3.12	HW controlled shift data input
Data Inputs (DX4)			

Universal Serial Interface Channel (USIC)

Table 17-24 USIC Module 0 Channel 1 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC0_CH1.DX4A	I	0	Shift data input; used for: <ul style="list-style-type: none"> • Quad SSC MTSR2/MRST2 • Can be ignored for all other protocols
USIC0_CH1.DX4B	I	0	
USIC0_CH1.DX4C	I	0	
USIC0_CH1.DX4D	I	0	
USIC0_CH1.DX4E	I	0	
USIC0_CH1.DX4F	I	0	
USIC0_CH1.DX4G	I	USIC0_CH1.DOUT2	Loop back shift data input
USIC0_CH1.HWIN2	I	P3.11	HW controlled shift data input
Data Inputs (DX5)			
USIC0_CH1.DX5A	I	0	Shift data input; used for: <ul style="list-style-type: none"> • Quad SSC MTSR3/MRST3 • Can be ignored for all other protocols
USIC0_CH1.DX5B	I	0	
USIC0_CH1.DX5C	I	0	
USIC0_CH1.DX5D	I	0	
USIC0_CH1.DX5E	I	0	
USIC0_CH1.DX5F	I	0	
USIC0_CH1.DX5G	I	USIC0_CH1.DOUT3	Loop back shift data input
USIC0_CH1.HWIN3	I	P3.10	HW controlled shift data input
Data Outputs			
USIC0_CH1.DOUT0	O	P2.5 P3.5 P3.13 P6.4 P3.13.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • ASC TXD • SSC MTSR/MRST • IIC SDA • IIS DOUT
USIC0_CH1.DOUT1	O	P3.12.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • Dual and Quad SSC MTSR1/MRST1 • Can be ignored for all other protocols
USIC0_CH1.DOUT2	O	P3.11.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • Quad SSC MTSR2/MRST2 • Can be ignored for all other protocols

Universal Serial Interface Channel (USIC)

Table 17-24 USIC Module 0 Channel 1 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC0_CH1.DOUT3	O	P3.10.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • Quad SSC MTSR3/MRST3 • Can be ignored for all other protocols
Clock Outputs			
USIC0_CH1.MCLKOUT	O	P6.5	Master clock output (optional for all protocols)
USIC0_CH1.SCLKOUT	O	P2.4 P3.0 P3.6 P6.2	Shift clock output; used for: <ul style="list-style-type: none"> • Master SCLKOUT in SSC and IIS • IIC SCL • Can be ignored in ASC
Control Outputs			
USIC0_CH1.SELO0	O	P2.3 P3.1 P4.1 P6.1	Shift control output; used for: <ul style="list-style-type: none"> • SSC Master SELO • IIS WA • Can be ignored for all other protocols
USIC0_CH1.SELO1	O	P3.12 P6.0	
USIC0_CH1.SELO2	O	P1.14 P3.11	
USIC0_CH1.SELO3	O	P1.13 P3.8	
USIC0_CH1.SELO4	O	not connected	
USIC0_CH1.SELO5	O	not connected	
USIC0_CH1.SELO6	O	not connected	
USIC0_CH1.SELO7	O	not connected	
System Related Outputs			
USIC0_CH1.DX0INS	O	USIC0_CH1.DX1F	Selected DX0 input signal
USIC0_CH1.DX1INS	O	not connected	Selected DX1 input signal
USIC0_CH1.DX2INS	O	CCU40.IN2L CCU42.IN1L CCU43.IN1L	Selected DX2 input signal
USIC0_CH1.DX3INS	O	not connected	Selected DX3 input signal

Universal Serial Interface Channel (USIC)

Table 17-24 USIC Module 0 Channel 1 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC0_CH1.DX4INS	O	not connected	Selected DX4 input signal
USIC0_CH1.DX5INS	O	not connected	Selected DX5 input signal

Table 17-25 USIC Module 0 Module Interconnects

Input/Output	I/O	Connected To	Description
USIC0_SR[1:0]	O	NVIC GPDMA	interrupt output lines (service requests SRx)
USIC0_SR[5:2]	O	NVIC	interrupt output lines (service requests SRx)

17.12.2 USIC Module 1 Interconnects

The interconnects of USIC module 1 is grouped into the following three categories:

- **USIC Module 1 Channel 0 Interconnects** (Table 17-26)
- **USIC Module 1 Channel 1 Interconnects** (Table 17-27)
- **USIC Module 1 Module Interconnects** (Table 17-28)

Table 17-26 USIC Module 1 Channel 0 Interconnects

Input/Output	I/O	Connected To	Description
Data Inputs (DX0)			
USIC1_CH0.DX0A	I	P0.4	Shift data input; used for: <ul style="list-style-type: none"> • ASC RXD • SSC MTSR/MRST • IIC SDA • IIS DIN
USIC1_CH0.DX0B	I	P0.5	
USIC1_CH0.DX0C	I	P2.15	
USIC1_CH0.DX0D	I	P2.14	
USIC1_CH0.DX0E	I	0	
USIC1_CH0.DX0F	I	XTAL1	
USIC1_CH0.DX0G	I	USIC1_CH0.DOUT0	Loop back shift data input
USIC1_CH0.HWIN0	I	P0.5	HW controlled shift data input
Clock Inputs			

Universal Serial Interface Channel (USIC)

Table 17-26 USIC Module 1 Channel 0 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC1_CH0.DX1A	I	P0.11	Shift clock input; used for: <ul style="list-style-type: none"> • SSC Slave SCLKIN • IIC SCL • IIS Slave SCLKIN • Optional for ASC, SSC Master and IIS Master
USIC1_CH0.DX1B	I	P5.8	
USIC1_CH0.DX1C	I	0	
USIC1_CH0.DX1D	I	0	
USIC1_CH0.DX1E	I	0	
USIC1_CH0.DX1F	I	USIC1_CH0.DX0INS	
USIC1_CH0.DX1G	I	USIC1_CH0.SCLKOUT	Loop back shift clock input

Control Inputs

USIC1_CH0.DX2A	I	P0.6	Shift control input; used for: <ul style="list-style-type: none"> • SSC Slave SELIN • IIS Slave WAIN • Optional for ASC, SSC Master, IIC and IIS Master
USIC1_CH0.DX2B	I	P5.9	
USIC1_CH0.DX2C	I	0	
USIC1_CH0.DX2D	I	0	
USIC1_CH0.DX2E	I	CCU41.SR1	
USIC1_CH0.DX2F	I	CCU81.SR1	
USIC1_CH0.DX2G	I	USIC1_CH0.SELO0	Loop back shift control input

Data Inputs (DX3)

USIC1_CH0.DX3A	I	0	Shift data input; used for: <ul style="list-style-type: none"> • Dual and Quad SSC MTSR1/MRST1 • Can be ignored for all other protocols
USIC1_CH0.DX3B	I	0	
USIC1_CH0.DX3C	I	0	
USIC1_CH0.DX3D	I	0	
USIC1_CH0.DX3E	I	0	
USIC1_CH0.DX3F	I	0	
USIC1_CH0.DX3G	I	USIC1_CH0.DOUT1	Loop back shift data input
USIC1_CH0.HWIN1	I	P0.4	HW controlled shift data input

Data Inputs (DX4)

Universal Serial Interface Channel (USIC)

Table 17-26 USIC Module 1 Channel 0 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC1_CH0.DX4A	I	0	Shift data input; used for: <ul style="list-style-type: none"> • Quad SSC MTSR2/MRST2 • Can be ignored for all other protocols
USIC1_CH0.DX4B	I	0	
USIC1_CH0.DX4C	I	0	
USIC1_CH0.DX4D	I	0	
USIC1_CH0.DX4E	I	0	
USIC1_CH0.DX4F	I	0	
USIC1_CH0.DX4G	I	USIC1_CH0.DOUT2	Loop back shift data input
USIC1_CH0.HWIN2	I	P0.3	HW controlled shift data input
Data Inputs (DX5)			
USIC1_CH0.DX5A	I	0	Shift data input; used for: <ul style="list-style-type: none"> • Quad SSC MTSR2/MRST2 • Can be ignored for all other protocols
USIC1_CH0.DX5B	I	0	
USIC1_CH0.DX5C	I	0	
USIC1_CH0.DX5D	I	0	
USIC1_CH0.DX5E	I	0	
USIC1_CH0.DX5F	I	0	
USIC1_CH0.DX5G	I	USIC1_CH0.DOUT3	Loop back shift data input
USIC1_CH0.HWIN3	I	P0.2	HW controlled shift data input
Data Outputs			
USIC1_CH0.DOUT0	O	P0.5 P2.14 P0.5.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • ASC TXD • SSC MTSR/MRST • IIC SDA • IIS DOUT
USIC1_CH0.DOUT1	O	P0.4.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • Dual and Quad SSC MTSR1/MRST1 • Can be ignored for all other protocols
USIC1_CH0.DOUT2	O	P0.3.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • Quad SSC MTSR2/MRST2 • Can be ignored for all other protocols

Universal Serial Interface Channel (USIC)

Table 17-26 USIC Module 1 Channel 0 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC1_CH0.DOUT3	O	P0.2.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • Quad SSC MTSR3/MRST3 • Can be ignored for all other protocols
Clock Outputs			
USIC1_CH0.MCLKOUT	O	P5.10	Master clock output (optional for all protocols)
USIC1_CH0.SCLKOUT	O	P0.11 P5.8	Shift clock output; used for: <ul style="list-style-type: none"> • Master SCLKOUT in SSC and IIS • IIC SCL • Can be ignored in ASC
Control Outputs			
USIC1_CH0.SELO0	O	P0.6 P5.9	Shift control output; used for: <ul style="list-style-type: none"> • SSC Master SELO • IIS WA • Can be ignored for all other protocols
USIC1_CH0.SELO1	O	P0.14 P5.11	
USIC1_CH0.SELO2	O	P0.15	
USIC1_CH0.SELO3	O	P3.14	
USIC1_CH0.SELO4	O	not connected	
USIC1_CH0.SELO5	O	not connected	
USIC1_CH0.SELO6	O	not connected	
USIC1_CH0.SELO7	O	not connected	
System Related Outputs			
USIC1_CH0.DX0INS	O	USIC1_CH0.DX1F	Selected DX0 input signal
USIC1_CH0.DX1INS	O	DAC.TRIGGER[7]	Selected DX1 input signal
USIC1_CH0.DX2INS	O	CCU40.IN3L CCU42.IN2L CCU43.IN2L	Selected DX2 input signal
USIC1_CH0.DX3INS	O	not connected	Selected DX3 input signal
USIC1_CH0.DX4INS	O	not connected	Selected DX4 input signal
USIC1_CH0.DX5INS	O	not connected	Selected DX5 input signal

Universal Serial Interface Channel (USIC)

Table 17-27 USIC Module 1 Channel 1 Interconnects

Input/Output	I/O	Connected To	Description
Data Inputs (DX0)			
USIC1_CH1.DX0A	I	P3.15	Shift data input; used for: <ul style="list-style-type: none"> • ASC RXD • SSC MTSR/MRST • IIC SDA • IIS DIN
USIC1_CH1.DX0B	I	P3.14	
USIC1_CH1.DX0C	I	P4.2	
USIC1_CH1.DX0D	I	P0.0	
USIC1_CH1.DX0E	I	CAN1INS	
USIC1_CH1.DX0F	I	XTAL1	
USIC1_CH1.DX0G	I	USIC1_CH1.DOUT0	Loop back shift data input
USIC1_CH1.HWIN0	I	P3.15	HW controlled shift data input
Clock Inputs			
USIC1_CH1.DX1A	I	P0.10	Shift clock input; used for: <ul style="list-style-type: none"> • SSC Slave SCLKIN • IIC SCL • IIS Slave SCLKIN • Optional for ASC, SSC Master and IIS Master
USIC1_CH1.DX1B	I	P0.13	
USIC1_CH1.DX1C	I	P4.0	
USIC1_CH1.DX1D	I	0	
USIC1_CH1.DX1E	I	0	
USIC1_CH1.DX1F	I	USIC1_CH1.DX0INS	
USIC1_CH1.DX1G	I	USIC1_CH1.SCLKOUT	Loop back shift clock input
Control Inputs			
USIC1_CH1.DX2A	I	P0.9	Shift control input; used for: <ul style="list-style-type: none"> • SSC Slave SELIN • IIS Slave WAIN • Optional for ASC, SSC Master, IIC and IIS Master
USIC1_CH1.DX2B	I	P0.12	
USIC1_CH1.DX2C	I	0	
USIC1_CH1.DX2D	I	0	
USIC1_CH1.DX2E	I	CCU43.SR1	
USIC1_CH1.DX2F	I	CCU81.SR1	
USIC1_CH1.DX2G	I	USIC1_CH1.SELO0	Loop back shift control input
Data Inputs (DX3)			

Universal Serial Interface Channel (USIC)

Table 17-27 USIC Module 1 Channel 1 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC1_CH1.DX3A	I	0	Shift data input; used for: <ul style="list-style-type: none"> • Dual and Quad SSC MTSR1/MRST1 • Can be ignored for all other protocols
USIC1_CH1.DX3B	I	0	
USIC1_CH1.DX3C	I	0	
USIC1_CH1.DX3D	I	0	
USIC1_CH1.DX3E	I	0	
USIC1_CH1.DX3F	I	0	
USIC1_CH1.DX3G	I	USIC1_CH1.DOUT1	Loop back shift data input
USIC1_CH1.HWIN1	I	P3.14	HW controlled shift data input
Data Inputs (DX4)			
USIC1_CH1.DX4A	I	0	Shift data input; used for: <ul style="list-style-type: none"> • Quad SSC MTSR2/MRST2 • Can be ignored for all other protocols
USIC1_CH1.DX4B	I	0	
USIC1_CH1.DX4C	I	0	
USIC1_CH1.DX4D	I	0	
USIC1_CH1.DX4E	I	0	
USIC1_CH1.DX4F	I	0	
USIC1_CH1.DX4G	I	USIC1_CH1.DOUT2	Loop back shift data input
USIC1_CH1.HWIN2	I	P0.15	HW controlled shift data input
Data Inputs (DX5)			
USIC1_CH1.DX5A	I	0	Shift data input; used for: <ul style="list-style-type: none"> • Quad SSC MTSR3/MRST3 • Can be ignored for all other protocols
USIC1_CH1.DX5B	I	0	
USIC1_CH1.DX5C	I	0	
USIC1_CH1.DX5D	I	0	
USIC1_CH1.DX5E	I	0	
USIC1_CH1.DX5F	I	0	
USIC1_CH1.DX5G	I	USIC1_CH1.DOUT3	Loop back shift data input
USIC1_CH1.HWIN3	I	P0.14	HW controlled shift data input
Data Outputs			

Universal Serial Interface Channel (USIC)

Table 17-27 USIC Module 1 Channel 1 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC1_CH1.DOUT0	O	P0.1 P3.15 P4.2 P3.15.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • ASC TXD • SSC MTSR/MRST • IIC SDA • IIS DOUT
USIC1_CH1.DOUT1	O	P3.14.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • Dual and Quad SSC MTSR1/MRST1 • Can be ignored for all other protocols
USIC1_CH1.DOUT2	O	P0.15.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • Quad SSC MTSR2/MRST2 • Can be ignored for all other protocols
USIC1_CH1.DOUT3	O	P0.14.HW0_OUT	Shift data output; used for: <ul style="list-style-type: none"> • Quad SSC MTSR3/MRST3 • Can be ignored for all other protocols

Clock Outputs

USIC1_CH1.MCLKOUT	O	P4.1	Master clock output (optional for all protocols)
USIC1_CH1.SCLKOUT	O	P0.10 P0.13	Shift clock output; used for: <ul style="list-style-type: none"> • Master SCLKOUT in SSC and IIS • IIC SCL • Can be ignored in ASC

Control Outputs

Universal Serial Interface Channel (USIC)

Table 17-27 USIC Module 1 Channel 1 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC1_CH1.SELO0	O	P0.9 P0.12	Shift control output; used for: <ul style="list-style-type: none"> • SSC Master SELO • IIS WA • Can be ignored for all other protocols
USIC1_CH1.SELO1	O	P0.2 P3.3	
USIC1_CH1.SELO2	O	P3.4	
USIC1_CH1.SELO3	O	P3.5	
USIC1_CH1.SELO4	O	P3.6	
USIC1_CH1.SELO5	O	not connected	
USIC1_CH1.SELO6	O	not connected	
USIC1_CH1.SELO7	O	not connected	

System Related Outputs

USIC1_CH1.DX0INS	O	USIC1_CH1.DX1F	Selected DX0 input signal
USIC1_CH1.DX1INS	O	not connected	Selected DX1 input signal
USIC1_CH1.DX2INS	O	CCU42.IN3L CCU43.IN3L	Selected DX2 input signal
USIC1_CH1.DX3INS	O	not connected	Selected DX3 input signal
USIC1_CH1.DX4INS	O	not connected	Selected DX4 input signal
USIC1_CH1.DX5INS	O	not connected	Selected DX5 input signal

Table 17-28 USIC Module 1 Module Interconnects

Input/Output	I/O	Connected To	Description
USIC1_SR[1:0]	O	NVIC GPDMA	interrupt output lines (service requests SRx)
USIC1_SR[5:2]	O	NVIC	interrupt output lines (service requests SRx)

18 Controller Area Network Controller (MultiCAN+)

This chapter describes the MultiCAN+ controller of the XMC4300. It contains the following sections:

- CAN basics (see [Page 18-2](#))
- Overview of the CAN Module in the XMC4300 (see [Page 18-10](#))
- Functional description of the MultiCAN+ Kernel (see [Page 18-13](#))
- MultiCAN+ Kernel register description (see [Page 18-57](#))
- XMC4300 implementation-specific details (port connections and control, interrupt control, address decoding, clock control, see [Page 18-113](#)).

Table 18-1 Fixed Module Constants

Constant	Description
n_objects	Number of Message Objects available.
n_interrupts	Number of Interrupt Output Lines available.
n_pendings n_pendingregs	Number of Message Pending Bits available. There are n_pendings/32 message pending registers.
n_lists	Number of Lists available for allocation of Message Objects.
n_nodes	Number of CAN Nodes available As each CAN node has its own list in addition to the list of un-allocated elements, the relation n_nodes < n_lists is true.

18.1 CAN Basics

CAN is an asynchronous serial bus system with one logical bus line. It has an open, linear bus structure with equal bus participants called nodes. A CAN bus consists of two or more nodes.

The bus logic corresponds to a “wired-AND” mechanism. Recessive bits (equivalent to the logic 1 level) are overwritten by dominant bits (logic 0 level). As long as no bus node is sending a dominant bit, the bus is in the recessive state. In this state, a dominant bit from any bus node generates a dominant bus state. The maximum CAN bus speed is, by definition, 1 Mbit/s. This speed limits the CAN bus to a length of up to 40 m. For bus lengths longer than 40 m, the bus speed must be reduced.

The binary data of a CAN frame is coded in NRZ code (Non-Return-to-Zero). To ensure re-synchronization of all bus nodes, bit stuffing is used. This means that during the transmission of a message, a maximum of five consecutive bits can have the same polarity. Whenever five consecutive bits of the same polarity have been transmitted, the transmitter will insert one additional bit (stuff bit) of the opposite polarity into the bit stream before transmitting further bits. The receiver also checks the number of bits with the same polarity and removes the stuff bits from the bit stream (= destuffing).

18.1.1 Addressing and Bus Arbitration

In the CAN protocol, address information is defined in the identifier field of a message. The identifier indicates the contents of the message and its priority. The lower the binary value of the identifier, the higher is the priority of the message.

For bus arbitration, CSMA/CD with NDA (Carrier Sense Multiple Access/Collision Detection with Non-Destructive Arbitration) is used. If bus node A attempts to transmit a message across the network, it first checks that the bus is in the idle state (“Carrier Sense”) i.e. no node is currently transmitting. If this is the case (and no other node wishes to start a transmission at the same moment), node A becomes the bus master and sends its message. All other nodes switch to receive mode during the first transmitted bit (Start-Of-Frame bit). After correct reception of the message (acknowledged by each node), each bus node checks the message identifier and stores the message, if required. Otherwise, the message is discarded.

If two or more bus nodes start their transmission at the same time (“Multiple Access”), bus collision of the messages is avoided by bit-wise arbitration (“Collision Detection / Non-Destructive Arbitration” together with the “Wired-AND” mechanism, dominant bits override recessive bits). Each node that sends also reads back the bus level. When a recessive bit is sent but a dominant one is read back, bus arbitration is lost and the transmitting node switches to receive mode. This condition occurs for example when the message identifier of a competing node has a lower binary value and therefore sends a message with a higher priority. In this way, the bus node with the highest priority message wins arbitration without losing time by having to repeat the message. Other nodes that lost arbitration will automatically try to repeat their transmission once the bus

Controller Area Network Controller (MultiCAN+)

returns to idle state. Therefore, the same identifier can be sent in a Data Frame only by one node in the system. There must not be more than one node programmed to send Data Frames with the same identifier.

Standard message identifier has a length of 11 bits. CAN specification 2.0B extended the message identifier lengths to 29 bits, i.e. the extended identifier. Both frame formats are part of the ISO 11898-1. The identifier is available for Classical CAN.

18.1.2 CAN Frame Types

There are three types of CAN frames:

- Data Frames
- Remote Frames
- Error Frames

A Data Frame for Classical CAN contains a Data Field of 0 to 8 bytes in length. A Remote Frame contains no Data Field and is typically generated as a request for data (e.g. from a sensor). Data and Remote Frames can use an 11-bit “Standard” identifier or a 29-bit “Extended” identifier. An Error Frame can be generated by any node that detects a CAN bus error.

18.1.2.1 Data Frames

There are two types of Data Frames defined (see [Figure 18-1](#)):

- 11bit ID Data Frame
- 29bit ID Data Frame

11-bit Data Frame (Classical CAN Format)

A Data Frame begins with the Start-Of-Frame bit (SOF = dominant level) for hard synchronization of all nodes. The SOF is followed by the Arbitration Field consisting of 12 bits, the 11-bit identifier (reflecting the contents and priority of the message), and the RTR (Remote Transmission Request for Classical CAN) bit. With RTR at dominant level, the frame is marked as Data Frame. With RTR at recessive level, the frame is defined as a Remote Frame.

The next field is the Control Field consisting of 6 bits. The first bit of this field is the IDE (Identifier Extension) bit and is at dominant level for the Standard Data Frame. The following bit is reserved and defined as a dominant bit. The remaining 4 bits of the Control Field are the Data Length Code (DLC) that specifies the number of bytes in the Data Field. The Data Field can be 0 to 8 bytes wide. The Cyclic Redundancy (CRC) Field that follows the data bytes is used to detect possible transmission errors. It consists of a 15-bit CRC sequence completed by a recessive CRC delimiter bit.

The final field is the Acknowledge Field. During the ACK Slot, the transmitting node sends out a recessive bit. Any node that has received an error free frame acknowledges

Controller Area Network Controller (MultiCAN+)

the correct reception of the frame by sending back a dominant bit, regardless of whether or not the node is configured to accept that specific message. This behavior assigns the CAN protocol to the “in-bit-response” group of protocols. The recessive ACK delimiter bit, which must not be overwritten by a dominant bit, completes the Acknowledge Field.

Seven recessive End-of-Frame (EOF) bits finish the Data Frame. Between any two consecutive frames, the bus must remain in the recessive state for at least 3 bit times (called Inter Frame Space). If after the Inter Frame Space, no other nodes attempt to transmit the bus remains in idle state with a recessive level.

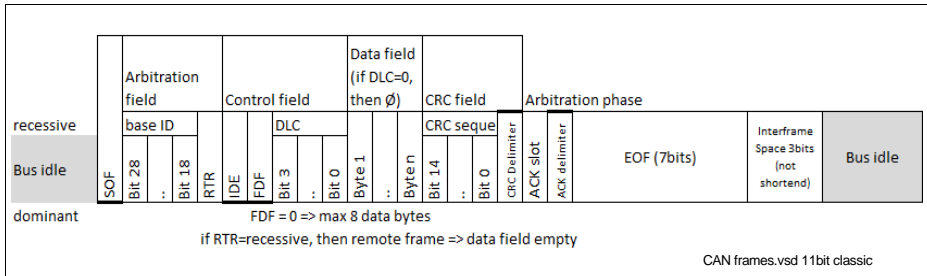


Figure 18-1 Classical 11-bit ID CAN Data Frame

Extended Data Frame (Classical CAN Format)

In the Extended CAN Data Frame, the message identifier of the standard frame has been extended to 29-bit. A split of the extended identifier into two parts, an 11-bit least significant section (as in classical CAN frame) and an 18-bit most significant section, ensures that the Identifier Extension bit (IDE) can remain at the same bit position in both standard and extended frames.

In the Extended CAN Data Frame, the SOF bit is followed by the 32-bit Arbitration Field. The first 11 bits are the least significant bits of the 29-bit Identifier (“Base-ID”). These 11 bits are followed by the recessive Substitute Remote Request (SRR) bit. The SRR is further followed by the recessive IDE bit, which indicates the frame to be an Extended CAN frame. If arbitration remains unresolved after transmission of the first 11 bits of the identifier, and if one of the nodes involved in arbitration is sending a classical CAN frame, then the CAN frame will win arbitration due to the assertion of its dominant IDE bit. Therefore, the SRR bit in an Extended CAN frame is recessive to allow the assertion of a dominant RTR bit by a node that is sending a CAN Remote Frame. The SRR and IDE bits are followed by the remaining 18 bits of the extended identifier and the RTR bit.

Control field and frame termination is identical to the Classical Data Frame.

Controller Area Network Controller (MultiCAN+)

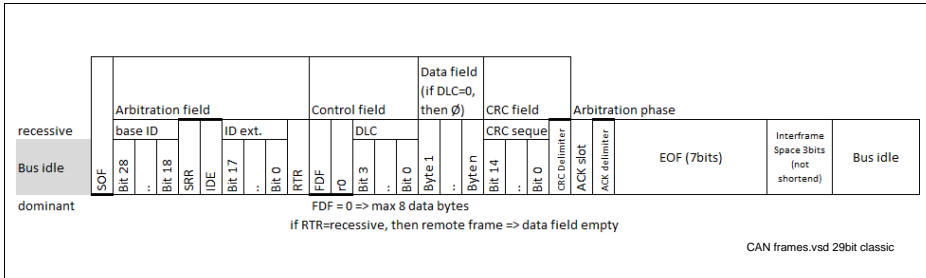


Figure 18-2 29 bit ID CAN Data Frame

18.1.2.2 Remote Frames

Normally, data transmission is performed on an autonomous basis with the data source node (e.g. a sensor) sending out a Data Frame. It is also possible, however, for a destination node (or nodes) to request the data from the source. For this purpose, the destination node sends a Remote Frame with an identifier that matches the identifier of the required Data Frame. The appropriate data source node will then send a Data Frame as a response to this remote request.

There are 2 differences between a Remote Frame and a Data Frame.

- The RTR bit is in the recessive state in a Remote Frame.
- There is no Data Field in a Remote Frame.

If a Data Frame and a Remote Frame with the same identifier are transmitted at the same time, the Data Frame wins arbitration due to the dominant RTR bit following the identifier. In this way, the node that transmitted the Remote Frame receives the requested data immediately.

18.1.2.3 Error Frames

An Error Frame is generated by any node that detects a bus error. An Error Frame consists of two fields, an Error Flag field followed by an Error Delimiter field. The Error Delimiter Field consists of 8 recessive bits and allows the bus nodes to restart bus communications after an error. There are, however, two forms of Error Flag fields. The form of the Error Flag field depends on the error status of the node that detects the error.

When an error-active node detects a bus error, the node generates an Error Frame with an active-error flag. The error-active flag is composed of six consecutive dominant bits that actively violate the bit-stuffing rule. All other stations recognize a bit-stuffing error and generate Error Frames themselves. The resulting Error Flag field on the CAN bus therefore consists of six to twelve consecutive dominant bits (generated by one or more nodes). The Error Delimiter field completes the Error Frame. After completion of the

Controller Area Network Controller (MultiCAN+)

Error Frame, bus activity returns to normal and the interrupted node attempts to re-send the aborted message.

If an error-passive node detects a bus error, the node transmits an error-passive flag followed, again, by the Error Delimiter field. The error-passive flag consists of six consecutive recessive bits, and therefore the Error Frame (for an error-passive node) consists of 14 recessive bits (i.e. no dominant bits). Therefore, the transmission of an Error Frame by an error-passive node will not affect any other node on the network, unless the bus error is detected by the node that is actually transmitting (i.e. the bus master). If the bus master node generates an error-passive flag, this may cause other nodes to generate Error Frames due to the resulting bit-stuffing violation. After transmission of an Error Frame an error-passive node must wait for 6 consecutive recessive bits on the bus before attempting to rejoin bus communications.

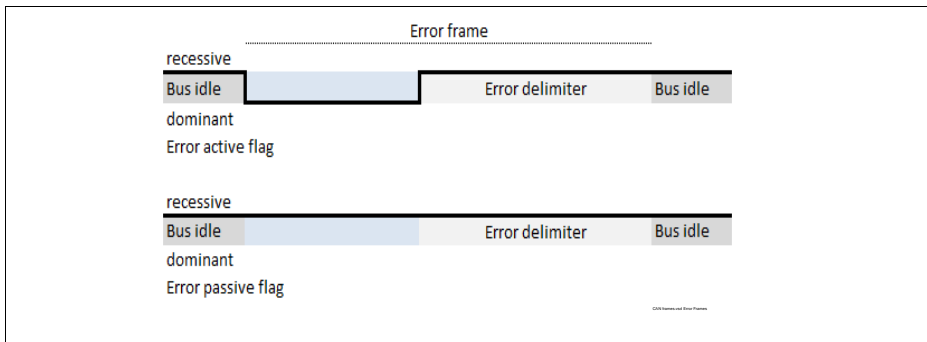


Figure 18-3 CAN Error Frames

18.1.3 The Nominal Bit Time

One bit cell (this means one high or low pulse of the NRZ code) is composed by four segments. Each segment is an integer multiple of Time Quanta t_Q . The Time Quanta is the smallest discrete timing resolution used by a CAN node. The nominal bit time definition with its segments is shown in [Figure 18-4](#).

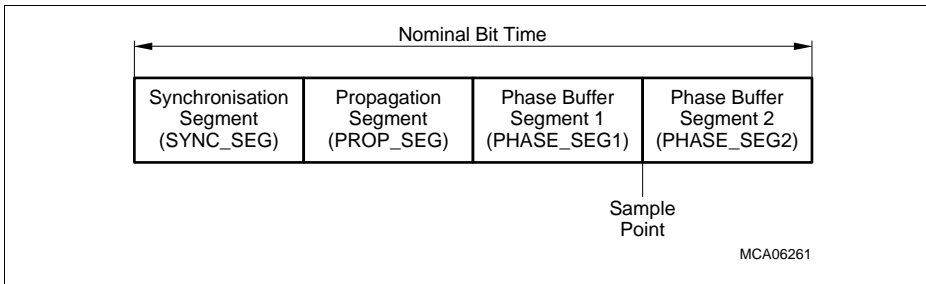


Figure 18-4 Partition of Nominal Bit Time

The Synchronization Segment (SYNC_SEG) is used to synchronize the various bus nodes. If there is a bit state change between the previous bit and the current bit, then the bus state change is expected to occur within this segment. The length of this segment is always $1 t_Q$.

The Propagation Segment (PROP_SEG) is used to compensate for signal delays across the network. These delays are caused by signal propagation delay on the bus line and through the electronic interface circuits of the bus nodes.

The Phase Segments 1 and 2 (PHASE_SEG1, PHASE_SEG2) are used to compensate for edge phase errors. These segments can be lengthened or shortened by re-synchronization. PHASE_SEG2 is reserved for calculation of the subsequent bit level, and is $\geq 2 t_Q$. At the sample point, the bus level is read and interpreted as the value of the bit cell. It occurs at the end of PHASE_SEG1.

The total number of t_Q in a bit time is between 8 and 25.

As a result of re-synchronization, PHASE_SEG1 can be lengthened or PHASE_SEG2 can be shortened. The amount of lengthening or shortening the phase buffer segments has an upper limit given by the re-synchronization jump width. The re-synchronization jump width may be between 1 and $4 t_Q$, but it may not be longer than PHASE_SEG1.

18.1.4 Error Detection and Error Handling

The CAN protocol has sophisticated error detection mechanisms. The following errors can be detected:

- **Cyclic Redundancy Check (CRC) Error**

With the CRC, the transmitter calculates special check bits for the bit sequence from the start of a frame until the end of the Data Field. This CRC sequence is transmitted in the CRC Field. The receiving node also calculates the CRC sequence using the same formula, and performs a comparison to the received sequence. If a mismatch is detected, a CRC error has occurred and an Error Frame is generated. The message is repeated.

Controller Area Network Controller (MultiCAN+)

- **Acknowledge Error**
In the Acknowledge Field of a message, the transmitter checks whether a dominant bit is read during the Acknowledge Slot (that is sent out as a recessive bit). If not, no other node has received the frame correctly, an Acknowledge Error has occurred, and the message must be repeated. No Error Frame is generated.
- **Form Error**
If a transmitter detects a dominant bit in one of the four segments End of Frame, Interframe Space, Acknowledge Delimiter, or CRC Delimiter, a Form Error has occurred, and an Error Frame is generated. The message is repeated.
- **Bit Error**
A Bit Error occurs if a) a transmitter sends a dominant bit and detects a recessive bit or b) if the transmitter sends a recessive bit and detects a dominant bit when monitoring the actual bus level and comparing it to the just transmitted bit. In case b), no error occurs during the Arbitration Field (ID, RTR, IDE) and the Acknowledge Slot.
- **Stuff Error**
If between Start of Frame and CRC Delimiter, six consecutive bits with the same polarity are detected, the bit-stuffing rule has been violated. A stuff error occurs and an Error Frame is generated. The message is repeated.

Detected errors are made public to all other nodes via Error Frames (except Acknowledge Errors). The transmission of the erroneous message is aborted and the frame is repeated as soon as possible. Furthermore, each CAN node is in one of the three error states (error-active, error-passive or bus-off) according to the value of the internal error counters. The error-active state is the usual state where the bus node can transmit messages and active-error frames (made of dominant bits) without any restrictions. In the error-passive state, messages and passive-error frames (made of recessive bits) may be transmitted. The bus-off state makes it temporarily impossible for the node to participate in the bus communication. During this state, messages can be neither received nor transmitted.

Basic CAN, Full CAN

There is one more CAN characteristic that is related to the interface of a CAN module (controller) and the host CPU: Basic-CAN and Full-CAN functionality.

In Basic-CAN devices, only basic functions of the protocol are implemented in hardware, such as the generation and the check of the bit stream. The decision, whether a received message has to be stored or not (acceptance filtering), and the complete message management must be done by software.

Full-CAN devices (this is the case for the MultiCAN+ controller as implemented in XMC4300) manage the whole bus protocol in hardware, including the acceptance filtering and message management. Full-CAN devices contain message objects that handle autonomously the identifier, the data, the direction (receive or transmit) and the information of CAN operation. During the initialization of the device, the host CPU

Controller Area Network Controller (MultiCAN+)

determines which messages are to be sent and which are to be received. The host CPU is informed by interrupt if the identifier of a received message matches with one of the programmed (receive-) message objects. The CPU load of Full-CAN devices is greatly reduced. When using Full-CAN devices, high baud rates and high bus loads with many messages can be handled.

Normally, the CAN device also provides only one transmit buffer and one or two receive buffers. Therefore, the host CPU load is quite high when using Basic-CAN modules. The main advantage of Basic-CAN is a reduced chip size leading to low costs of these devices.

Controller Area Network Controller (MultiCAN+)

18.2 Overview

The MultiCAN+ module provides a communication interface which is fully compliant with CAN specification V2.0B (active), providing communications at up to 1 Mbit/s in Classical CAN (ISO 11898-1:2003(E) mode).

The MultiCAN+ module for the XMC4300 consists of 1 module (i.e MultiCAN with 2 CAN nodes), representing 2 serial communication interfaces. Each CAN node communicates over two pins (TXD and RXD). The device ports which are used for TXD and RXD may be individually configured within the PORTS block. Several port configuration options are available to provide application-specific flexibility.

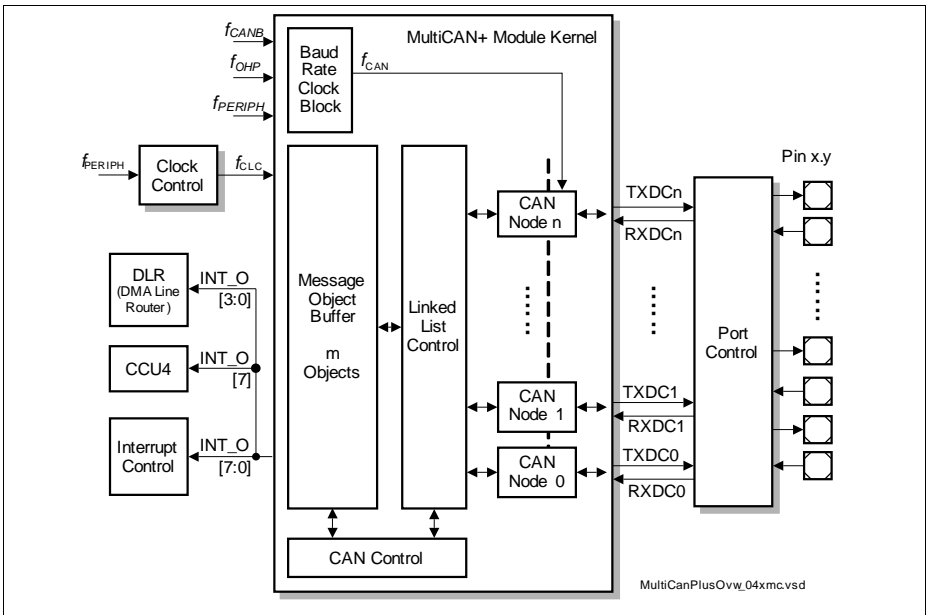


Figure 18-5 Overview of the MultiCAN+ Module. The module has 2 nodes and 64 objects. XMC4300

The MultiCAN+ module contains 2 independently operating CAN nodes with Full-CAN functionality that are able to exchange Data and Remote Frames via a gateway function. Each CAN node can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers.

All CAN nodes share a common set of 64 message objects. Each message object can be individually allocated to one of the CAN nodes. Besides serving as a storage container for incoming and outgoing frames, message objects can be combined to build gateways between the CAN nodes or to setup a FIFO buffer.

Controller Area Network Controller (MultiCAN+)

The message objects are organized in double-chained linked lists, where each CAN node has its own list of message objects. A CAN node stores frames only into message objects that are allocated to the message object list of the CAN node, and it transmits only messages belonging to this message object list. A powerful, command-driven list controller performs all message object list operations.

The bit timings for the CAN nodes are derived from the module timer clock (f_{CAN}) and are programmable up to a data rate of 1 Mbit/s. External bus transceivers are connected to a CAN node via a pair of receive and transmit pins.

18.2.1 Features List

The MultiCAN+ module provides the following features:

- Compliant with ISO 11898 and SAE J 1939
- CAN functionality according to CAN specification V2.0 B active
- Dedicated control registers for each CAN node
- Data transfer rates up to 1 Mbit/s
- Support for asynchronous clock sources for baud rate generation by providing separate frequency domain and input:
 - System frequency clock f_{CLC}
 - Direct oscillator clock (e.g. from ceramic resonator)
- Flexible and powerful message transfer control and error handling capabilities
- Advanced CAN bus bit timing analysis and baud rate detection for each CAN node via a frame counter
- Full-CAN functionality: A set of 64 message objects can be individually
 - Allocated (assigned) to any CAN node
 - Configured as transmit or receive object
 - Setup to handle frames with 11-bit or 29-bit identifier
 - Identified by a timestamp via a frame counter
 - Configured to remote monitoring mode
- Advanced Acceptance Filtering
 - Each message object provides an individual acceptance mask to filter incoming frames
 - A message object can be configured to accept standard or extended frames or to accept both standard and extended frames
 - Message objects can be grouped into different priority classes for transmission and reception
 - The selection of the message to be transmitted first can be based on frame identifier, IDE bit and RTR bit according to CAN arbitration rules, or on its order in the list
- Advanced CAN node features
 - Analyzer Mode supports monitoring of bus traffic without actively participating on the bus
 - Internal Loop-Back Mode is available for test purposes

Controller Area Network Controller (MultiCAN+)

- Data transmission from a node can be stopped without affecting reception
- Programmable minimum delay between two consecutive messages
- Advanced message object functionality
 - Message objects can be combined to build FIFO message buffers of arbitrary size, limited only by the total number of message objects
 - Message objects can be linked to form a gateway that automatically transfers frames between 2 different CAN buses. A single gateway can link any two CAN nodes. An arbitrary number of gateways can be defined
- Advanced data management
 - The message objects are organized in double-chained lists
 - List reorganizations can be performed at any time, even during full operation of the CAN nodes
 - A powerful, command-driven list controller manages the organization of the list structure and ensures consistency of the list
 - Message FIFOs are based on the list structure and can easily be scaled in size during CAN operation
- Advanced interrupt handling
 - Message interrupts, node interrupts can be generated
 - Interrupt requests can be routed individually to one of the 8 interrupt output lines
 - Message post-processing notifications can be combined flexibly into a dedicated register field of 256 notification bits

18.3 MultiCAN+ Kernel Functional Description

This section describes the functionality of the MultiCAN+ module.

18.3.1 Module Structure

Figure 18-6 shows the general structure of the MultiCAN+ module.

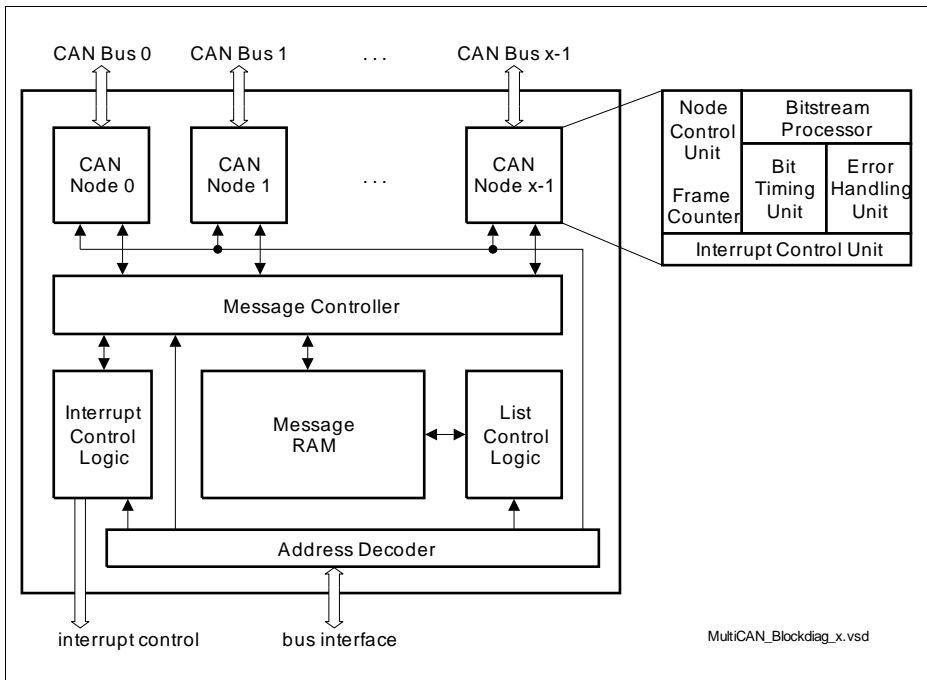


Figure 18-6 MultiCAN+ Block Diagram

CAN Nodes

Each CAN node consists of several sub-units.

- Bitstream Processor**
 The Bitstream Processor performs data, remote, error and overload frame processing according to the ISO 11898 standard. This includes conversion between the serial data stream and the input/output registers.
- Bit Timing Unit**
 The Bit Timing Unit determines the length of a bit time and the location of the sample point according to the user settings, taking into account propagation delays and phase shift errors. The Bit Timing Unit also performs resynchronization.

Controller Area Network Controller (MultiCAN+)

- **Error Handling Unit**

The Error Handling Unit manages the receive and transmit error counter. Depending on the contents of both counters, the CAN node is set into an error-active, error passive or bus-off state.

- **Node Control Unit**

The Node Control Unit coordinates the operation of the CAN node:

- Enable/disable CAN transfer of the node
- Enable/disable and generate node-specific events that lead to an interrupt request (CAN bus errors, successful frame transfers etc.)
- Administration of the frame counter

- **Interrupt Control Unit**

The Interrupt Control Unit in the CAN node controls the interrupt generation for the different conditions that can occur in the CAN node.

Message Controller

The Message Controller handles the exchange of CAN frames between the CAN nodes and the message objects that are stored in the Message RAM. The Message Controller performs several functions:

- Receive acceptance filtering to determine the correct message object for storing of a received CAN frame
- Transmit acceptance filtering to determine the message object to be transmitted first, individually for each CAN node
- Transfer contents between message objects and the CAN nodes, taking into account the status/control bits of the message objects
- Handling of the FIFO buffering and gateway functionality
- Aggregation of message-pending notification bits

List Controller

The List Controller performs all operations that lead to a modification of the double-chained message object lists. Only the list controller is allowed to modify the list structure. The allocation/deallocation or reallocation of a message object can be requested via a user command interface (command panel). The list controller state machine then performs the requested command autonomously.

Interrupt Control

The general interrupt structure is shown in [Figure 18-7](#). The interrupt event can trigger the interrupt generation. The interrupt pulse is generated independently of the interrupt flag in the interrupt status register. The interrupt flag can be reset by software by writing a 0 to it.

Controller Area Network Controller (MultiCAN+)

If enabled by the related interrupt enable bit in the interrupt enable register, an interrupt pulse can be generated at one of the 8 interrupt output lines INT_0m of the MultiCAN+ module. If more than one interrupt source is connected to the same interrupt node pointer (in the interrupt node pointer register), the requests are combined to one common line.

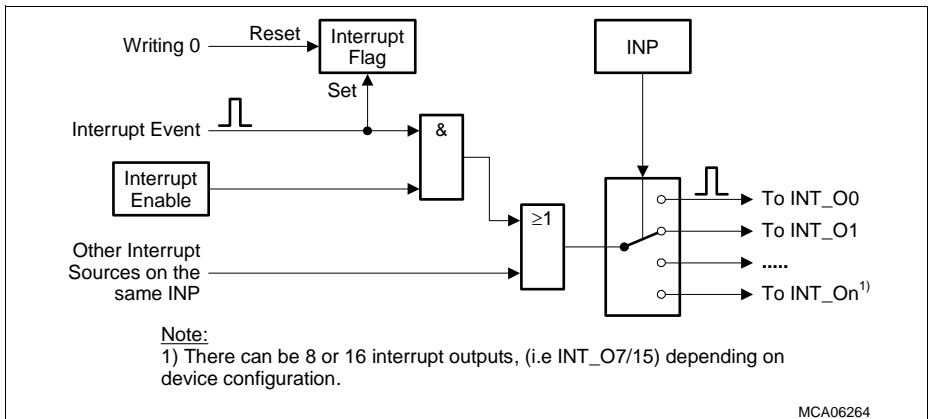


Figure 18-7 General Interrupt Structure

18.3.2 Clock Control

The CAN module timer clock f_{CAN} of the functional blocks of the MultiCAN+ module is derived from the synchronous clock source. The Fractional Divider is used to generate f_{CAN} used for bit timing calculation, The frequency of f_{CAN} is identical for all CAN nodes. The register file operates with the module control clock f_{CLC} . See also **“MultiCAN+ Clock Generation” on Page 18-16.**

The output clock f_{CAN} of the Fractional Divider is based on the clock f_A , but only every nth clock pulse is taken.

Controller Area Network Controller (MultiCAN+)

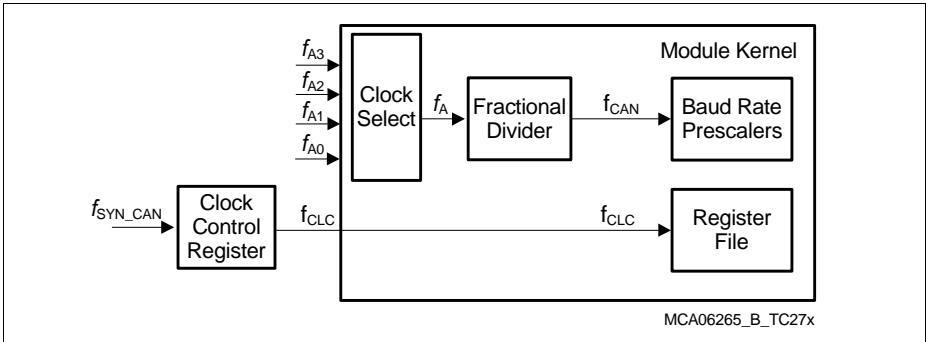


Figure 18-8 MultiCAN+ Clock Generation

The $f_{\text{SYN_CAN}}$ is identical to f_{PERIPH} . f_{Ai} is the asynchronous clock input.

Table 18-2 indicates the minimum operating frequencies in MHz for f_{CLC} that are required for a baud rate of 1 Mbit/s for the active CAN nodes. If a lower baud rate is desired, the values can be scaled linearly (e.g. for a maximum of 500 kbit/s, 50% of the indicated value are required).

The values imply that the CPU (or DMA) executes maximum accesses to the MultiCAN+ module. The values may contain rounding effects.

Controller Area Network Controller (MultiCAN+)

Table 18-2 Minimum Operating Frequencies¹⁾ [MHz]

Number of allocated message objects MO ²⁾ ,	Number of Active CAN Nodes				
	1	2	3	4	5
16 MO	12	19	26	33	40
32 MO	15	23	30	37	44
64 MO	21	28	37	46	53
128 MO	40	45	50	55	61
256 MO	72	77	82	88	93

1) In the case of 15 time quanta, the minimum operating frequency required is 15 MHz.

2) Only those message objects have to be taken into account that are allocated to a CAN node. The unallocated message objects have no influence on the minimum operating frequency.

The baud rate generation of the MultiCAN+ being based on f_A , this frequency has to be chosen carefully to allow correct CAN bit timing. The required value of f_A is given by an integer multiple (n) of the CAN baud rate multiplied by the number of time quanta per CAN bit time. For example, to reach 1 Mbit/s with 20 tq per bit time, possible values of f_A are given by formula $[n \times 20]$ MHz, with n being an integer value, starting at 1.

It is not advised to use fractional divider mode.

Additionally, for correct operation of the MultiCAN, the following conditions have to be fulfilled,

$$\text{Baudrate}_{\max} = [(8 \times T_{\text{CAN}}) + (8 \times T_{\text{CLC}}) + (4 \times \text{No. of active CAN nodes} \times T_{\text{CLC}})] \quad (18.1)$$

also
 $\text{NBTR.SJW} < \text{NBTR.TSEG1}$

As an example, when $f_{\text{CLC}} = 10$ MHz, $f_{\text{CAN}} = 20$ MHz, No of active CAN nodes =2,

$$\text{Baudrate}_{\max} = [(8 \times 50 \text{ ns}) + (8 \times 100 \text{ ns}) + (4 \times 2 \times 100 \text{ ns})] = 2000 \text{ ns} = 500 \text{ kBaud}$$

Table 18-3 below illustrates the minimum CAN module timer clock f_{CAN} and Module Control Clock f_{CLC} that's required to support a baudrate generation of 500 kBaud. If a higher baudrate is desired, the values need to be calculated as per **Equation (18.1)**.

Table 18-3 Minimum Operating Frequencies [MHz] required for 500kBaud

No. of active CAN nodes	$f_{\text{CAN}} = f_{\text{CLC}}$ (MHz)	$f_{\text{CAN}} \neq f_{\text{CLC}}$ (MHz)	
		f_{CAN}	f_{CLC}

Controller Area Network Controller (MultiCAN+)

Table 18-3 Minimum Operating Frequencies [MHz] required for 500kBaud

1	10	16	8
		20	8
		24	8
		80	7
2	12	16	11
		20	10
		24	10
		80	9

18.3.3 Port Input Control

It is possible to select the input lines for the RXDCx inputs for the CAN nodes. The selected input is connected to the CAN node and is also available to wake-up the system. More details are defined in [Section 18.6.4.2](#) on [Page 18-119](#).

18.3.4 CAN Node Control

Each CAN node may be configured and run independently of the other CAN node. Each CAN node is equipped with its own node control logic to configure the global behavior and to provide status information.

Note: In the following descriptions, index “x” stands for the node number and index “n” represents the message object number.

Configuration Mode is activated when bit NCRx.CCE is set to 1. This mode allows CAN bit timing parameters and the error counter registers to be modified.

CAN Analyzer Mode

CAN Analyzer Mode is activated when bit NCRx.CALM is set to 1. In this operation mode, Data And Remote Frames are monitored without active participation in any CAN transfer (CAN transmit pin is held on recessive level). Incoming Remote Frames are stored in a corresponding transmit message object, while arriving data frames are saved in a matching receive message object.

In CAN Analyzer Mode, the entire configuration information of the valid (including ACK) received frame is stored in the corresponding message object, and can be evaluated by the CPU to determine their identifier, IDE bit information and data length code (ID and DLC optionally if the Remote Monitoring Mode is active, bit MOFCRn.RMM = 1). Incoming frames are not acknowledged, and no Error Frames are generated. If CAN Analyzer Mode is enabled, Remote Frames are not responded to by the corresponding Data Frame, and Data Frames cannot be transmitted by setting the transmit request bit MOSTATn.TXRQ. Receive interrupts are generated in CAN Analyzer Mode (if enabled) for all error free received frames.

The node-specific interrupt configuration is also defined by the Node Control Logic via the NCRx register bits TRIE, ALIE and LECIE:

- If control bit TRIE is set to 1, a transfer interrupt is generated when the NSRx register has been updated (after each successfully completed message transfer).
- If control bit ALIE is set to 1, an alert interrupt is generated when a “bus-off” condition has been recognized or the Error Warning Level has been exceeded or under-run. Additionally, list or object errors lead to this type of interrupt.
- If control bit LECIE is set to 1, a last error code interrupt is generated when an error code > 0 is written into bit field NSRx.LEC by hardware.

Setting bit TXDIS in register NCRx stops the transmit activity of this node without affecting reception; bit CANDIS disables the node completely.

The Node x Status Register NSRx provides an overview about the current state of the respective CAN node x, comprising information about CAN transfers, CAN node status, and error conditions.

Controller Area Network Controller (MultiCAN+)

The CAN frame counter can be used to check the transfer sequence of message objects or to obtain information about the instant a frame has been transmitted or received from the associated CAN bus. CAN frame counting is performed by a 16-bit counter, controlled by register NFCRx. Bit fields NFCRx.CFMOD and NFCRx.CFSEL determine the operation mode and the trigger event incrementing the frame counter.

18.3.4.1 Bit Timing Unit

According to the ISO 11898 standard, a CAN bit time is subdivided into different segments (**Figure 18-9**). Each segment consists of multiples of a time quantum t_q . The magnitude of t_q is adjusted by Node x Bit Timing Register bit fields NBTRx.BRP and NBTRx.DIV8, both controlling the baud rate prescaler (register NBTRx is described on **Page 18-84**). The baud rate prescaler is driven by the module timer clock f_{CAN} (generation and control of f_{CAN} is described on **Page 18-115**).

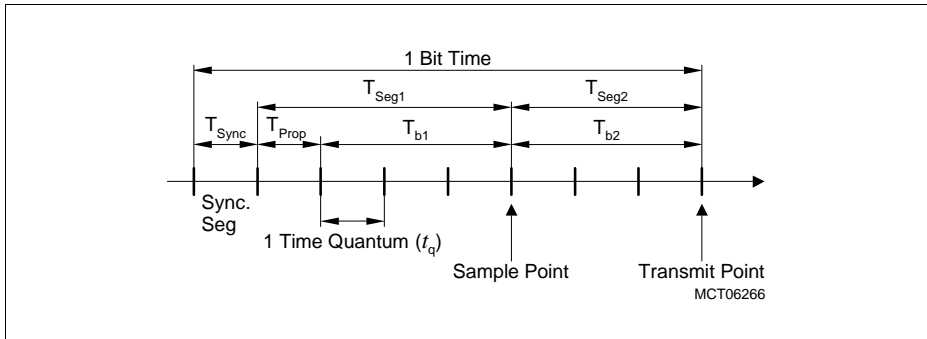


Figure 18-9 CAN Bus Bit Timing Standard

The Synchronization Segment (T_{Sync}) allows phase synchronization between transmitter and receiver time base. The Synchronization Segment length is always one t_q . The Propagation Time Segment (T_{Prop}) takes into account the physical propagation delay in the transmitter output driver on the CAN bus line and in the transceiver circuit. For a working collision detection mechanism, T_{Prop} must be two times the sum of all propagation delay quantities rounded up to a multiple of t_q . The phase buffer segments 1 and 2 (T_{b1} , T_{b2}) before and after the signal sample point are used to compensate for a mismatch between transmitter and receiver clock phases detected in the synchronization segment.

The maximum number of time quanta allowed for re-synchronization is defined by bit field NBTRx.SJW. The Propagation Time Segment and the Phase Buffer Segment 1 are combined to parameter T_{Seg1} , which is defined by the value NBTRx.TSEG1. A minimum of 3 time quanta is demanded by the ISO standard. Parameter T_{Seg2} , which is defined by the value of NBTRx.TSEG2, covers the Phase Buffer Segment 2. A minimum of 2 time

Controller Area Network Controller (MultiCAN+)

quanta is demanded by the ISO standard. According to ISO standard, a CAN bit time, calculated as the sum of T_{Sync} , T_{Seg1} and T_{Seg2} , must not fall below 8 time quanta.

Calculation of the bit time:

$$\begin{aligned}
 t_q &= (BRP + 1) / f_{CAN} && \text{if DIV8} = 0 \\
 &= 8 \times (BRP + 1) / f_{CAN} && \text{if DIV8} = 1 \\
 T_{Sync} &= 1 \times t_q \\
 T_{Seg1} &= (TSEG1 + 1) \times t_q && (\text{min. } 3 t_q) \\
 T_{Seg2} &= (TSEG2 + 1) \times t_q && (\text{min. } 2 t_q) \\
 \text{bit time} &= T_{Sync} + T_{Seg1} + T_{Seg2} && (\text{min. } 8 t_q)
 \end{aligned}$$

To compensate phase shifts between clocks of different CAN controllers, the CAN controller must synchronize on any edge from the recessive to the dominant bus level. The hard synchronization is enabled (at the start of frame), the bit time is restarted at the synchronization segment. Otherwise, the re-synchronization jump width T_{SJW} defines the maximum number of time quanta, a bit time may be shortened or lengthened by one re-synchronization. The value of SJW is defined by bit field NBTRx.SJW.

$$\begin{aligned}
 T_{SJW} &= (SJW + 1) \times t_q \\
 T_{Seg1} &\geq T_{SJW} + T_{prop} \\
 T_{Seg2} &\geq T_{SJW}
 \end{aligned}$$

The maximum relative tolerance for f_{CAN} depends on the Phase Buffer Segments, re-synchronization jump width and the bit time.

$$\begin{aligned}
 df_{CAN} &\leq \min(T_{b1}, T_{b2}) / [2 \times (13 \times \text{bit time} - T_{b2})] && \text{AND} \\
 df_{CAN} &\leq T_{SJW} / 20 \times \text{bit time}
 \end{aligned}$$

A valid CAN bit timing must be written to the CAN Node Bit Timing Register NBTR before resetting the INIT bit in the Node Control Register, i.e. before enabling the operation of the CAN node.

The Node Bit Timing Register may be written only if bit CCE (Configuration Change Enable) is set in the corresponding Node Control Register.

18.3.4.2 Bitstream Processor

Based on the message objects in the message buffer, the Bitstream Processor generates the remote and Data Frames to be transmitted via the CAN bus. It controls the

Controller Area Network Controller (MultiCAN+)

CRC generator and adds the checksum information to the new remote or Data Frame. After including the SOF bit and the EOF field, the Bitstream Processor starts the CAN bus arbitration procedure and continues with the frame transmission when the bus was found in idle state. While the data transmission is running, the Bitstream Processor continuously monitors the I/O line. If (outside the CAN bus arbitration phase or the acknowledge slot) a mismatch is detected between the voltage level on the I/O line and the logic state of the bit currently sent out by the transmit shift register, a CAN LEC error interrupt request is generated, and the error code is indicated by the Node x Status Register bit field NSRx.LEC.

The data consistency of an incoming frame is verified by checking the associated CRC field. When an error has been detected, a CAN LEC error interrupt request is generated and the associated error code is presented in the Node x Status Register NSRx. Furthermore, an Error Frame is generated and transmitted on the CAN bus. After decomposing a faultless frame into identifier and data portion, the received information is transferred to the message buffer executing remote and Data Frame handling, interrupt generation and status processing.

18.3.4.3 Error Handling Unit

The Error Handling Unit of a CAN node x is responsible for the fault confinement of the CAN device. Its two counters, the Receive Error Counter REC and the Transmit Error Counter TEC (bit fields of the Node x Error Counter Register NECNTx, see [Page 18-86](#)) are incremented and decremented by commands from the Bitstream Processor. If the Bitstream Processor itself detects an error while a transmit operation is running, the Transmit Error Counter is incremented by 8. An increment of 1 is used when the error condition was reported by an external CAN node via an Error Frame generation. For error analysis, the transfer direction of the disturbed message and the node that recognizes the transfer error are indicated for the respective CAN node x in register NECNTx. Depending on the values of the error counters, the CAN node is set into error-active, error-passive, or bus-off state.

The CAN node is in error-active state if both error counters are below the error-passive limit of 128. The CAN node is in error-passive state, if at least one of the error counters is equal to or greater than 128.

The bus-off state is activated if the Transmit Error Counter is equal to or greater than the bus-off limit of 256. This state is reported for CAN node x by the Node x Status Register flag NSRx.BOFF. The device remains in this state, until the “bus-off” recovery sequence is finished. Additionally, Node x Status Register flag NSRx.EWRN is set when at least one of the error counters is equal to or greater than the error warning limit defined by the Node x Error Count Register bit field NECNTx.EWRNLVL. Bit NSRx.EWRN is reset if both error counters fall below the error warning limit again (see [Page 18-75](#)).

18.3.4.4 CAN Frame Counter

Each CAN node is equipped with a frame counter that counts transmitted/received CAN frames or obtains information about the time when a frame has been started to transmit or be received by the CAN node. CAN frame counting/bit time counting is performed by a 16-bit counter that is controlled by Node x Frame Counter Register NFCRx (see [Page 18-87](#)). Bit field NFCRx.CFSEL determines the operation mode of the frame counter:

- **Frame Count Mode:**
After the successful transmission and/or reception of a CAN frame, the frame counter is copied into the CFCVAL bit field of the MOIPRn register of the message object involved in the transfer. Afterwards, the frame counter is incremented.
- **Time Stamp Mode:**
The frame counter is incremented (internally) with the beginning of a new bit time. Its value is permanently sampled in the CFC field while the bus is idle. The value sampled just before the SOF bit of a new frame is detected is written to the corresponding message object. When the treatment of a message object is finished, the sampling continues.
- **Bit Timing Mode:**
Used for baud rate detection and analysis of the bit timing ([Chapter 18.3.6.3](#)).
- **Error Count Mode:**
The frame counter is incremented when an error frame is received or an error is detected by the node (001_B to 110_B) (see [Table 18-9](#) for [Encoding of the LEC Bit field](#)). If the NFCRx.CFCIE interrupt bit is enabled, the NFCRx.CFCOV overflow flag will be set when the frame counter overflows. Configuration of CFSEL has no influence.

18.3.4.5 CAN Node Interrupts

Each CAN node has four hardware triggered interrupt request types that are able to generate an interrupt request upon:

- The successful transmission or reception of a frame
- A CAN protocol error with a last error code
- An alert condition: Transmit/receive error counters reach the warning limit, bus-off state changes, a List Length Error occurs, or a List Object Error occurs
- An overflow of the frame counter

Besides the hardware generated interrupts, software initiated interrupts can be generated using the Module Interrupt Trigger Register MITR. Writing a 1 to bit n of bit field MITR.IT generates an interrupt request signal on the corresponding interrupt output line INT_On. When writing MITR.IT more than one bit can be set resulting in activation of multiple INT_On interrupt output lines at the same time. See also [“Interrupt Control” on Page 18-122](#) for further processing of the CAN node interrupts.

Controller Area Network Controller (MultiCAN+)

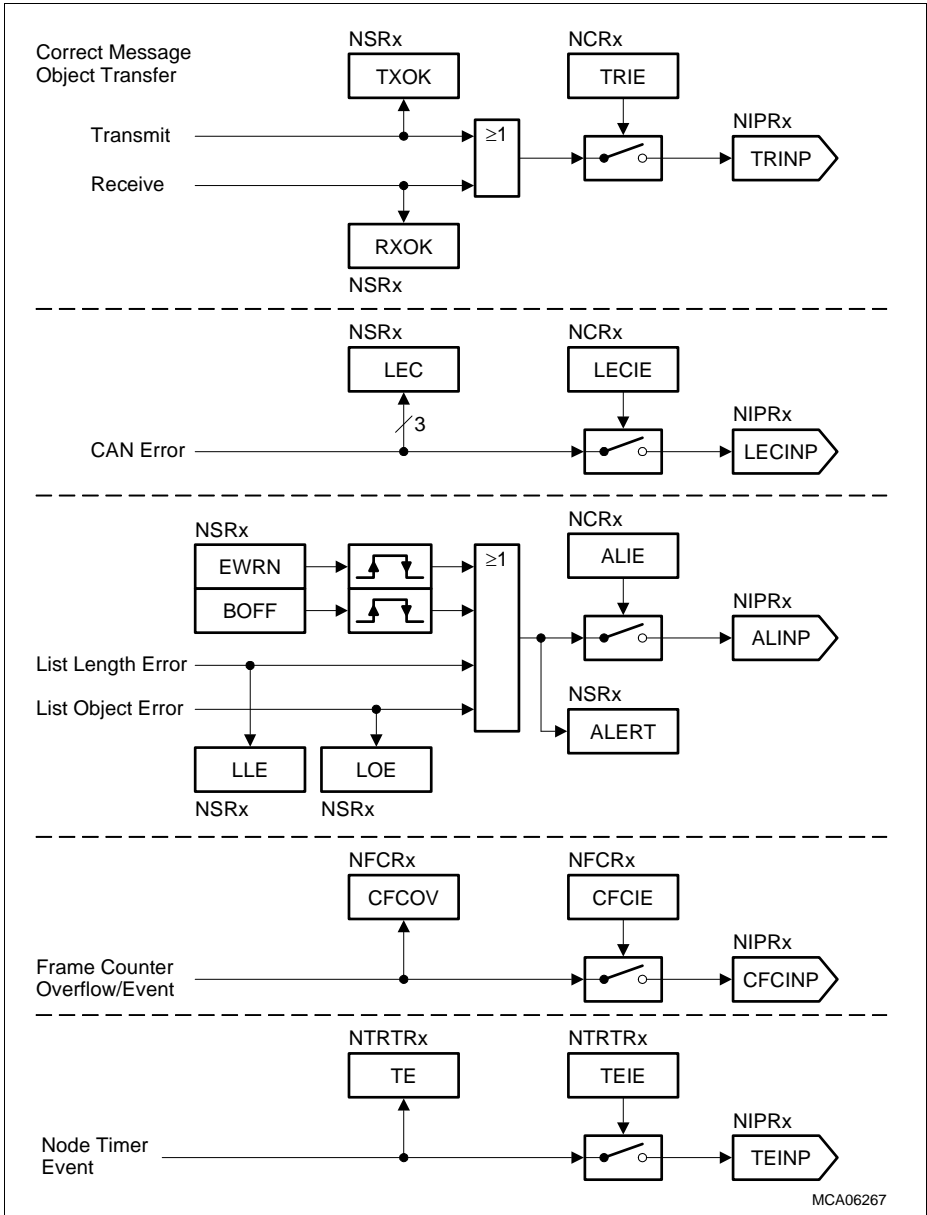


Figure 18-10 CAN Node Interrupts

18.3.5 Message Object List Structure

This section describes the structure of the message object lists in the MultiCAN+ module.

18.3.5.1 Basics

The message objects of the MultiCAN+ module are organized in double-chained lists, where each message object has a pointer to the previous message object in the list as well as a pointer to the next message object in the list. The MultiCAN+ module provides 16 lists. Each message object is allocated to one of these lists. In the example in [Figure 18-11](#), the three message objects (3, 5, and 16) are allocated to the list with index 2 (List Register LIST2).

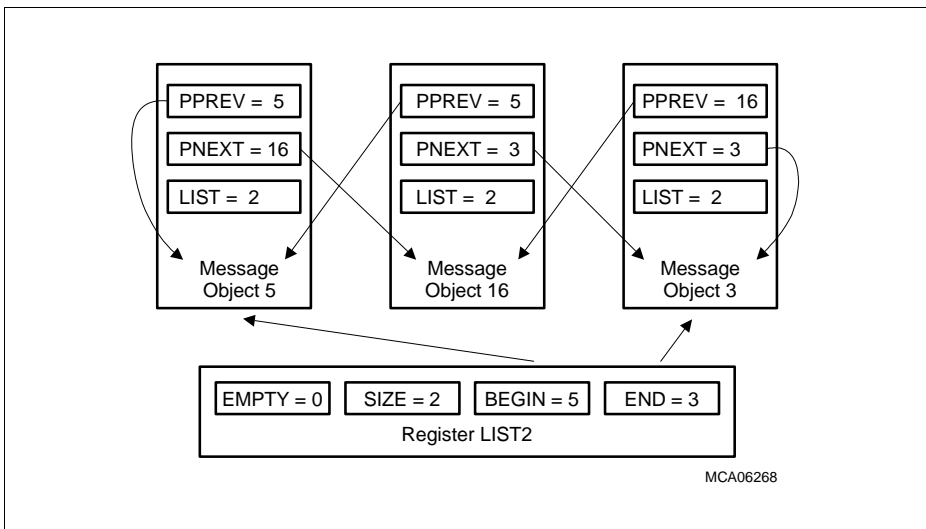


Figure 18-11 Example Allocation of Message Objects to a List

Bit field BEGIN in the List Register (for definition, see [Page 18-70](#)) points to the first element in the list (object 5 in the example), and bit field END points to the last element in the list (object 3 in the example). The number of elements in the list is indicated by bit field SIZE of the List Register (SIZE = number of list elements - 1, thus SIZE = 2 for the 3 elements in the example). The EMPTY bit of the List Register indicates whether or not a list is empty (EMPTY = 0 in the example, because list 2 is not empty).

Each message object n has a pointer PNEXT in its Message Object n Status Register MOSTATn (see [Page 18-95](#)) that points to the next message object in the list, and a pointer PPREV that points to the previous message object in the list. PPREV of the first message object points to the message object itself because the first message object has

Controller Area Network Controller (MultiCAN+)

no predecessor (in the example message object 5 is the first message object in the list, indicated by PPREV = 5). PNEXT of the last message object also points to the message object itself because the last message object has no successor (in the example, object 3 is the last message object in the list, indicated by PNEXT = 3).

Bit field MOSTATn.LIST indicates the list index number to which the message object is currently allocated to. The message objects of the example are allocated to list 2. Therefore, all LIST bit fields for the message objects assigned to list 2 are set to LIST = 2.

18.3.5.2 List of Unallocated Elements

The list with list index 0 has a special meaning: it is the list of all unallocated elements. An element is called unallocated if it belongs to list 0 (MOSTATn.LIST = 0). It is called allocated if it belongs to a list with an index not equal to 0 (MOSTATn.LIST > 0).

After reset, all message objects are unallocated. This means that they are assigned to the list of unallocated elements with MOSTATn.LIST = 0. After this initial allocation of the message objects caused by reset, the list of all unallocated message objects is ordered by message number (predecessor of message object n is object n-1, successor of object n is object n+1).

18.3.5.3 Connection to the CAN Nodes

Each CAN node is linked to one unique list of message objects. A CAN node performs message transfer only with the message objects that are allocated to the list of the CAN node. This is illustrated in [Figure 18-12](#). Frames that are received on a CAN node may only be stored in one of the message objects that belongs to the CAN node; frames to be transmitted on a CAN node are selected only from the message objects that are allocated to that node, as indicated by the vertical arrows.

There are more lists (16) than CAN nodes (2). This means that some lists are not linked to one of the CAN nodes. A message object that is allocated to one of these unlinked lists cannot receive messages directly from a CAN node and it may not transmit messages.

FIFO and gateway mechanisms refer to message numbers and not directly to a specific list. The user must take care that the message objects targeted by FIFO/gateway belong to the desired list. The mechanisms make it possible to work with lists that do not belong to the CAN node.

Controller Area Network Controller (MultiCAN+)

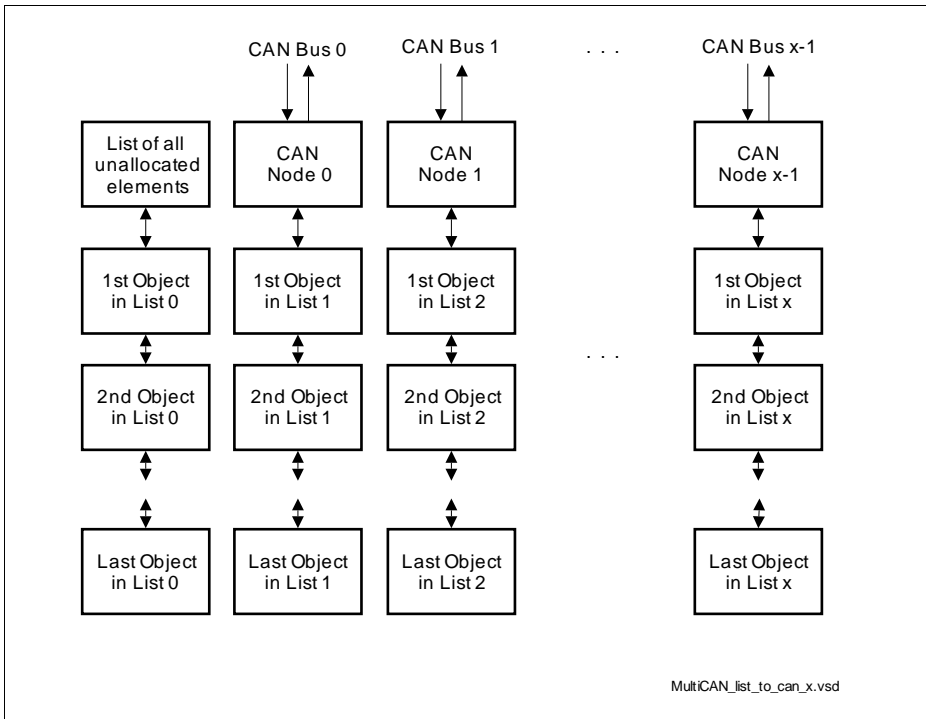


Figure 18-12 Message Objects Linked to CAN Nodes

18.3.5.4 List Command Panel

The list structure cannot be modified directly by write accesses to the LIST registers and the PPREV, PNEXT and LIST bit fields in the Message Object Status Registers, as they are read only. The list structure is managed by and limited to the list controller inside the MultiCAN+ module. The list controller is controlled via a command panel allowing the user to issue list allocation commands to the list controller. The list controller has two main purposes:

1. Ensure that all operations that modify the list structure result in a consistent list structure.
2. Present maximum ease of use and flexibility to the user.

The list controller and the associated command panel allows the programmer to concentrate on the final properties of the list, which are characterized by the allocation of message objects to a CAN node, and the ordering relation between objects that are allocated to the same list. The process of list (re-)building is done in the list controller.

Controller Area Network Controller (MultiCAN+)

Table 18-4 gives an overview on the available panel commands while **Table 18-8** on **Page 18-64** describes the panel commands in more detail.

Table 18-4 Panel Commands Overview

Command Name	Description
No Operation	No new command is started.
Initialize Lists	Run the initialization sequence to reset the CTRL and LIST field of all message objects.
Static Allocate	Allocate message object to a list.
Dynamic Allocate	Allocate the first message object of the list of unallocated objects to the selected list.
Static Insert Before	Remove a message object (source object) from the list that it currently belongs to, and insert it before a given destination object into the list structure of the destination object.
Dynamic Insert Before	Insert a new message object before a given destination object.
Static Insert Behind	Remove a message object (source object) from the list that it currently belongs to, and insert it behind a given destination object into the list structure of the destination object.
Dynamic Insert Behind	Insert a new message object behind a given destination object.

A panel command is started by writing the respective command code into the Panel Control Register bit field PANCTR.PANCMD (see **Page 18-63**). The corresponding command arguments must be written into bit fields PANCTR.PANAR1 and PANCTR.PANAR2 before writing the command code, or latest along with the command code in a single 32-bit write access to the Panel Control Register.

With the write operation of a valid command code, the PANCTR.BUSY flag is set and further write accesses to the Panel Control Register are ignored. The BUSY flag remains active and the control panel remains locked until the execution of the requested command has been completed. After a reset and resetting the CLC.DISR (see **Page 18-117**) register, the list controller builds up list 0. Afterwards the BUSY bit is set, dependent on core speed this might be visible. During list controller initialization, BUSY is set and other accesses to the CAN RAM are forbidden. The CAN RAM can be accessed again when BUSY becomes inactive.

Note: The CAN RAM is automatically initialized after enabling the clocks of the module, by the list controller in order to ensure correct list pointers in each message object. The operation is indicated by automatically setting the BUSY bit. The end of this CAN RAM initialization is indicated by bit PANCTR.BUSY becoming inactive. It is

Controller Area Network Controller (MultiCAN+)

advised to initialize some registers within the CAN controller before polling the PANCTR.BUSY the first time.

In case of a dynamic allocation command that takes an element from the list of unallocated objects, the PANCTR.RBUSY bit is also set along with the BUSY bit (RBUSY = BUSY = 1). This indicates that bit fields PANAR1 and PANAR2 are going to be updated by the list controller in the following way:

1. The message number of the message object taken from the list of unallocated elements is written to PANAR1.
2. If ERR (bit 7 of PANAR2) is set to 1, the list of unallocated elements was empty and the command is aborted. If ERR is 0, the list was not empty and the command will be performed successfully.

The results of a dynamic allocation command are written before the list controller starts the actual allocation process. As soon as the results are available, RBUSY becomes inactive (RBUSY = 0) again, while BUSY still remains active until completion of the command. This allows the user to set up the new message object while it is still in the process of list allocation. The access to message objects is not limited during ongoing list operations. However, any access to a register resource located inside the RAM delays the ongoing allocation process by one access cycle.

As soon as the command is finished, the BUSY flag becomes inactive (BUSY = 0) and write accesses to the Panel Control Register are enabled again. Also, the “No Operation” command code is automatically written to the PANCTR.PANCMD field. A new command may be started any time when BUSY = 0.

All fields of the Panel Control Register PANCTR except BUSY and RBUSY may be written by the user. This makes it possible to save and restore the Panel Control Register if the Command Panel is used within independent (mutually interruptible) interrupt service routines. If this is the case, any task that uses the Command Panel and that may interrupt another task that also uses the Command Panel should poll the BUSY flag until it becomes inactive and save the whole PANCTR register to a memory location before issuing a command. At the end of the interrupt service routine, the task should restore PANCTR from the memory location.

Before a message object that is allocated to the list of an active CAN node shall be moved to another list or to another position within the same list, bit MOSTATn.MSGVAL (“Message Valid”) of message object n must be cleared.

18.3.6 CAN Node Analyzer Mode

The chapter describes the CAN node analyzer capabilities of the MultiCAN+ module.

18.3.6.1 Analyzer Mode

The CAN Analyzer Mode makes it possible to monitor the CAN traffic for each CAN node individually without affecting the logical state of the CAN bus. The CAN Analyzer Mode for CAN node x is selected by setting Node x Control Register bit NCRx.CALM.

In CAN Analyzer Mode, the transmit pin of a CAN node is held at a recessive level permanently. The CAN node may receive frames (Data, Remote, and Error Frames) but is not allowed to transmit. Received Data/Remote Frames are not acknowledged (i.e. acknowledge slot is sent recessive) but will be received and stored in matching message objects as long as there is any other node that acknowledges the frame. The complete message object functionality is available, but no transmit request will be executed.

18.3.6.2 Loop-Back Mode

The MultiCAN+ module provides a Loop-Back Mode to enable an in-system test of the MultiCAN+ module as well as the development of CAN driver software without access to an external CAN bus.

The loop-back feature consists of an internal CAN bus (inside the MultiCAN+ module) and a bus select switch for each CAN node (see [Figure 18-13](#)). With the switch, each CAN node can be connected either to the internal CAN bus (Loop-Back Mode activated) or the external CAN bus, respectively to transmit and receive pins (normal operation). The CAN bus that is not currently selected is driven recessive; this means the transmit pin is held at 1, and the receive pin is ignored by the CAN nodes that are in Loop-Back Mode.

The Loop-Back Mode is selected for CAN node x by setting the Node x Port Control Register bit NPCRx.LBM. All CAN nodes that are in Loop-Back Mode may communicate together via the internal CAN bus without affecting the normal operation of the other CAN nodes that are not in Loop-Back Mode.

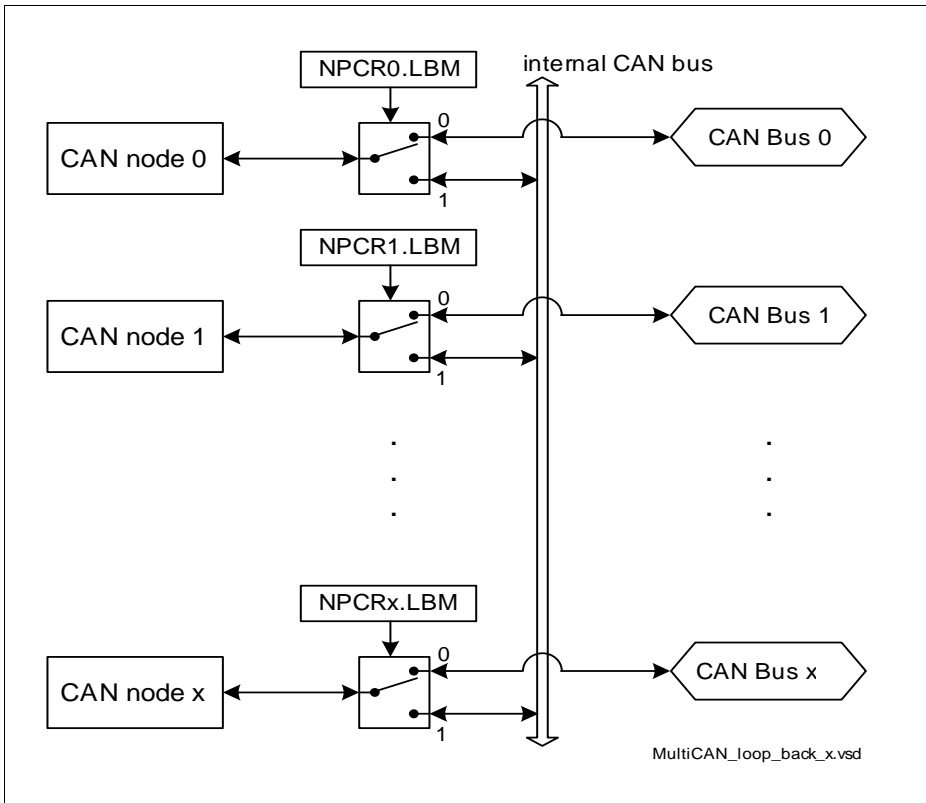


Figure 18-13 Loop-Back Mode

18.3.6.3 Bit Timing Analysis

Detailed analysis of the bit timing can be performed for each CAN node using the analysis modes of the CAN frame counter. The bit timing analysis functionality of the frame counter may be used for automatic detection of the CAN baud rate, as well as to analyze the timing of the CAN network.

Bit timing analysis for CAN node x is selected when bit field $NFCRx.CFMODE = 10_B$. Bit timing analysis does not affect the operation of the CAN node.

The bit timing measurement results are written into the $NFCRx.CFC$ bit field. Whenever $NFCRx.CFC$ is updated in bit timing analysis mode, bit $NFCRx.CFCOV$ is also set to indicate the CFC update event. The value of $NFCRx.CFC$ is valid one module cycle later when $NFCRx.CFCOV$ is set. If $NFCRx.CFCIE$ is set, an interrupt request can be generated (see [Figure 18-10](#)).

Automatic Baud Rate Detection

For automatic baud rate detection, the time between the observation of subsequent dominant edges on the CAN bus must be measured. This measurement is automatically performed if bit field $\text{NFCRx.CFSEL} = 000_{\text{B}}$. With each dominant edge monitored on the CAN receive input line, the time (measured in f_{CLC} clock cycles) between this edge and the most recent dominant edge is stored in the NFCRx.CFC bit field.

Synchronization Analysis

The bit time synchronization is monitored if $\text{NFCRx.CFSEL} = 010_{\text{B}}$. The time between the first dominant edge and the sample point is measured and stored in the NFCRx.CFC bit field. The bit timing synchronization offset may be derived from this time as the first edge after the sample point triggers synchronization and there is only one synchronization between consecutive sample points.

Synchronization analysis can be used, for example, for fine tuning of the baud rate during reception of the first CAN frame with the measured baud rate.

Driver Delay Measurement

The delay between a transmitted edge and the corresponding received edge is measured when $\text{NFCRx.CFSEL} = 011_{\text{B}}$ (dominant to dominant) and $\text{NFCRx.CFSEL} = 100_{\text{B}}$ (recessive to recessive). These delays indicate the time needed to represent a new bit value on the physical implementation of the CAN bus.

18.3.7 Message Acceptance Filtering

The chapter describes the Message Acceptance Filtering capabilities of the MultiCAN+ module.

18.3.7.1 Receive Acceptance Filtering

When a CAN frame is received by a CAN node, a unique message object is determined in which the received frame is stored after successful frame reception. A message object is qualified for reception of a frame if the following six conditions are met.

- The message object is allocated to the message object list of the CAN node by which the frame is received.
- Bit MOSTATn.MSGVAL in the Message Object Status Register (see [Page 18-95](#)) is set.
- Bit MOSTATn.RXEN is set.
- Bit MOSTATn.DIR is equal to bit RTR of the received frame.

If bit $\text{MOSTATn.DIR} = 1$ (transmit object), the message object accepts only Remote Frames. If bit $\text{MOSTATn.DIR} = 0$ (receive object), the message object accepts only Data Frames.

Controller Area Network Controller (MultiCAN+)

- If bit MOAMRn.MIDE = 1, the IDE bit of the received frame becomes evaluated in the following way: If MOARn.IDE = 1, the IDE bit of the received frame must be set (indicates extended identifier). If MOARn.IDE = 0, the IDE bit of the received frame must be cleared (indicates standard identifier).
If bit MOAMRn.MIDE = 0, the IDE bit of the received frame is “don't care”. In this case, message objects with standard and extended frames are accepted.
- The identifier of the received frame matches the identifier stored in the Arbitration Register of the message object as qualified by the acceptance mask in the MOAMRn register. This means that each bit of the received message object identifier is equal to the bit field MOARn.ID, except those bits for which the corresponding acceptance mask bits in bit field MOAMRn.AM are cleared. These identifier bits are “don't care” for reception. **Figure 18-14** illustrates this receive message identifier check.

Among all messages that fulfill all six qualifying criteria the message object with the highest receive priority wins receive acceptance filtering and becomes selected to store the received frame. All other message objects lose receive acceptance filtering.

The following priority scheme is defined for the message objects:

A message object a (MOa) has higher receive priority than a message object b (MOb) if the following two conditions are fulfilled (see **Page 18-109**):

1. MOa has a higher priority class than MOb. This means, the 2-bit priority bit field MOARa.PRI must be equal or less than bit field MOARb.PRI.
2. If both message objects have the same priority class (MOARa.PRI = MOARb.PRI), MOb is a list successor of MOa. This means that MOb can be reached by means of successively stepping forward in the list, starting from a.

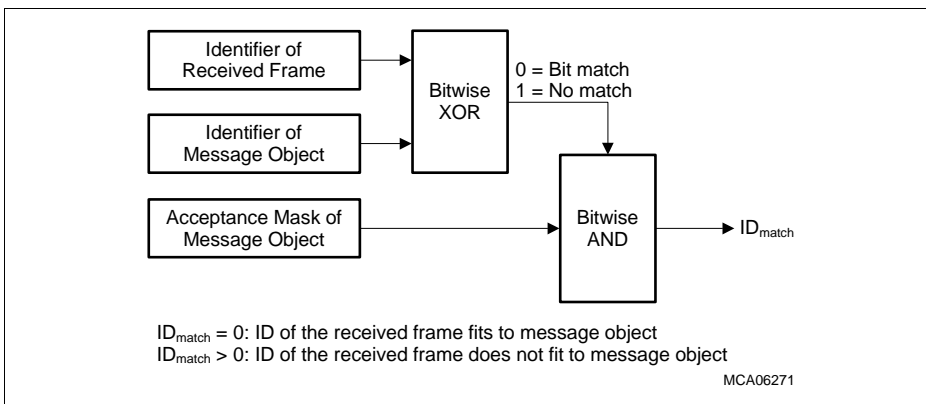


Figure 18-14 Received Message Identifier Acceptance Check

18.3.7.2 Transmit Acceptance Filtering

A message is requested for transmission by setting a transmit request in the message object that holds the message. If more than one message object have a valid transmit request for the same CAN node, one of these message objects is chosen for transmission, because only a single message object can be transmitted at one time on a CAN bus.

A message object is qualified for transmission on a CAN node if the following four conditions are met (see also [Figure 18-15](#)).

1. The message object is allocated to the message object list of the CAN node.
2. Bit MOSTATn.MSGVAL is set.
3. Bit MOSTATn.TXRQ is set.
4. Bit MOSTATn.TXEN0 and MOSTATn.TXEN1 are set.

A priority scheme determines which one of all qualifying message objects is transmitted first. It is assumed that message object a (MOa) and message object b (MOb) are two message objects qualified for transmission. MOb is a list successor of MOa. For both message objects, CAN messages CANa and CANb are defined (identifier, IDE, and RTR are taken from the message-specific bit fields and bits MOARn.ID, MOARn.IDE and MOSTATn.DIR).

If both message objects belong to the same priority class (identical PRI bit field in register MOARn), MOa has a higher transmit priority than MOb if one of the following conditions is fulfilled.

- $PRI = 10_B$ and CAN message MOa has higher or equal priority than CAN message MOb with respect to CAN arbitration rules (see [Table 18-14](#) on [Page 18-110](#)).
- $PRI = 01_B$ or $PRI = 11_B$ (priority by list order).
- $PRI = 00_B$ is reserved and makes the message object to have no function.

The message object that is qualified for transmission and has highest transmit priority wins the transmit acceptance filtering, and will be transmitted first. All other message objects lose the current transmit acceptance filtering round. They get a new chance in subsequent acceptance filtering rounds.

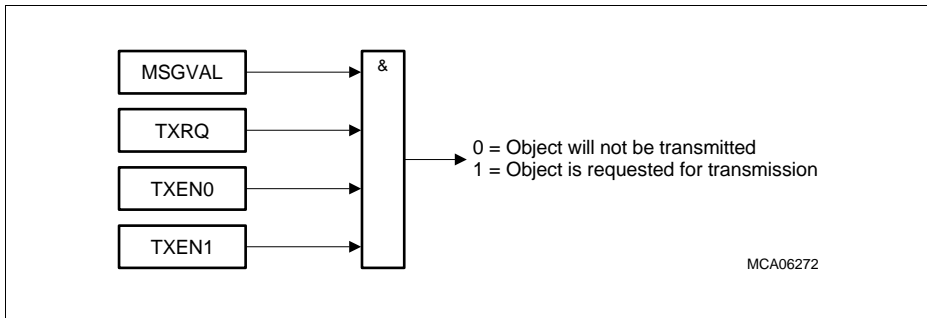


Figure 18-15 Effective Transmit Request of Message Object

18.3.8 Message Postprocessing

After a message object has successfully received or transmitted a frame, the CPU can be notified to perform a postprocessing on the message object. The postprocessing of the MultiCAN+ module consists of two elements:

1. Message interrupts to trigger postprocessing.
2. Message pending registers to collect pending message interrupts into a common structure for postprocessing.

18.3.8.1 Message Object Interrupts

When the storage of a received frame into a message object or the successful transmission of a frame is completed, a message interrupt can be issued. For each message object, a transmit and a receive interrupt can be generated and routed to one of the sixteen CAN interrupt output lines (see [Figure 18-16](#)). A receive interrupt occurs also after a frame storage event that has been induced by a FIFO or a gateway action. The status bits TXPND and RXPND in the Message Object n Status Register are always set after a successful transmission/reception, whether or not the respective message interrupt is enabled.

A third FIFO full interrupt condition of a message object is provided. If bit field MOFCRn.OVIE (Overflow Interrupt Enable) is set, the FIFO full interrupt will be activated depending on the actual message object type.

In case of a Receive FIFO Base Object (MOFCRn.MMC = 0001_B), the FIFO full interrupt is routed to the interrupt output line INT_{Om} as defined by the transmit interrupt node pointer MOIPRn.TXINP.

In case of a Transmit FIFO Base Object (MOFCRn.MMC = 0010_B), the FIFO full interrupt becomes routed to the interrupt output line INT_{Om} as defined by the receive interrupt node pointer MOIPRn.RXINP.

Controller Area Network Controller (MultiCAN+)

See also **“Interrupt Control” on Page 18-122** for further processing of the message object interrupts.

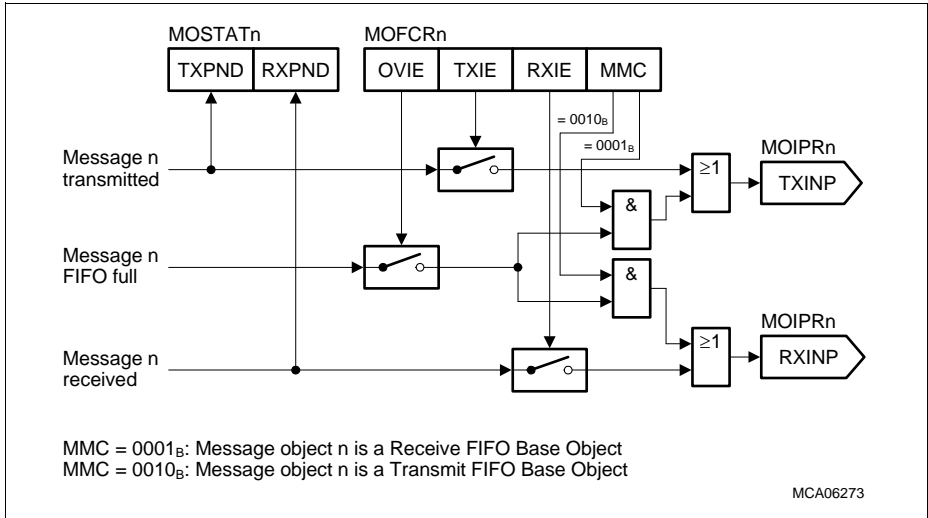


Figure 18-16 Message Interrupt Request Routing

Controller Area Network Controller (MultiCAN+)

18.3.8.2 Pending Messages

When a message interrupt request is generated, a message pending bit is set in one of the Message Pending Registers. There are 8 Message Pending Registers, MSPNDk (k = 0-7) with 32 pending bits available each. The general **Figure 18-17** shows the allocation of the message pending bits in case that the maximum possible number of eight Message Pending Registers are implemented and available on the chip.

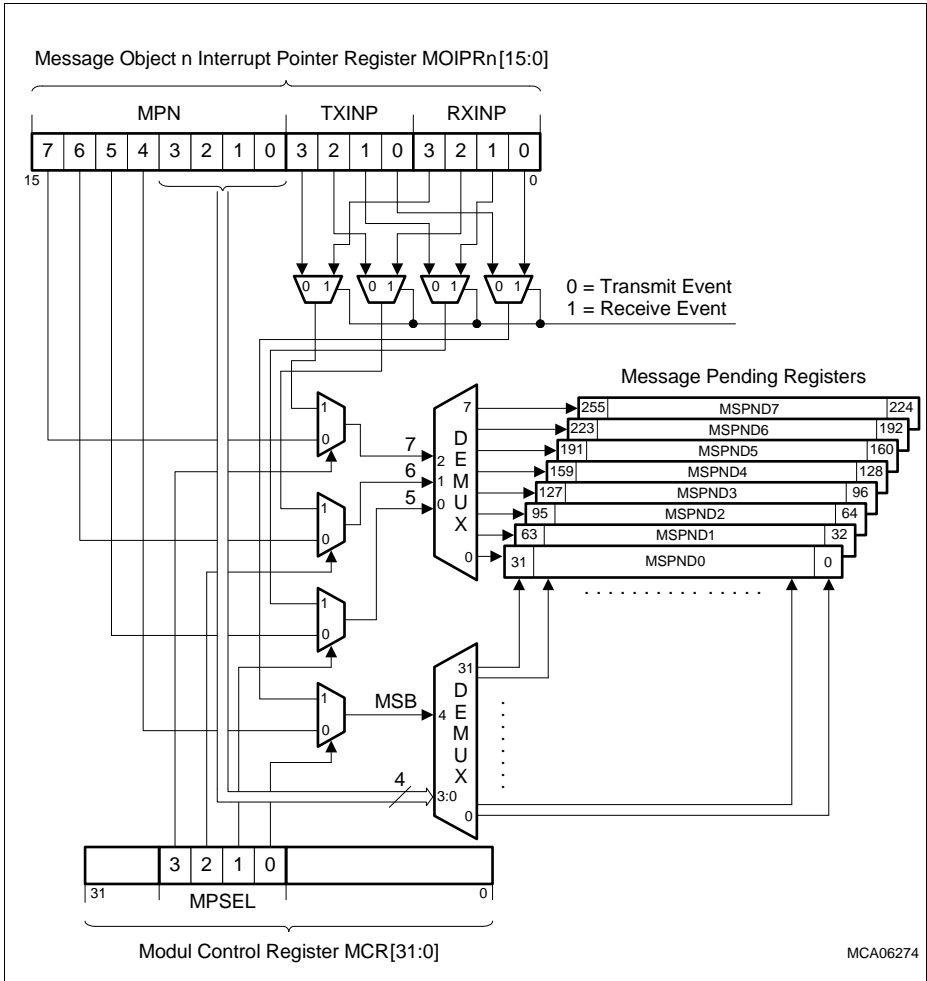


Figure 18-17 Message Pending Bit Allocation

Controller Area Network Controller (MultiCAN+)

The location of a pending bit is defined by two demultiplexers selecting the number k of the MSPNDk registers (3-bit demux), and the bit location within the corresponding MSPNDk register (5-bit demux).

Allocation Case 1

In this allocation case, bit field MCR.MPSEL = 0000_B (see [Page 18-67](#)). The location selection consists of 2 parts:

- The upper three bits of MOIPRn.MPN (MPN[7:5]) select the number k of a Message Pending Register MSPNDk in which the pending bit will be set.
- The lower five bits of MOIPRn.MPN (MPN[4:0]) select the bit position (0-31) in MSPNDk for the pending bit to be set.

Allocation Case 2

In this allocation case, bit field MCR.MPSEL is taken into account for pending bit allocation. Bit field MCR.MPSEL makes it possible to include the interrupt request node pointer for reception (MOIPRn.RXINP) or transmission (MOIPRn.TXINP) for pending bit allocation in such a way that different target locations for the pending bits are used in receive and transmit case. If MPSEL = 1111_B, the location selection operates in the following way:

- At a transmit event, the upper 3 bits of TXINP determine the number k of a Message Pending Register MSPNDk in which the pending bit will be set. At a receive event, the upper 3 bits of RXINP determine the number k .
- The bit position (0-31) in MSPNDk for the pending bit to be set is selected by the lowest bit of TXINP or RXINP (selects between low and high half-word of MSPNDk) and the four least significant bits of MPN.

General Hints

The Message Pending Registers MSPNDk can be written by software. Bits that are written with 1 are left unchanged, and bits which are written with 0 are cleared. This makes it possible to clear individual MSPNDk bits with a single register write access. Therefore, access conflicts are avoided when the MultiCAN+ module (hardware) sets another pending bit at the same time when software writes to the register.

Each Message Pending Register MSPNDk is associated with a Message Index Register MSIDk (see [Page 18-73](#)) which indicates the lowest bit position of all set (1) bits in Message Pending Register k . The MSIDk register is a read-only register that is updated immediately when a value in the corresponding Message Pending Register k is changed by software or hardware.

18.3.9 Message Object Data Handling

This chapter describes the handling capabilities for the Message Object Data of the MultiCAN+ module.

18.3.9.1 Frame Reception

After the reception of a message, it is stored in a message object according to the scheme shown in [Figure 18-18](#). The MultiCAN+ module not only copies the received data into the message object, and it provides advanced features to enable consistent data exchange between MultiCAN+ and CPU.

MSGVAL

Bit MSGVAL (Message Valid) in the Message Object n Status Register MOSTATn is the main switch of the message object. During the frame reception, information is stored in the message object only when MSGVAL = 1. If bit MSGVAL is reset by the CPU, the MultiCAN+ module stops all ongoing write accesses to the message object. Now the message object can be re-configured by the CPU with subsequent write accesses to it without being disturbed by the MultiCAN+.

RTSEL

When the CPU re-configures a message object during CAN operation (for example, clears MSGVAL, modifies the message object and sets MSGVAL again), the following scenario can occur:

1. The message object wins receive acceptance filtering.
2. The CPU clears MSGVAL to re-configure the message object.
3. The CPU sets MSGVAL again after re-configuration.
4. The end of the received frame is reached. As MSGVAL is set, the received data is stored in the message object, a message interrupt request is generated, gateway and FIFO actions are processed, etc.

After the re-configuration of the message object (after step 3 above) the storage of further received data may be undesirable. This can be achieved through bit MOSTATn.RTSEL (Receive/Transmit Selected) that makes it possible to disconnect a message object from an ongoing frame reception.

When a message object wins the receive acceptance filtering, its RTSEL bit is set by the MultiCAN+ module to indicate an upcoming frame delivery. The MultiCAN+ module checks RTSEL whether it is set on successful frame reception to verify that the object is still ready for receiving the frame. The received frame is then stored in the message object (along with all subsequent actions such as message interrupts, FIFO & gateway actions, flag updates) only if RTSEL = 1.

When a message object is invalidated during CAN operation (resetting bit MSGVAL), RTSEL should be cleared before setting MSGVAL again (latest with the same write

Controller Area Network Controller (MultiCAN+)

access that sets MSGVAL) to prevent the storage of a frame that belongs to the old context of the message object. Therefore, a message object re-configuration should consist of the following steps:

1. Clear MSGVAL bit
2. Re-configure the message object while MSGVAL = 0
3. Clear RTSEL bit and set MSGVAL again

RXEN

Bit MOSTATn.RXEN enables a message object for frame reception. A message object can receive CAN messages from the CAN bus only if RXEN = 1. The MultiCAN+ module evaluates RXEN only during receive acceptance filtering. After receive acceptance filtering, RXEN is ignored and has no further influence on the actual storage of a received message in a message object.

Bit RXEN enables the “soft phase out” of a message object: after clearing RXEN, a currently received CAN message for which the message object has won acceptance filtering is still stored in the message object but for subsequent messages the message object no longer wins receive acceptance filtering.

RXUPD, NEWDAT and MSGLST

An ongoing frame storage process is indicated by the RXUPD (Receive Updating) flag in the MOSTATn register. RXUPD is set with the start and cleared with the end of a message object update, which consists of frame storage as well as flag updates.

After storing the received frame (identifier, IDE bit, DLC; including the Data Field for Data Frames), the NEWDAT (New Data) bit of the message object is set. If NEWDAT was already set before it becomes set again, bit MSGLST (Message Lost) is set to indicate a data loss condition.

The RXUPD and NEWDAT flags can help to read consistent frame data from the message object during an ongoing CAN operation. The following steps are recommended to be executed:

1. Clear NEWDAT bit.
2. Read message content (identifier, data etc.) from the message object.
3. Check that both, NEWDAT and RXUPD, are cleared. If this is not the case, go back to step 1.
4. When step 3 was successful, the message object contents is consistent and has not been updated by the MultiCAN+ module while reading.

Bits RXUPD, NEWDAT and MSGLST have the same behavior for the reception of Data as well as Remote Frames.

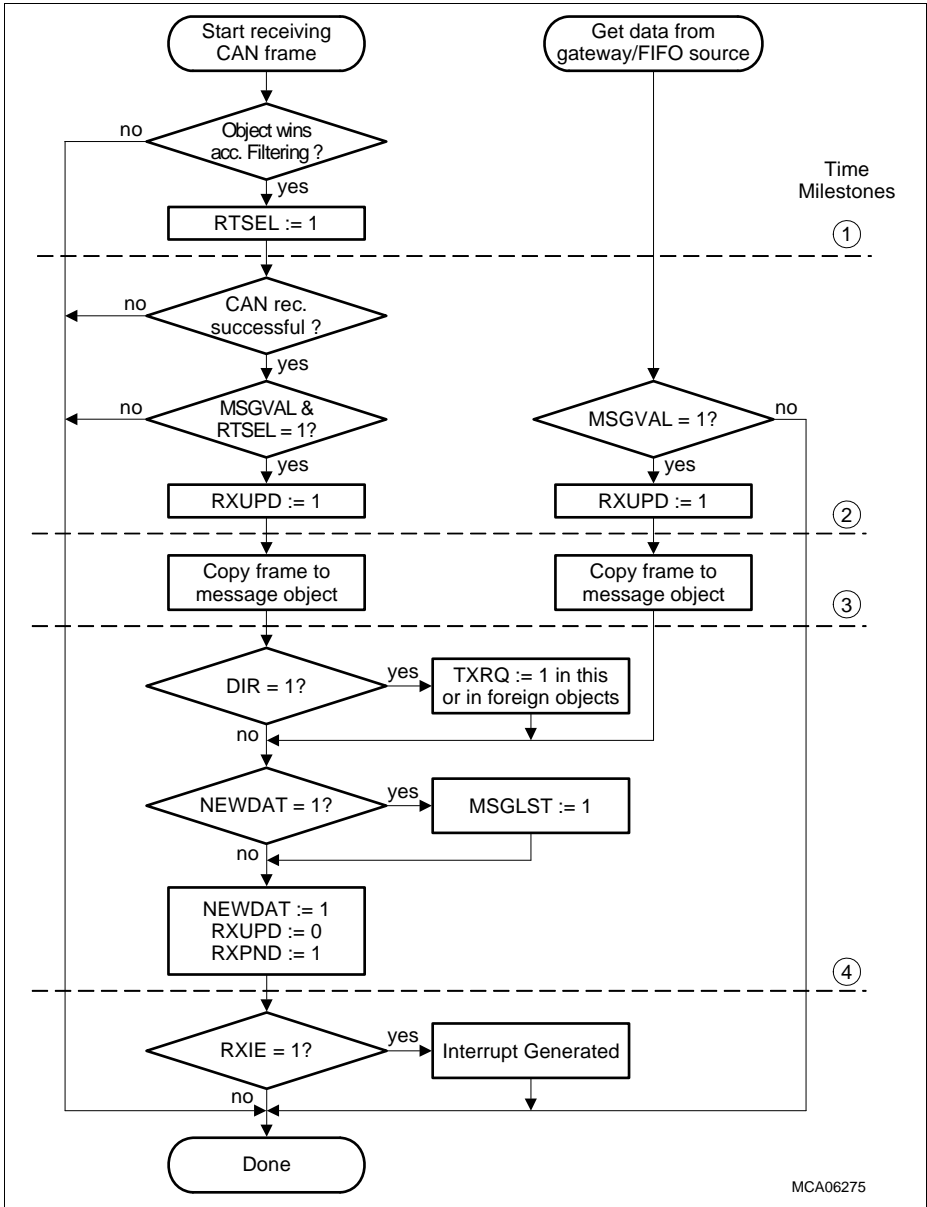


Figure 18-18 Reception of a Message Object

18.3.9.2 Frame Transmission

The process of a message object transmission is shown in [Figure 18-19](#). Along with the copy of the message object content to be transmitted (identifier, IDE bit, RTR = DIR bit, DLC, including the Data Field for Data Frames) into the internal transmit buffer of the assigned CAN node, several status flags are also served and monitored to control consistent data handling.

The transmission process of a message object starting after the transmit acceptance filtering is identical for Remote and Data Frames.

MSGVAL, TXRQ, TXEN0, TXEN1

A message can only be transmitted if all four bits in registers MOSTATn, MSGVAL (Message Valid), TXRQ (Transmit Request), TXEN0 (Transmit Enable 0), TXEN1 (Transmit Enable 1) are set as shown in [Figure 18-15](#). Although these bits are equivalent with respect to the transmission process, they have different semantics:

Table 18-5 Message Transmission Bit Definitions

Bit	Description
MSGVAL	<p>Message Valid This is the main switch bit of the message object.</p>
TXRQ	<p>Transmit Request This is the standard transmit request bit. This bit must be set whenever a message object should be transmitted. TXRQ is cleared by hardware at the end of a successful transmission, except when there is new data (indicated by NEWDAT = 1) to be transmitted. When bit MOFCRn.STT (“Single Transmit Trial”) is set, TXRQ becomes already cleared when the contents of the message object are copied into the transmit frame buffer of the CAN node. A received remote request (after a Remote Frame reception) sets bit TXRQ to request the transmission of the requested data frame.</p>
TXEN0	<p>Transmit Enable 0 This bit can be temporarily cleared by software to suppress the transmission of this message object when it writes new content to the Data Field. This avoids transmission of inconsistent frames that consist of a mixture of old and new data. Remote requests are still accepted when TXEN0 = 0, but transmission of the Data Frame is suspended until transmission is re-enabled by software (setting TXEN0).</p>

Controller Area Network Controller (MultiCAN+)

Table 18-5 Message Transmission Bit Definitions (cont'd)

Bit	Description
TXEN1	<p>Transmit Enable 1</p> <p>This bit is used in transmit FIFOs to select the message object that is transmit active within the FIFO structure.</p> <p>For message objects that are not transmit FIFO elements, TXEN1 can either be set permanently to 1 or can be used as a second independent transmission enable bit.</p>

RTSEL

When a message object has been identified to be transmitted next after transmission acceptance filtering, bit MOSTATn.RTSEL (Receive/Transmit Selected) is set.

When the message object is copied into the internal transmit buffer, bit RTSEL is checked, and the message is transmitted only if RTSEL = 1. After the successful transmission of the message, bit RTSEL is checked again and the message postprocessing is only executed if RTSEL = 1.

For a complete re-configuration of a valid message object, the following steps should be executed:

1. Clear MSGVAL bit
2. Re-configure the message object while MSGVAL = 0
3. Clear RTSEL and set MSGVAL

Clearing of RTSEL ensures that the message object is disconnected from an ongoing/scheduled transmission and no message object processing (copying message to transmit buffer including clearing NEWDAT, clearing TXRQ, time stamp update, message interrupt, etc.) within the old context of the object can occur after the message object becomes valid again, but within a new context.

NEWDAT

When the contents of a message object have been transferred to the internal transmit buffer of the CAN node, bit MOSTATn.NEWDAT (New Data) is cleared by hardware to indicate that the transmit message object data is no longer new.

When the transmission of the frame is successful and NEWDAT is still cleared (if no new data has been copied into the message object meanwhile), TXRQ (Transmit Request) is cleared automatically by hardware.

If, however, the NEWDAT bit has been set again by the software (because a new frame should be transmitted), TXRQ is not cleared to enable the transmission of the new data.

Controller Area Network Controller (MultiCAN+)

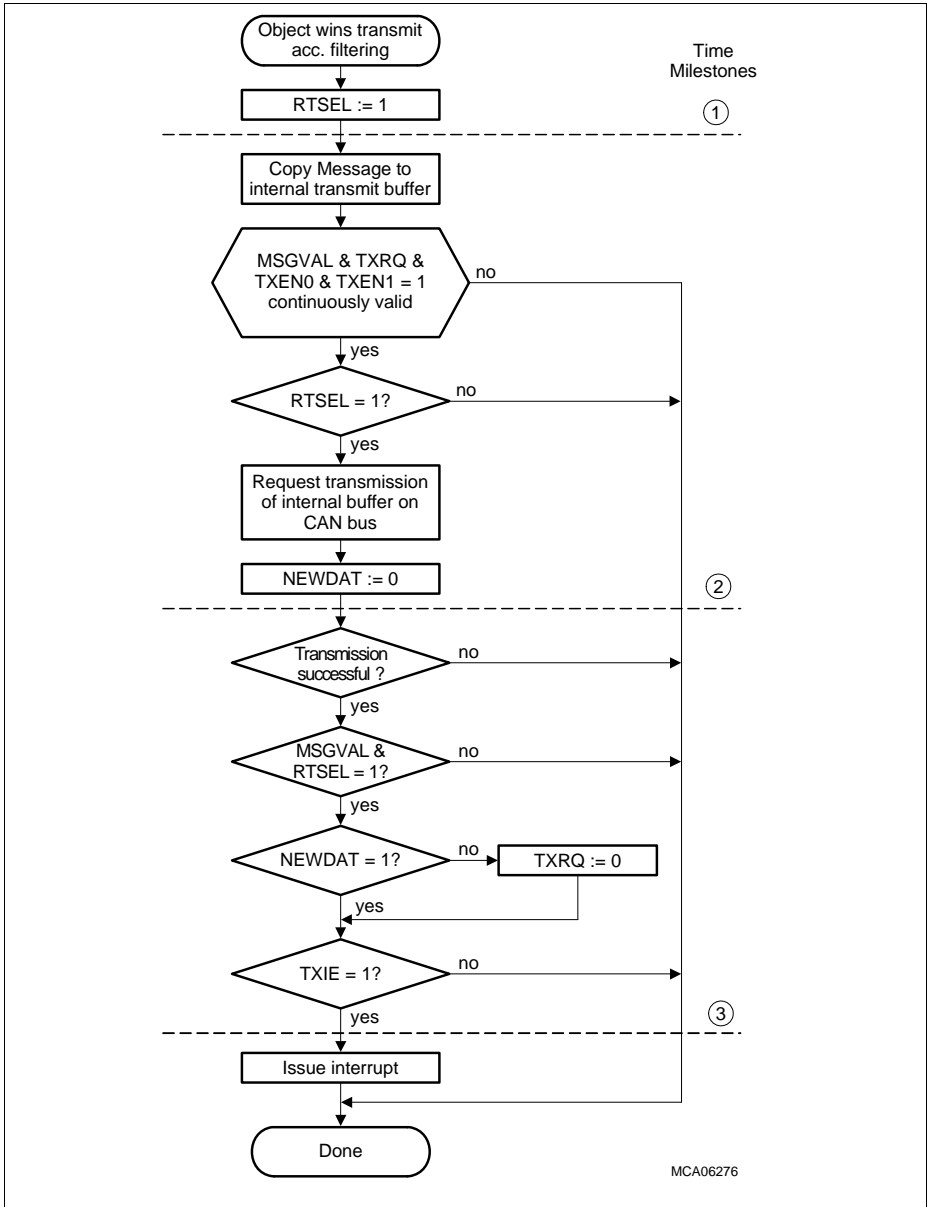


Figure 18-19 Transmission of a Message Object

18.3.10 Message Object Functionality

This chapter describes the functionality of the Message Objects in the MultiCAN+ module.

18.3.10.1 Standard Message Object

A message object is selected as standard message object when bit field MOFCRn.MMC = 0000_B (see [Page 18-87](#)). The standard message object can transmit and receive CAN frames according to the basic rules described in the previous sections. Additional services such as Single Data Transfer Mode or Single Transmit Trial (see following sections) are available and can be individually selected.

18.3.10.2 Single Data Transfer Mode

Single Data Transfer Mode is a useful feature in order to broadcast data over the CAN bus without unintended duplication of information. Single Data Transfer Mode is selected via bit MOFCRn.SDT.

Message Reception

When a received message stored in a message object is overwritten by a new received message, the contents of the first message are lost and replaced with the contents of the new received message (indicated by MSGLST = 1).

If SDT is set (Single Data Transfer Mode activated), bit MSGVAL of the message object is automatically cleared by hardware after the storage of a received Data or Remote Frame. This prevents the reception of further messages.

Message Transmission

When a message object receives a series of multiple remote requests, it transmits several Data Frames in response to the remote requests. If the data within the message object has not been updated in the time between the transmissions, the same data can be sent more than once on the CAN bus.

In Single Data Transfer Mode (SDT = 1), this is avoided because MSGVAL is automatically cleared after the successful transmission of a Data or Remote Frame.

18.3.10.3 Single Transmit Trial

If the bit STT in the message object function register is set (STT = 1), the transmission request is cleared (TXRQ = 0) when the frame contents of the message object have been copied to the internal transmit buffer of the CAN node. Thus, the transmission of the message object is not tried again when it fails due to CAN bus errors.

18.3.10.4 Message Object FIFO Structure

In case of high CPU load it may be difficult to process a series of CAN frames in time. This may happen if multiple messages are received or must be transmitted in short time.

Therefore, a FIFO buffer structure is available to avoid loss of incoming messages and to minimize the setup time for outgoing messages. The FIFO structure can also be used to automate the reception or transmission of a series of CAN messages and to generate a single message interrupt when the whole CAN frame series is done.

There can be several FIFOs in parallel. The number of FIFOs and their size are limited only by the number of available message objects. A FIFO can be installed, resized and de-installed at any time, even during CAN operation.

The basic structure of a FIFO is shown in [Figure 18-20](#). A FIFO consists of one base object and n slave objects. The slave objects are chained together in a list structure (similar as in message object lists). The base object may be allocated to any list. Although [Figure 18-20](#) shows the base object as a separate part beside the slave objects, it is also possible to integrate the base object at any place into the chain of slave objects. This means that the base object is slave object, too (not possible for gateways). The absolute object numbers of the message objects have no impact on the operation of the FIFO.

The base object does not need to be allocated to the same list as the slave objects. Only the slave object must be allocated to a common list (as they are chained together). Several pointers (BOT, CUR and TOP) that are located in the Message Object n FIFO/Gateway Pointer Register MOFGPRn link the base object to the slave objects, regardless whether the base object is allocated to the same or to another **list** than the slave objects.

The smallest FIFO would be a single message object which is both, FIFO base and FIFO slave (not very useful). The biggest possible FIFO structure would include all message objects of the MultiCAN+ module. Any FIFO sizes between these limits are possible.

In the FIFO base object, the FIFO boundaries are defined. Bit field MOFGPRn.BOT of the base object points to (includes the number of) the bottom slave object in the FIFO structure. The MOFGPRn.TOP bit field points to (includes the number of) the top slave object in the FIFO structure. The MOFGPRn.CUR bit field points to (includes the number of) the slave object that is actually selected by the MultiCAN+ module for message transfer. When a message transfer takes place with this object, CUR is set to the next message object in the list structure of the slave objects (CUR = PNEXT of current object). If CUR was equal to TOP (top of the FIFO reached), the next update of CUR will result in CUR = BOT (wrap-around from the top to the bottom of the FIFO). This scheme represents a circular FIFO structure where the bit fields BOT and TOP establish the link from the last to the first element.

Bit field MOFGPRn.SEL of the base object can be used for monitoring purposes. It makes it possible to define a slave object within the list at which a message interrupt is

Controller Area Network Controller (MultiCAN+)

generated whenever the CUR pointer reaches the value of the SEL pointer. Thus SEL makes it possible to detect the end of a predefined message transfer series or to issue a warning interrupt when the FIFO becomes full.

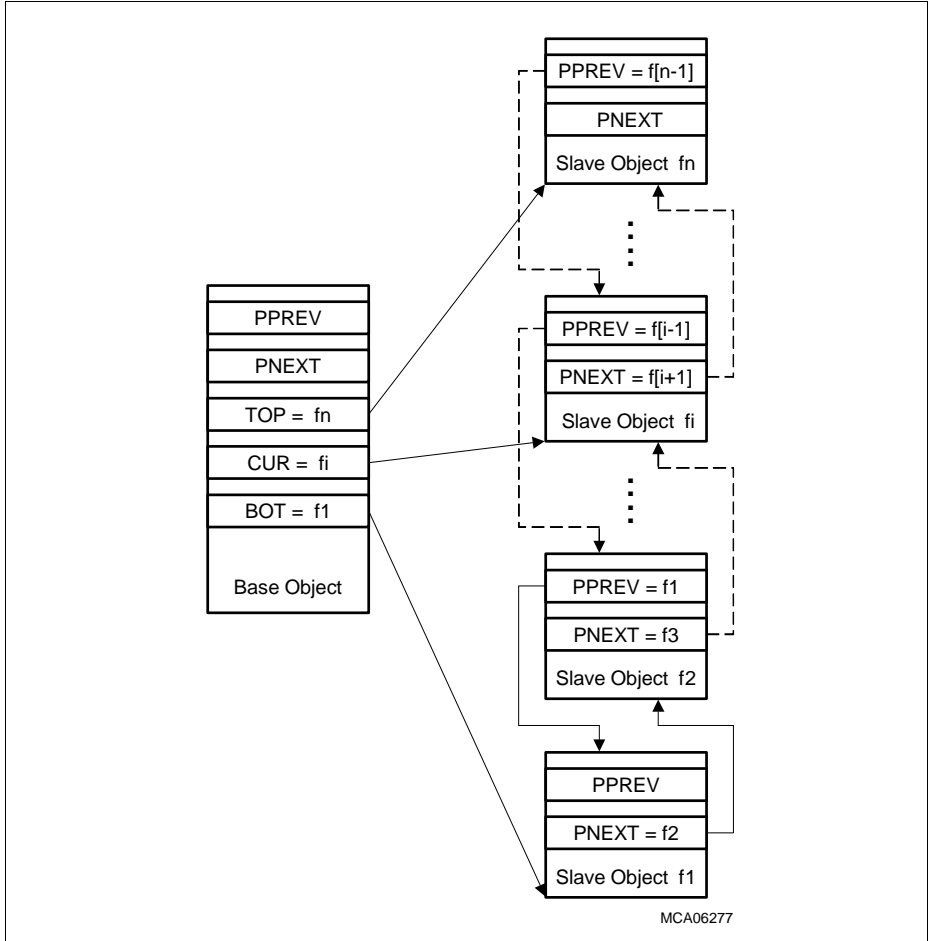


Figure 18-20 FIFO Structure with FIFO Base Object and n FIFO Slave Objects

18.3.10.5 Receive FIFO

The Receive FIFO structure is used to buffer incoming (received) Remote or Data Frames.

A Receive FIFO is selected by setting $\text{MOFCRn.MMC} = 0001_{\text{B}}$ in the FIFO base object. This MMC code automatically designates a message object as FIFO base object. The message modes of the FIFO slave objects are not relevant for the operation of the Receive FIFO.

When the FIFO base object receives a frame from the CAN node it belongs to, the frame is not stored in the base object itself but in the message object that is selected by the base object's MOFGPRn.CUR pointer. This message object receives the CAN message as if it is the direct receiver of the message. However, $\text{MOFCRn.MMC} = 0000_{\text{B}}$ is implicitly assumed for the FIFO slave object, and a standard message delivery is performed. The actual message mode (MMC setting) of the FIFO slave object is ignored. For the slave object, no acceptance filtering takes place that checks the received frame for a match with the identifier, IDE bit, and DIR bit.

With the reception of a CAN frame, the current pointer CUR of the base object is set to the number of the next message object in the FIFO structure. This message object will then be used to store the next incoming message.

If bit field MOFCRn.OVIE ("Overflow Interrupt Enable") of the FIFO base object is set and the current pointer MOFGPRn.CUR becomes equal to MOFGPRn.SEL , a FIFO overflow interrupt request is generated. This interrupt request is generated on interrupt node TXINP of the base object immediately after the storage of the received frame in the slave object. Transmit interrupts are still generated if TXIE is set.

A CAN message is stored in FIFO base and slave object only if $\text{MSGVAL} = 1$.

In order to avoid direct reception of a message by a slave message object, as if it was an independent message object and not a part of a FIFO, the bit RXEN of each slave object must be cleared. The setting of the bit RXEN is "don't care" only if the slave object is located in a list not assigned to a CAN node.

18.3.10.6 Transmit FIFO

The Transmit FIFO structure is used to buffer a series of Data or Remote Frames that must be transmitted. A transmit FIFO consists of one base message object and one or more slave message objects.

A Transmit FIFO is selected by setting $\text{MOFCRn.MMC} = 0010_{\text{B}}$ in the FIFO base object. Unlike the Receive FIFO, slave objects assigned to the Transmit FIFO must explicitly set their bit fields $\text{MOFCRn.MMC} = 0011_{\text{B}}$. The CUR pointer in all slave objects must point back to the Transmit FIFO Base Object (to be initialized by software).

The MOSTATn.TXEN1 bits (Transmit Enable 1) of all message objects except the one which is selected by the CUR pointer of the base object must be cleared by software.

Controller Area Network Controller (MultiCAN+)

TXEN1 of the message (slave) object selected by CUR must be set. CUR (of the base object) may be initialized to any FIFO slave object.

When tagging the message objects of the FIFO as valid to start the operation of the FIFO, then the base object must be tagged valid (MSGVAL = 1) first.

Before a Transmit FIFO becomes de-installed during operation, its slave objects must be tagged invalid (MSGVAL = 0).

The Transmit FIFO uses the TXEN1 bit in the Message Object Status Register of all FIFO elements to select the actual message for transmission. Transmit acceptance filtering evaluates TXEN1 for each message object and a message object can win transmit acceptance filtering only if its TXEN1 bit is set. When a FIFO object has transmitted a message, the hardware clears its TXEN1 bit in addition to standard transmit postprocessing (clear TXRQ, transmit interrupt etc.), and moves the CUR pointer in the next FIFO base object to be transmitted. TXEN1 is set automatically (by hardware) in the next message object. Thus, TXEN1 moves along the Transmit FIFO structure as a token that selects the active element.

If bit field MOFCRn.OVIE (“Overflow Interrupt Enable”) of the FIFO base object is set and the current pointer CUR becomes equal to MOFGPRn.SEL, a FIFO overflow interrupt request is generated. The interrupt request is generated on interrupt node RXINP of the base object after postprocessing of the received frame. Receive interrupts are still generated for the Transmit FIFO base object if bit RXIE is set.

18.3.10.7 Gateway Mode

The Gateway Mode makes it possible to establish an automatic information transfer between two independent CAN buses without CPU interaction.

The Gateway Mode operates on message object level. In Gateway mode, information is transferred between two message objects, resulting in an information transfer between the two CAN nodes to which the message objects are allocated. A gateway may be established with any pair of CAN nodes, and there can be as many gateways as there are message objects available to build the gateway structure.

Gateway Mode is selected by setting MOFCRs.MMC = 0100_B for the gateway source object s. The gateway destination object d is selected by the MOFGPRs.CUR=d pointer of the source object. The gateway destination object only needs to be valid (its MSGVAL = 1). All other settings are not relevant for the information transfer from the source object to the destination object.

Gateway source object behaves as a standard message object with the difference that some additional actions are performed by the MultiCAN+ module when a CAN frame has been received and stored in the source object (see [Figure 18-21](#)):

1. If bit MOFCRs.DLCC is set, the data length code MOFCRs.DLC is copied from the gateway source object to the gateway destination object.

Controller Area Network Controller (MultiCAN+)

2. If bit MOFCRs.IDC is set, the identifier MOARs.ID and the identifier extension MOARs.IDE are copied from the gateway source object to the gateway destination object.
3. If bit MOFCRs.DATC is set, the data bytes stored in the two data registers MODATALs and MODATAHs are copied from the gateway source object to the gateway destination object. All 8 data bytes are copied, even if MOFCRs.DLC indicates less than 8 data bytes.
4. If bit MOFCRs.GDFS is set, the transmit request flag MOSTATd.TXRQ is set in the gateway destination object.
5. The receive pending bit MOSTATd.RXPND and the new data bit MOSTATd.NEWDAT are set in the gateway destination object.
6. A message interrupt request is generated for the gateway destination object if its MOSTATd.RXIE is set.
7. The current object pointer MOFGPRs.CUR of the gateway source object is moved to the next destination object according to the FIFO rules as described on [Page 18-46](#). A gateway with a single (static) destination object is obtained by setting MOFGPRs.TOP = MOFGPRs.BOT = MOFGPRs.CUR = destination object.

The link from the gateway source object to the gateway destination object works in the same way as the link from a FIFO base to a FIFO slave. This means that a gateway with an integrated destination FIFO may be created; in [Figure 18-20](#), the object on the left is the gateway source object and the message object on the right side is the gateway destination objects.

The gateway operates equivalent for the reception of data frames (source object is receive object, i.e. DIR = 0) as well as for the reception of Remote Frames (source object is transmit object).

Controller Area Network Controller (MultiCAN+)

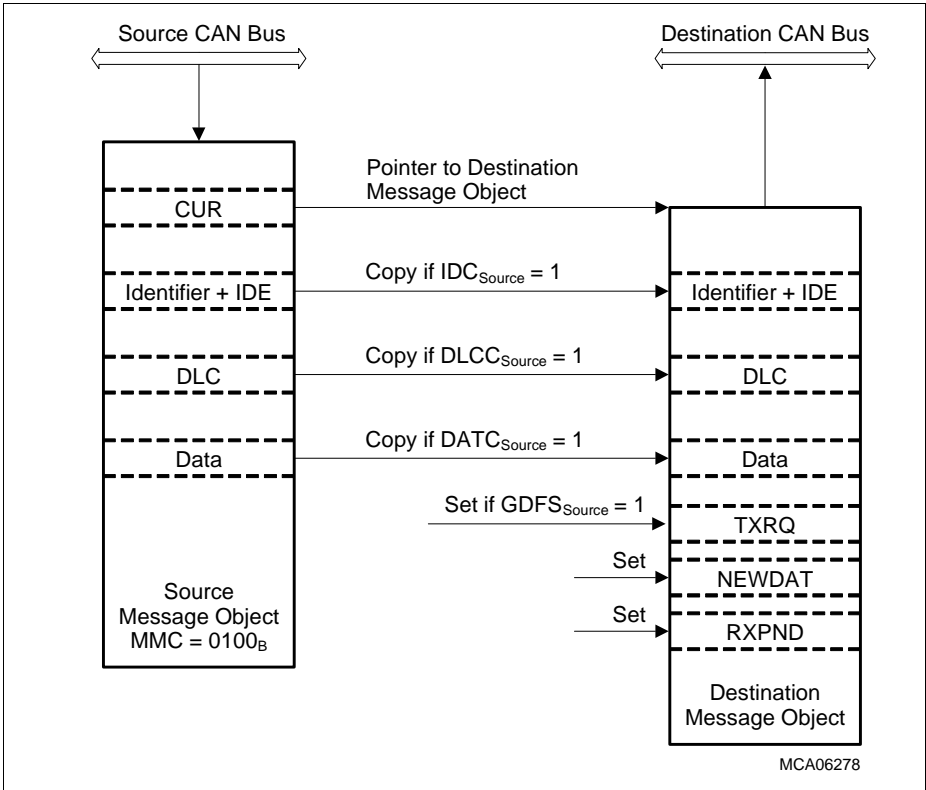


Figure 18-21 Gateway Transfer from Source to Destination

18.3.10.8 Foreign Remote Requests

When a Remote Frame has been received on a CAN node and is stored in a message object, a transmit request is set to trigger the answer (transmission of a Data Frame) to the request or to automatically issue a secondary request. If the Foreign Remote Request Enable bit MOFCRn.FRREN is cleared in the message object in which the remote request is stored, MOSTATn.TXRQ is set in the same message object.

If bit FRREN is set (FRREN = 1: foreign remote request enabled), TXRQ is set in the message object that is referenced by pointer MOFGPRn.CUR. The value of CUR is, however, not changed by this feature.

Although the foreign remote request feature works independently of the selected message mode, it is especially useful for gateways to issue a remote request on the source bus of a gateway after the reception of a remote request on the gateway destination bus. According to the setting of FRREN in the gateway destination object, there are two capabilities to handle remote requests that appear on the destination side (assuming that the source object is a receive object and the destination is a transmit object, i.e. $DIR_{source} = 0$ and $DIR_{destination} = 1$):

FRREN = 0 in the Gateway Destination Object

1. A Remote Frame is received by gateway destination object.
2. TXRQ is set automatically in the gateway destination object.
3. A Data Frame with the current data stored in the destination object is transmitted on the destination bus.

FRREN = 1 in the Gateway Destination Object

1. A Remote Frame is received by gateway destination object.
2. TXRQ is set automatically in the gateway source object (must be referenced by CUR pointer of the destination object).
3. A remote request is transmitted by the source object (which is a receive object) on the source CAN bus.
4. The receiver of the remote request responds with a Data Frame on the source bus.
5. The Data Frame is stored in the source object.
6. The Data Frame is copied to the destination object (gateway action).
7. TXRQ is set in the destination object (assuming $GDFS_{source} = 1$).
8. The new data stored in the destination object is transmitted on the destination bus, in response to the initial remote request on the destination bus.

18.4 Use Case Example MultiCAN+

This section explains the core functionality of the MultiCAN+ module with a code example. The example realizes sending a CAN message via internal CAN-bus from node 0 to node 1.

The MultiCAN+ module for the XMC4300 consists of module (i.e MultiCAN with 2 CAN nodes), representing serial communication interfaces. Each CAN node can either be connected to a port or to the internal CAN bus (see [Figure 18-24](#)). So a maximum of 2 CAN channels can be realized with the MultiCAN+ module (For more information and further functions see [Section 18.2](#)).

All CAN nodes share a common set of 64 message objects. A message object function like a container for a specific CAN message; both transmitting and receiving of CAN messages can be realized. The message objects are organized in double-chained lists, each list is dedicated to a certain CAN node (list 1 is node 0, list 2 is node 1, etc.). After a reset, all message objects are unallocated and therefore assigned to list 0, this list does not belong to a CAN node. During initialization it is possible to allocate the message objects individually to certain CAN node lists. Each allocated message object can be set up with all necessary information like the message ID etc. (see chapter 26.2 and following for further explanations of the CAN module).

In the use case example CAN node 0 will send a CAN message via internal CAN bus to CAN node 1. The transmitting message object (MO) will be MO 0 and the receiving message object will be MO 1. As soon as the CAN frame successfully receives at MO 1 a receive interrupt will occur. The initialization process follows the following order:

1. Load global MultiCAN+ module registers
2. Initialize the CAN nodes
3. Allocate the message objects to the CAN nodes
4. Initialize the message objects
5. Start the CAN nodes
6. Start transmit request for CAN message from node 0 to node 1.
7. Receive message at node 1, Rx interrupt occurs.

Step description to initialize the MultiCAN+ module:

(Line 2) enable control of the module, in clock control register CLC (**CLC**).

(Line 3) read back (dummy variable has to be defined). The reading process ensures that the write process from line 2 is done.

(Line 4) load fractional divider to $f_{CAN} = f_{PERIPH}$ (see also 26.3.2 Clock Control (**Fractional Divider**), f_{PERIPH} see **MCR**).

(Line 5) read back (dummy variable to be defined).

(Line 6) set clk source to f_{PERIPH} in module control register MCR(**MCR**).

Controller Area Network Controller (MultiCAN+)

Note: The reconfiguration of the clk source must be done by two writes and a certain delay between these. See MCR for more details.

(Line 7) The setting of protection CCE[6] and INIT[0] activates the initialization and configuration mode for CAN node 0. This is necessary for the upcoming changes in CAN node 0. (see also node configuration register NCRn (**CAN_NCRx (x = 0-1)**))

(Line 8) This example uses the internal loop-back mode, so this line connects CAN node 0 to the internal CAN Bus. (see also Figure 26-13(**Figure 18-13**) and node port control register NPCR(**CAN_NPCRx (x = 0-1)**))

(Line 9) The CAN bit time is subdivided into different segments. (see also 26.3.6.1 Bit Timing Unit(**Bit Timing Unit**)). Assuming $f_{CAN} = 100$ MHz this line then sets the baudrate to 500 kBaud and the sample point to 80% in the node bit timing register NBTR (**CAN_NBTRx (x = 0-1)**).

(Line 10 - 12) same as line 7 - 9 but with node 1.

(Line 13) The allocation of the message objects to certain CAN node lists function via a list command panel. The panel control register PANCTR(**PANCTR**) can only be written if both busy flags are reset. Therefore this while loop waits until the register is ready.

(Line 14) This line allocates message object 0 to list 1 (belongs to CAN node 0).

(Line 15) see line 13.

(Line 16) This line allocates message object 1 to list 2 (belongs to CAN node 0).

(Line 17) Initialize MO 0 in the message object register MOCTR(**CAN_MOCTRz (z = 0-62)**) as transmit MO by setting bits [27:25]. Set also MSGVAL[21], that's the main switch bit of the MO.

(Line 18) this line sets the message data length of MO 0 to 8 bytes.(see also message object function control register MOFCR (**CAN_MOFCRn (n = 0-63)**)).

(Line 19) the message ID of MO 0 gets set to standard message (11 bit length) with a message ID of FF_H in the Message object arbitration register MOAR (**CAN_MOARn (n = 0-63)**).

(Line 20) Initialize MO 1 in the message object register MOCTR as receive MO by setting bit RXEN[23]. Set also MSGVAL[21], that's the main switch bit of the MO.

(Line 21) the receive interrupt gets enabled in the message object function control register MOFCR .

(Line 22) same as Line 19 but for MO 1. The same message ID is necessary to receive CAN messages from the bus with this specific ID.

(Line 23) the Rx interrupt gets connected to the CAN interrupt output line 1.(see also MOIPR(**CAN_MOIPRn (n = 0-63)**))

(Line 24) This line enables the Rx interrupt in the service request control register SRC_CANINT1 and sets the interrupt priority to CAN_SR1INT (1...255)

(Line 25) This function sets the interrupt priority of the receive interrupt.

Controller Area Network Controller (MultiCAN+)

(Line 26) The 4 lower data bytes are getting loaded in the MODATAL(**CAN_MODATALn (n = 0-63)**) register. the data itself is here just an example.

(Line 27) The 4 higher data bytes are getting loaded in the MODATAH(**CAN_MODATAHn (n = 0-63)**) register. (just an data example as well)

(Line 28) This line resets INIT[0] and CCE[6] bit in the node control register from node 0. The node is now synchronizing itself to the bus.

(Line 29) same as line 28, but with node 1.

(Line 30) The NEWDATA bit gets set in the MOCTR register, this should always be done after a write process in the data registers from line 26/27. The transmit request bit TXRQ gets set, this starts the transmission of the CAN message. The RTSEL gets reset to ensure the transmission.

(Line 31) If the message is received the MOCTR.NEWDAT[19] bit in the receive MO 1 is set. This line just waits for the reception. The occurrence of the Rx interrupt can also be used as a proof that the message arrived.

(Line 32/33) Access the data bytes without clearing NEWDAT.

Note: The Rx ISR function prototype would be: void CAN1_Rx_irq (void); here could be your ISR code;

Note: Line 1 does not apply for ARM products, therefore this line does not exist.

Initialization of the MultiCAN+ module:

```
// Load global MultiCAN+ registers:
(2) CAN_CLC = 0x000;           // enable module control
(3) dummy = CAN_CLC;         //read to check the made changes
(4) CAN_FDR = (1 << 14) | 0x3FF; //DM=1,STEP=1023

(5) dummy = CAN_FDR;         //read to check the made changes
(6) CAN_MCR = 0x1;           // CLKSEL=1
//Init CAN nodes 0 and 1:
(7) CAN_NCR0 = (1 << 6) | 1; // CCE=1,INIT=1
(8) CAN_NPCR0 = (1 << 8);    // LBM=1
(9) CAN_NBTR0 = 0x3EC9;     //set bit segments
(10) CAN_NCR1 = (1 << 6) | 1; // CCE=1,INIT=1
(11) CAN_NPCR1 = (1 << 8);   // LBM=1
(12) CAN_NBTR1 = 0x3EC9;    //set bit segments
//Allocate message objects to CAN nodes:
(13) while(CAN_PANCTR.U & (0x00000100 | 0x00000200)); // busy?
(14) CAN_PANCTR = (1 << 24) | (0 << 16) | 2; // MO 0 to list 1
(15) while(CAN_PANCTR.U & (0x00000100 | 0x00000200)); // busy?
(16) CAN_PANCTR = (2 << 24) | (1 << 16) | 2; // MO 1 to list 2
//Init MO_0 (list 1, node 0)
```

Controller Area Network Controller (MultiCAN+)

```

(17) CAN_MOCTR0 = (1<<27)|(1<<26)|(1<<25)|(1<<21); // set to Tx
(18) CAN_MOFCR0 = (8 << 24); // DLC=8
(19) CAN_MOAR0 = (1 << 30)|(0xFF << 18); // PRI=01, ID=0xFF
//Init MO_1 (list 2, node 1)
(20) CAN_MOCTR1 = (1 << 23)|(1 << 21); // set to Rx
(21) CAN_MOFCR1 = (1 << 16); // RXIE=1
(22) CAN_MOAR1 = (1 << 30)|(0xFF << 18); // PRI=01, ID=0xFF
(23) CAN_MOIPR1 = 0x1; // RXINP=1 -> select INT_01
(24) SRC_CANINT1 = (1 << 10) | CAN_SRN1INT; // enable INT_01
(25) NVIC_SetPriority (CAN_SRN1INT, & CAN1_Rx_irq);
// Load Data, Start CAN nodes
(26) CAN_MODATAL0 = 0x0D0D0D0D; // load data, lower 4 Bytes
(27) CAN_MODATAH0 = 0x0E0E0E0E; // load data, higher 4 Bytes
(28) CAN_NCR0 &= ~(1<<6)| 1; // reset CCE and INIT
(29) CAN_NCR1 &= ~(1<<6)| 1; // reset CCE and INIT
// set transmit request to send message
(30) CAN_MOCTR0 = (1<<24)|(1<<19)|(1<<6); //TXRQ=1,NEWDAT=1
(31) while((CAN_MOSTAT1.B.NEWDAT) != 1); //check if Rx
(32) readfirstfourdatabytes1=CAN_MODATAL1; //read_receive
(33) readnextfourdatabytes1=CAN_MODATAH1; //read_receive

```

Controller Area Network Controller (MultiCAN+)

18.5 MultiCAN+ Kernel Registers

This section describes the kernel registers of the MultiCAN+ module. All MultiCAN+ kernel register names described in this section are also referenced in other parts of the XMC4300 Reference Manual by the module name prefix "CAN_".

MultiCAN+ Kernel Register Overview

The MultiCAN+ Kernel include three blocks of registers:

- Global Module Registers
- Node Registers, for each CAN node x
- Message Object Registers, for each message object n

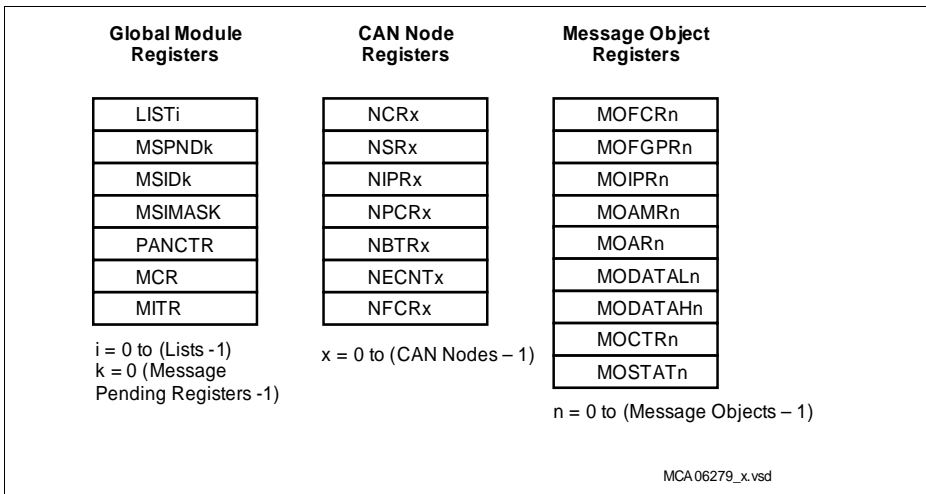


Figure 18-22 MultiCAN+ Kernel Registers

The registers of the MultiCAN+ module kernel are listed below.

Table 18-6 Registers Address Space - MultiCAN+ Kernel Registers

Module	Base Address	End Address	Note
CAN	4801 4000 _H	4801 7FFF _H	-

Controller Area Network Controller (MultiCAN+)

Table 18-7 Registers Overview - MultiCAN+ Kernel Registers

Short Name	Description	Offset Addr ¹⁾	Access Mode ²⁾		Reset	Description see
			Read	Write		
Global Module Registers						
LISTi	List Register i	0100 _H + i × 4 _H	U, PV	U, PV	Application Reset	Page 18-70
MSPNDk	Message Pending Register k	0140 _H + k × 4 _H	U, PV	U, PV	Application Reset	Page 18-72
MSIDk	Message Index Register k	0180 _H + k × 4 _H	U, PV	U, PV	Application Reset	Page 18-73
MSIMASK	Message Index Mask Register	01C0 _H	U, PV	U, PV	Application Reset	Page 18-74
PANCTR	Panel Control Register	01C4 _H	U, PV	U, PV	Application Reset	Page 18-63
MCR	Module Control Register	01C8 _H	U, PV	U, PV	Application Reset	Page 18-67
MITR	Module Interrupt Trigger Reg.	01CC _H	U, PV	U, PV	Application Reset	Page 18-69
CAN Node Registers						
NCRx	Node x Control Register	0200 _H + x × 100 _H	U, PV	U, PV	Application Reset	Page 18-75
NSRx	Node x Status Register	0204 _H + x × 100 _H	U, PV	U, PV	Application Reset	Page 18-78
NIPRx	Node x Interrupt Pointer Reg.	0208 _H + x × 100 _H	U, PV	U, PV	Application Reset	Page 18-81

Controller Area Network Controller (MultiCAN+)

Table 18-7 Registers Overview - MultiCAN+ Kernel Registers (cont'd)

Short Name	Description	Offset Addr ¹⁾	Access Mode ²⁾		Reset	Description see
			Read	Write		
NPCR _x	Node x Port Control Register	020C _H + x × 100 _H	U, PV	U, PV	Application Reset	Page 18-83
NBTR _x	Node x Bit Timing Register	0210 _H + x × 100 _H	U, PV	U, PV	Application Reset	Page 18-84
NECNT _x	Node x Error Counter Register	0214 _H + x × 100 _H	U, PV	U, PV	Application Reset	Page 18-86
NFCR _x	Node x Frame Counter Register	0218 _H + x × 100 _H	U, PV	U, PV	Application Reset	Page 18-87

Message Object Registers

MOFCR _n	Message Object n Function Control Register	1000 _H + n × 20 _H	U, PV	U, PV	Application Reset	Page 18-102
MOFGPR _n	Message Object n FIFO/Gateway Pointer Register	1004 _H + n × 20 _H	U, PV	U, PV	Application Reset	Page 18-106
MOIPR _n	Message Object n Interrupt Pointer Register	1008 _H + n × 20 _H	U, PV	U, PV	Application Reset	Page 18-100
MOAMR _n	Message Object n Acceptance Mask Register	100C _H + n × 20 _H	U, PV	U, PV	Application Reset	Page 18-107
MODATAL _n	Message Object n Data Register Low	1010 _H + n × 20 _H	U, PV	U, PV	Application Reset	Page 18-111
MODATAH _n	Message Object n Data Register High	1014 _H + n × 20 _H	U, PV	U, PV	Application Reset	Page 18-112

Controller Area Network Controller (MultiCAN+)

Table 18-7 Registers Overview - MultiCAN+ Kernel Registers (cont'd)

Short Name	Description	Offset Addr ¹⁾	Access Mode ²⁾		Reset	Description see
			Read	Write		
MOARn	Message Object n Arbitration Register	$1018_H + n \times 20_H$	U, PV	U, PV	Application Reset	Page 18-108
MOCTRn MOSTATn	Message Object n Control Reg. Message Object n Status Reg.	$101C_H + n \times 20_H$	U, PV	U, PV	Application Reset	Page 18-92 Page 18-95

1) The absolute register address is calculated as follows:

Module Base Address ([Table 18-6](#)) + Offset Address (shown in this column)

Further, the following ranges for parameters i, k, x, and n are valid: i = 0-15, k = 0-7, x = 0-1, n = 0-63.

2) Accesses to empty addresses: nBE

Controller Area Network Controller (MultiCAN+)

Figure 18-23

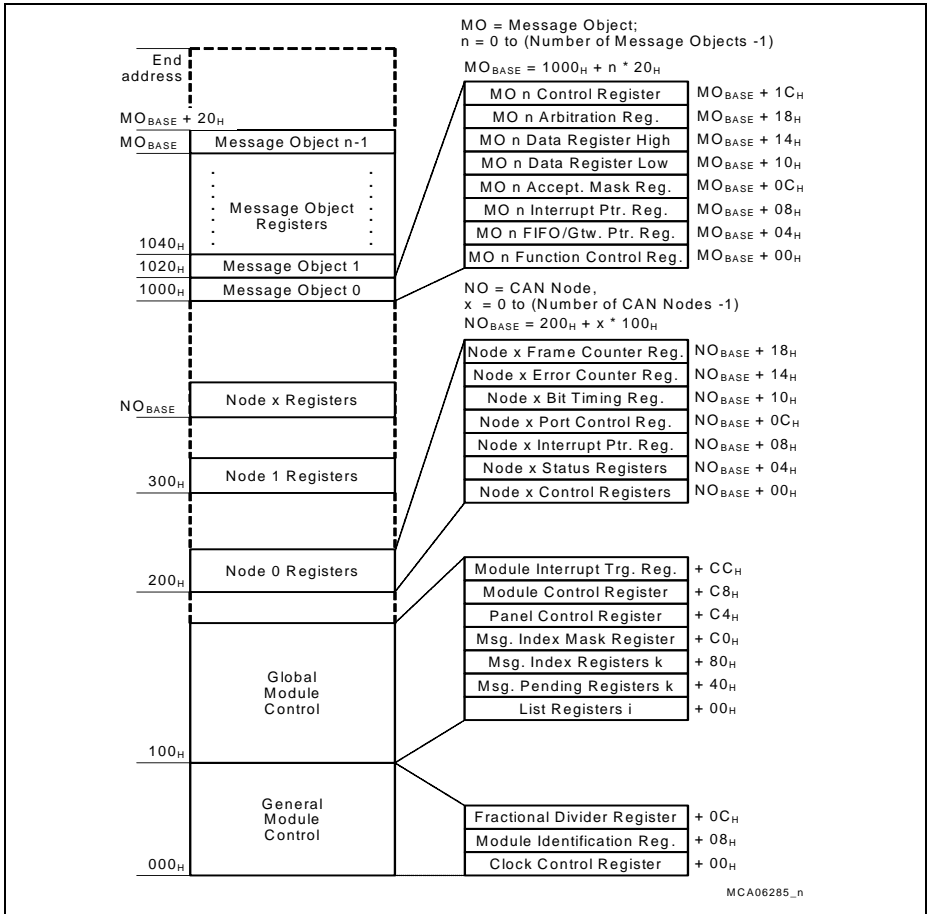


Figure 18-23 MultiCAN+ Register Address Map XMC

18.5.1 Global Module Registers

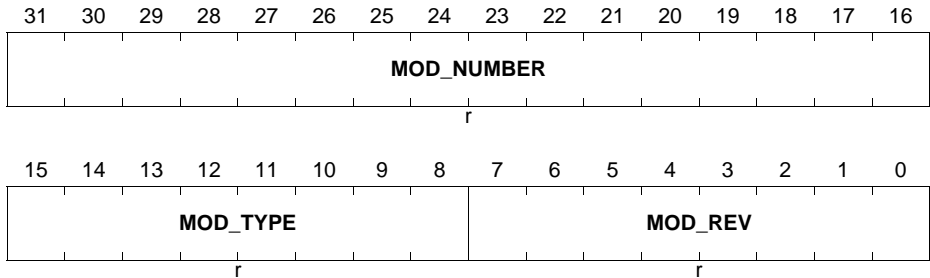
All list operations such as allocation, de-allocation and relocation of message objects within the list structure are performed via the Command Panel. It is not possible to modify the list structure directly by software by writing to the message objects and the LIST registers.

Module Identification Register.

Controller Area Network Controller (MultiCAN+)

ID

Module Identification Register (008_H) **Reset Value: 00B5 C0XX_H**



Field	Bits	Type	Description
MOD_REV	[7:0]	r	Module Revision Number MOD_REV defines the revision number. The value of a module revision starts with 01 _H (first revision).
MOD_TYPE	[15:8]	r	Module Type C0 _H Define the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	Module Number Value This bit field defines the MultiCAN+ module identification number (=00B5H)

Controller Area Network Controller (MultiCAN+)

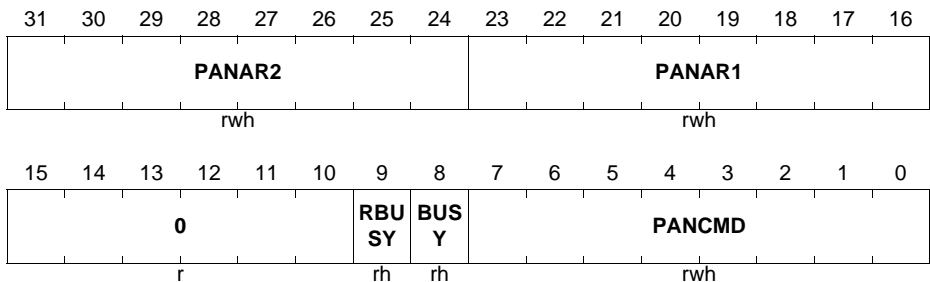
The Panel Control Register PANCTR is used to start a new command by writing the command arguments and the command code into its bit fields.

PANCTR

Panel Control Register

(1C4_H)

Reset Value: 0000 0301_H



Field	Bits	Type	Description
PANCMD	[7:0]	rwh	Panel Command This bit field is used to start a new command by writing a panel command code into it. At the end of a panel command, the NOP (no operation) command code is automatically written into PANCMD. The coding of PANCMD is defined in Table 18-8 .
BUSY	8	rh	Panel Busy Flag 0 _B Panel has finished command and is ready to accept a new command. 1 _B Panel operation is in progress. Initial list controller initialization must be finalized, when INIT bit is reset.
RBUSY	9	rh	Result Busy Flag 0 _B No update of PANAR1 and PANAR2 is scheduled by the list controller. 1 _B A list command is running (BUSY = 1) that will write results to PANAR1 and PANAR2, but the results are not yet available.
PANAR1	[23:16]	rwh	Panel Argument 1 See Table 18-8 .
PANAR2	[31:24]	rwh	Panel Argument 2 See Table 18-8 .

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
0	[15:10]	r	Reserved Read as 0; should be written with 0.

Panel Commands

A panel operation consists of a command code (PANCMD) and up to two panel arguments (PANAR1, PANAR2). Commands that have a return value deliver it to the PANAR1 bit field. Commands that return an error flag deliver it to bit 31 of the Panel Control Register, this means bit 7 of PANAR2.

Table 18-8 Panel Commands

PANCMD	PANAR2	PANAR1	Command Description
00 _H	–	–	No Operation Writing 00 _H to PANCMD has no effect. No new command is started.
01 _H	Result: Bit 7: ERR Bit 6-0: undefined	–	Initialize Lists Run the initialization sequence to reset the CTRL and LIST fields of all message objects. List registers LIST[7:0] are set to their reset values. This results in the de-allocation of all message objects. The initialization command requires that bits NCRx.INIT and NCRx.CCE are set for all CAN nodes. Bit 7 of PANAR2 (ERR) reports the success of the operation: 0 _B Initialization was successful 1 _B Not all NCRx.INIT and NCRx.CCE bits are set. Therefore, no initialization is performed. The initialize lists command is automatically performed with each reset of the MultiCAN+ module, but with the exception that all message object registers are reset, too.

Controller Area Network Controller (MultiCAN+)

Table 18-8 Panel Commands (cont'd)

PANCMD	PANAR2	PANAR1	Command Description
02 _H	Argument: List Index	Argument: Message Object Number	Static Allocate Allocate message object to a list. The message object is removed from the list that it currently belongs to, and appended to the end of the list, given by PANAR2. This command is also used to deallocate a message object. In this case, the target list is the list of unallocated elements (PANAR2 = 0).
03 _H	Argument: List Index Result: Bit 7: ERR Bit 6-0: undefined	Result: Message Object Number	Dynamic Allocate Allocate the first message object of the list of unallocated objects to the selected list. The message object is appended to the end of the list. The message number of the message object is returned in PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0 _B Success. 1 _B The operation has not been performed because the list of unallocated elements was empty.
04 _H	Argument: Destination Object Number	Argument: Source Object Number	Static Insert Before Remove a message object (source object) from the list that it currently belongs to, and insert it before a given destination object into the list structure of the destination object. The source object thus becomes the predecessor of the destination object.

Controller Area Network Controller (MultiCAN+)

Table 18-8 Panel Commands (cont'd)

PANCMD	PANAR2	PANAR1	Command Description
05 _H	Argument: Destination Object Number Result: Bit 7: ERR Bit 6-0: undefined	Result: Object Number of inserted object	Dynamic Insert Before Insert a new message object before a given destination object. The new object is taken from the list of unallocated elements (the first element is chosen). The number of the new object is delivered as a result to PANAR1. An ERR bit (bit 7 of PANAR1) reports the success of the operation: 0 _B Success. 1 _B The operation has not been performed because the list of unallocated elements was empty.
06 _H	Argument: Destination Object Number	Argument: Source Object Number	Static Insert Behind Remove a message object (source object) from the list that it currently belongs to, and insert it behind a given destination object into the list structure of the destination object. The source object thus becomes the successor of the destination object.
07 _H	Argument: Destination Object Number Result: Bit 7: ERR Bit 6-0: undefined	Result: Object Number of inserted object	Dynamic Insert Behind Insert a new message object behind a given destination object. The new object is taken from the list of unallocated elements (the first element is chosen). The number of the new object is delivered as result to PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0 _B Success. 1 _B The operation has not been performed because the list of unallocated elements was empty.
08 _H - FF _H	–	–	Reserved

Controller Area Network Controller (MultiCAN+)

The Module Control Register MCR contains basic settings that determine the operation of the MultiCAN+ module.

The write access to the lowest byte of the MCR register is possible only if the CCE bits of all CAN nodes are set (NCRx.CCE bits). The NCRx.INIT bits will be automatically set when the lowest byte of the MCR register is written, independent of the setting of the CCE bits. The INIT bits have to be reset by software in order to activate the CAN nodes.

The reconfiguration of the clock source has to be done by using two writes: first a write of zero to the CLKSEL bit field, and then a second write defining the new clock source. Between the first and the second write a delay of $4 / f_A + 2 / f_{CAN}$ number of cycles must be inserted by software, where f_A is the frequency being switched off with the first write. Exception: in case that f_{PERIPH} is selected as the baud rate logic clock (MCR.CLKSEL = 1), no delay cycles between the writes are necessary. In both cases, simply using one write defining the new clock source is not allowed.

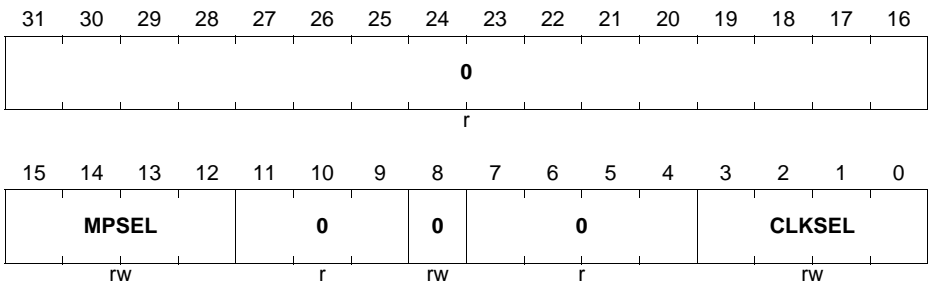
Note: If the baud rate logic is supplied from an unstable clock source, or no clock at all, the CAN functionality is not guaranteed.

MCR

Module Control Register

(1C8_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
CLKSEL	[3:0]	rw	Baud Rate Logic Clock Select 0000 _B No clock supplied 0001 _B f_{PERIPH} 0010 _B 0100 _B not allowed 1000 _B hard wired to 0 ... _B not allowed
0	8	rw	Reserved Read as 0; should be written with 0.

Controller Area Network Controller (MultiCAN+)

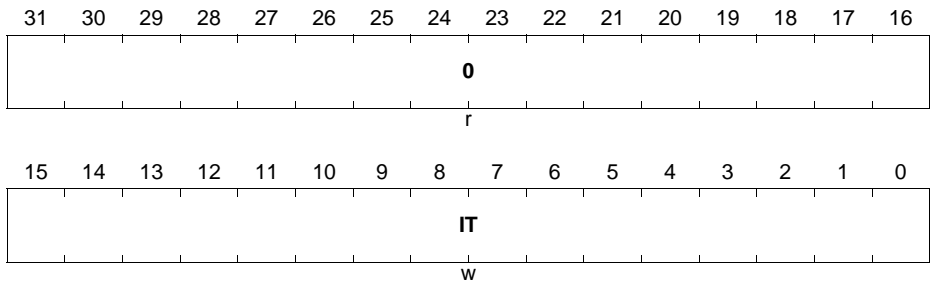
Field	Bits	Type	Description
MPSEL	[15:12]	rw	Message Pending Selector Bit field MPSEL makes it possible to select the bit position of the message pending bit after a message reception/transmission by a mixture of the MOIPRn register bit fields RXINP, TXINP, and MPN. Selection details are given in Figure 18-17 on Page 18-37 .
0	[31:16], [11:9], [7:4]	r	Reserved Read as 0; should be written with 0.

Controller Area Network Controller (MultiCAN+)

The Interrupt Trigger Register ITR is used to trigger interrupt requests on each interrupt output line by software.

MITR

Module Interrupt Trigger Register (1CC_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
IT	[7:0]	w	Interrupt Trigger Writing a 1 to IT[m] (m = 0-7) generates an interrupt request on interrupt output line INT_O[m]. Writing a 0 to IT[m] has no effect. Bit field IT is always read as 0. Multiple interrupt requests can be generated with a single write operation to MITR by writing a 1 to several bit positions of IT. All 16 interrupts are existing, even if the interrupt request unit is not connected.
0	[31:8]	r	Reserved Read as 0; should be written with 0.

Controller Area Network Controller (MultiCAN+)

List Pointer and List Register

Each CAN node has a list that determines the allocated message objects. Additionally, a list of all unallocated objects is available. Furthermore, general purpose lists are available which are not associated to a CAN node. The List Registers are assigned in the following way:

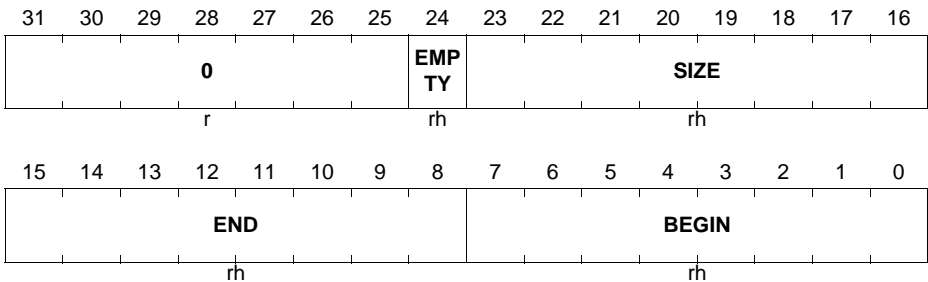
- LIST0 provides the list of all unallocated objects
- LIST1 provides the list for CAN node 0
- LIST2 provides the list for CAN node 1
- ...
- LIST \underline{n} provides the list for CAN node \underline{n}

LIST0

List Register 0 (100_H) **Reset Value: 003F 3F00_H**

LISTn (n = 1-15)

List Register n (100_H+n*4_H) **Reset Value: 0100 0000_H**



Field	Bits	Type	Description
BEGIN	[7:0]	rh	List Begin BEGIN indicates the number of the first message object in list i.
END	[15:8]	rh	List End END indicates the number of the last message object in list i.
SIZE	[23:16]	rh	List Size SIZE indicates the number of elements in the list i. SIZE = number of list elements - 1 SIZE = 0 indicates that list i is empty.

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
EMPTY	24	rh	List Empty Indication 0 _B At least one message object is allocated to list i. 1 _B No message object is allocated to the list i. List i is empty.
0	[31:25]	r	Reserved Read as 0.

Controller Area Network Controller (MultiCAN+)

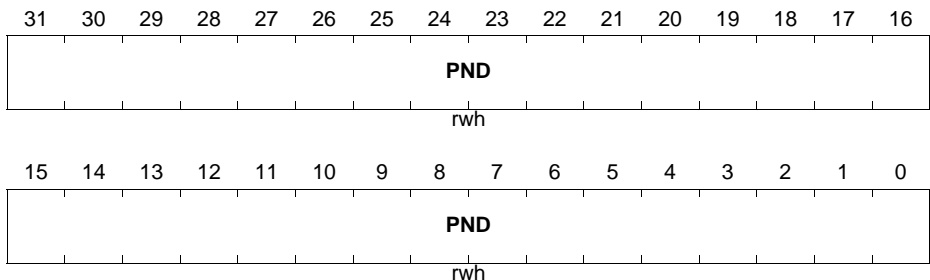
Message Notifications

When a message object n generates an interrupt request upon the transmission or reception of a message, then the request is routed to the interrupt output line selected by the bit field $MOIPRn.TXINP$ or $MOIPRn.RXINP$ of the message object n . As there are more message objects than interrupt output lines, an interrupt routine typically processes requests from more than one message object. Therefore, a priority selection mechanism is implemented in the MultiCAN+ module to select the highest priority object within a collection of message objects.

The Message Pending Register $MSPNDk$ contains the pending interrupt notification of list i .

MSPNDk (k = 0-7)

Message Pending Register k **(140_H+k*4_H)** **Reset Value: 0000 0000_H**



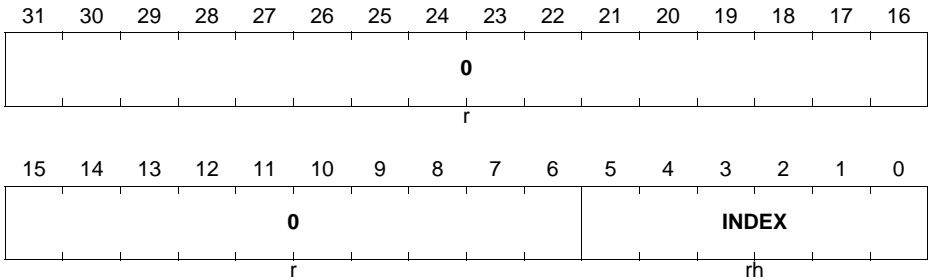
Field	Bits	Type	Description
PND	[31:0]	rwh	Message Pending When a message interrupt occurs, the message object sets a bit in one of the MSPND register, where the bit position is given by the MPN[4:0] field of the IPR register of the message object. The register selection n is given by the higher bits of MPN. The register bits can be cleared by software (write 0). Writing a 1 has no effect.

Controller Area Network Controller (MultiCAN+)

Each Message Pending Register has a Message Index Register MSID_k associated with it. The Message Index Register shows the active (set) pending bit with lowest bit position within groups of pending bits.

MSID_k (k = 0-7)

Message Index Register k **(180_H+k*4_H)** **Reset Value: 0000 0020_H**



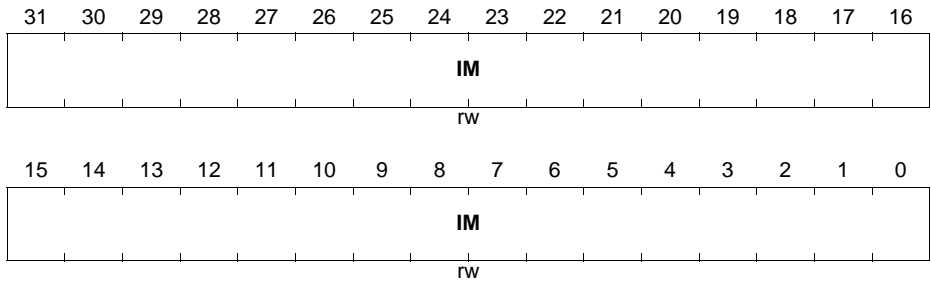
Field	Bits	Type	Description
INDEX	[5:0]	rh	<p>Message Pending Index</p> <p>The value of INDEX is given by the bit position i of the pending bit of MSPNDk with the following properties:</p> <ol style="list-style-type: none"> MSPNDk[i] & IM[i] = 1 i = 0 or MSPNDk[i-1:0] & IM[i-1:0] = 0 <p>If no bit of MSPNDk satisfies these conditions then INDEX reads 100000_B.</p> <p>Thus INDEX shows the position of the first pending bit of MSPNDk, in which only those bits of MSPNDk that are selected in the Message Index Mask Register are taken into account.</p>
0	[31:6]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Controller Area Network Controller (MultiCAN+)

The Message Index Mask Register MSIMASK selects individual bits for the calculation of the Message Pending Index. The Message Index Mask Register is used commonly for all Message Pending registers and their associated Message Index registers.

MSIMASK

Message Index Mask Register (1C0_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
IM	[31:0]	rw	Message Index Mask Only those bits in MSPNDk for which the corresponding Index Mask bits are set contribute to the calculation of the Message Index.

18.5.2 CAN Node Registers

The CAN node registers are built in for each CAN node of the MultiCAN+ module. They contain information that is directly related to the operation of the CAN nodes and are shared among the nodes.

The Node Control Register contains basic settings that determine the operation of the CAN node.

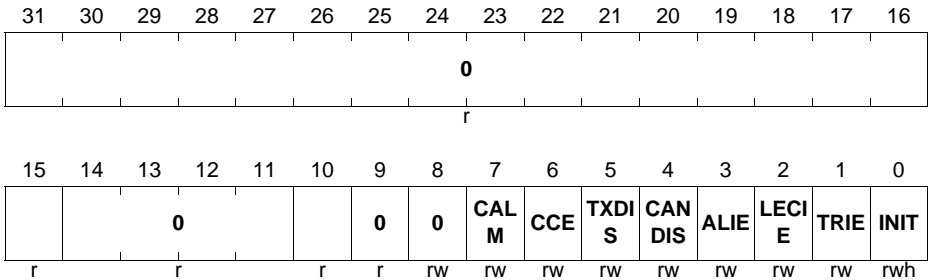
Controller Area Network Controller (MultiCAN+)

CAN_NCR_x (x = 0-1)

Node x Control Register

(200_H+x*100_H)

Reset Value: 0000 0041_H



Field	Bits	Type	Description
INIT	0	rwh	<p>Node Initialization</p> <p>0_B Resetting bit INIT enables the participation of the node in the CAN traffic. If the CAN node is in the bus-off state, the ongoing bus-off recovery (which does not depend on the INIT bit) is continued. With the end of the bus-off recovery sequence the CAN node is allowed to take part in the CAN traffic. If the CAN node is not in the bus-off state, a sequence of 11 consecutive recessive bits must be detected before the node is allowed to take part in the CAN traffic.</p> <p>1_B Setting this bit terminates the participation of this node in the CAN traffic. Any ongoing frame transfer is cancelled and the transmit line goes recessive. If the CAN node is in the bus-off state, then the running bus-off recovery sequence is continued. If the INIT bit is still set after the successful completion of the bus-off recovery sequence, i.e. after detecting 128 sequences of 11 consecutive recessive bits (11 × 1), then the CAN node leaves the bus-off state but remains inactive as long as INIT remains set.</p> <p>Bit INIT is automatically set when the CAN node enters the bus-off state (see Page 18-22).</p>

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
TRIE	1	rw	<p>Transfer Interrupt Enable</p> <p>TRIE enables the transfer interrupt of CAN node x. This interrupt is generated after the successful reception or transmission of a CAN frame in node x.</p> <p>0_B Transfer interrupt is disabled. 1_B Transfer interrupt is enabled.</p> <p>Bit field NIPRx.TRINP selects the interrupt output line which becomes activated at this type of interrupt.</p>
LECIE	2	rw	<p>LEC Indicated Error Interrupt Enable</p> <p>LECIE enables the last error code interrupt of CAN node x. This interrupt is generated with each hardware update of bit field NSRx.LEC with LEC > 0 (CAN protocol error).</p> <p>0_B Last error code interrupt is disabled. 1_B Last error code interrupt is enabled.</p> <p>Bit field NIPRx.LECINP selects the interrupt output line which becomes activated at this type of interrupt.</p>
ALIE	3	rw	<p>Alert Interrupt Enable</p> <p>ALIE enables the alert interrupt of CAN node x. This interrupt is generated by any one of the following events:</p> <ul style="list-style-type: none"> • A change of bit NSRx.BOFF • A change of bit NSRx.EWRN • A List Length Error, which also sets bit NSRx.LLE • A List Object Error, which also sets bit NSRx.LOE <p>0_B Alert interrupt is disabled. 1_B Alert interrupt is enabled.</p> <p>Bit field NIPRx.ALINP selects the interrupt output line which becomes activated at this type of interrupt.</p>
CANDIS	4	rw	<p>CAN Disable</p> <p>Setting this bit disables the CAN node. The CAN node first waits until it is bus-idle or bus-off. Then bit NCRx.INIT is automatically set, and an alert interrupt is generated if bit ALIE is set.</p>
TXDIS	5	rw	<p>Transmit Disable</p> <p>Setting this bit disables the transmission on CAN node x as soon as bus-idle is reached. Reception and bits in MOSTATn, e.g. TXRQ, will not be influenced.</p>

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
CCE	6	rw	<p>Configuration Change Enable</p> <p>0_B The Bit Timing Register, the Port Control Register, and the Error Counter Register may only be read. All attempts to modify them are ignored.</p> <p>1_B The Bit Timing Register, the Port Control Register, and the Error Counter Register may be read and written.</p>
CALM	7	rw	<p>CAN Analyzer Mode</p> <p>If this bit is set, then the CAN node operates in Analyzer Mode. This means that messages may be received, but not transmitted. No acknowledge is sent on the CAN bus upon frame reception. Active-error flags are sent recessive instead of dominant. The transmit line is continuously held at recessive (1) level. Bit CALM can be written only while bit INIT is set.</p>
0	8	rw	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Controller Area Network Controller (MultiCAN+)

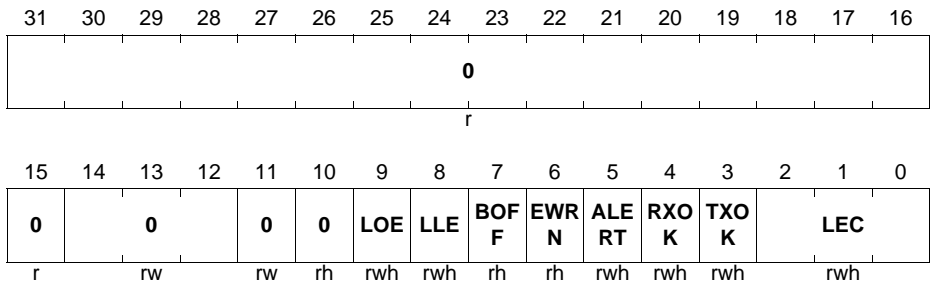
The Node Status Register NSRx reports errors as well as successfully transferred CAN frames.

CAN_NSRx (x = 0-1)

Node x Status Register

(204_H+x*100_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
LEC	[2:0]	rwh	Last Error Code This bit field indicates the type of the last (most recent) CAN error. The encoding of this bit field is described in Table 18-9 .
TXOK	3	rwh	Message Transmitted Successfully 0 _B No successful transmission since last (most recent) flag reset. 1 _B A message has been transmitted successfully (error-free and acknowledged by at least another node). TXOK must be reset by software (write 0). Writing 1 has no effect.
RXOK	4	rwh	Message Received Successfully 0 _B No successful reception since last (most recent) flag reset. 1 _B A message has been received successfully. RXOK must be reset by software (write 0). Writing 1 has no effect.

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
ALERT	5	rwh	<p>Alert Warning</p> <p>The ALERT bit is set upon the occurrence of one of the following events (the same events which also trigger an alert interrupt if ALIE is set):</p> <ul style="list-style-type: none"> • A change of bit NSRx.BOFF • A change of bit NSRx.EWRN • A List Length Error, which also sets bit NSRx.LLE • A List Object Error, which also sets bit NSRx.LOE <p>ALERT must be reset by software (write 0). Writing 1 has no effect.</p>
EWRN	6	rh	<p>Error Warning Status</p> <p>0_B No warning limit exceeded. 1_B One of the error counters REC or TEC reached the warning limit EWRNLVL.</p>
BOFF	7	rh	<p>Bus-off Status</p> <p>0_B CAN controller is not in the bus-off state. 1_B CAN controller is in the bus-off state.</p>
LLE	8	rwh	<p>List Length Error</p> <p>0_B No List Length Error since last (most recent) flag reset. 1_B A List Length Error has been detected during message acceptance filtering. The number of elements in the list that belongs to this CAN node differs from the list SIZE given in the list termination pointer.</p> <p>LLE must be reset by software (write 0). Writing 1 has no effect.</p>
LOE	9	rwh	<p>List Object Error</p> <p>0_B No List Object Error since last (most recent) flag reset. 1_B A List Object Error has been detected during message acceptance filtering. A message object with wrong LIST index entry in the Message Object Status Register has been detected.</p> <p>LOE must be reset by software (write 0). Writing 1 has no effect.</p>
0	10	rh	Reserved

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
0	[31:11]	r	Reserved Read as 0; should be written with 0.

Encoding of the LEC Bit field

Table 18-9 Encoding of the LEC Bit field

LEC Value	Signification
000 _B	No Error: No error was detected for the last (most recent) message on the CAN bus.
001 _B	Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
010 _B	Form Error: A fixed format part of a received frame has the wrong format.
011 _B	Ack Error: The transmitted message was not acknowledged by another node.
100 _B	Bit1 Error: During a message transmission, the CAN node tried to send a recessive level (1) outside the arbitration field and the acknowledge slot, but the monitored bus value was dominant.
101 _B	Bit0 Error: Two different conditions are signaled by this code: <ol style="list-style-type: none"> 1. During transmission of a message (or acknowledge bit, active-error flag, overload flag), the CAN node tried to send a dominant level (0), but the monitored bus value was recessive. 2. During bus-off recovery, this code is set each time a sequence of 11 recessive bits has been monitored. The CPU may use this code as indication that the bus is not continuously disturbed.
110 _B	CRC Error: The CRC checksum of the received message was incorrect.
111 _B	CPU write to LEC: Whenever the CPU writes the value 111 to LEC, it takes the value 111. Whenever the CPU writes another value to LEC, the written LEC value is ignored.

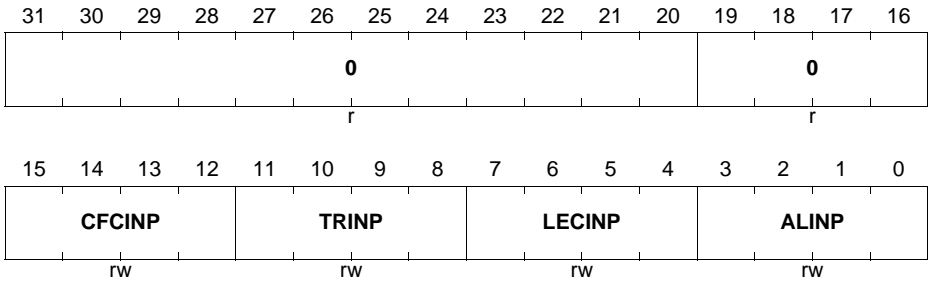
Controller Area Network Controller (MultiCAN+)

The four interrupt pointers in the Node Interrupt Pointer Register NIPRx select one out of the sixteen interrupt outputs individually for each type of CAN node interrupt. See also [Page 18-23](#) for more CAN node interrupt details.

CAN_NIPRx (x = 0-1)

Node x Interrupt Pointer Register (208_H+x*100_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ALINP	[3:0]	rw	<p>Alert Interrupt Node Pointer</p> <p>ALINP selects the interrupt output line INT_Om (m = 0-7) for an alert interrupt of CAN Node x.</p> <p>0000_B Interrupt output line INT_O0 is selected.</p> <p>0001_B Interrupt output line INT_O1 is selected.</p> <p>..._B ...</p> <p>1110_B Interrupt output line INT_O14 is selected.</p> <p>1111_B Interrupt output line INT_O15 is selected.</p>
LECINP	[7:4]	rw	<p>Last Error Code Interrupt Node Pointer</p> <p>LECINP selects the interrupt output line INT_Om (m = 0-7) for an LEC interrupt of CAN Node x.</p> <p>0000_B Interrupt output line INT_O0 is selected.</p> <p>0001_B Interrupt output line INT_O1 is selected.</p> <p>..._B ...</p> <p>1110_B Interrupt output line INT_O14 is selected.</p> <p>1111_B Interrupt output line INT_O15 is selected.</p>

Controller Area Network Controller (MultiCAN+)

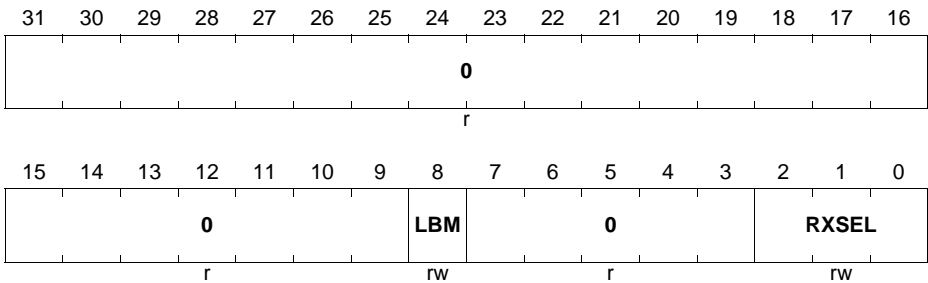
Field	Bits	Type	Description
TRINP	[11:8]	rw	Transfer OK Interrupt Node Pointer TRINP selects the interrupt output line INT_Om (m = 0-7) for a transfer OK interrupt of CAN Node x. 0000 _B Interrupt output line INT_O0 is selected. 0001 _B Interrupt output line INT_O1 is selected. ... _B ... 1110 _B Interrupt output line INT_O14 is selected. 1111 _B Interrupt output line INT_O15 is selected.
CFCINP	[15:12]	rw	Frame Counter Interrupt Node Pointer CFCINP selects the interrupt output line INT_Om (m = 0-7) for a frame counter overflow interrupt of CAN Node x. 0000 _B Interrupt output line INT_O0 is selected. 0001 _B Interrupt output line INT_O1 is selected. ... _B ... 1110 _B Interrupt output line INT_O14 is selected. 1111 _B Interrupt output line INT_O15 is selected.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Controller Area Network Controller (MultiCAN+)

The Node Port Control Register NPCRx configures the CAN bus transmit/receive ports. NPCRx can be written only if bit NCRx.CCE is set.

CAN_NPCRx (x = 0-1)

Node x Port Control Register (20C_H+x*100_H) **Reset Value: 0000 0000_H**



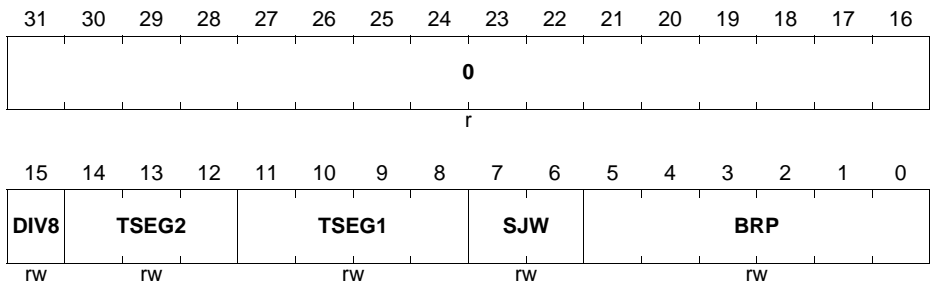
Field	Bits	Type	Description
RXSEL	[2:0]	rw	<p>Receive Select</p> <p>RXSEL selects one out of 8 possible receive inputs. The CAN receive signal is performed only through the selected input.</p> <p><i>Note: In XMC4300, only specific combinations of RXSEL are available (see also “Node Receive Input Selection” on Page 18-119 for description and the page before for RXSEL selections).</i></p>
LBM	8	rw	<p>Loop-Back Mode</p> <p>0_B Loop-Back Mode is disabled. 1_B Loop-Back Mode is enabled. This node is connected to an internal (virtual) loop-back CAN bus. All CAN nodes which are in Loop-Back Mode are connected to this virtual CAN bus so that they can communicate with each other internally. The external transmit line is forced recessive in Loop-Back Mode.</p>
0	[7:3], [31:9]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

Controller Area Network Controller (MultiCAN+)

The Node Bit Timing Register NBTRx contains all parameters to set up the bit timing for the CAN transfer. NBTRx can be written only if bit NCRx.CCE is set.

CAN_NBTRx (x = 0-1)

Node x Bit Timing Register (210_H+x*100_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
BRP	[5:0]	rw	Baud Rate Prescaler The duration of one time quantum is given by (BRP + 1) clock cycles if DIV8 = 0. The duration of one time quantum is given by 8 × (BRP + 1) clock cycles if DIV8 = 1.
SJW	[7:6]	rw	(Re) Synchronization Jump Width (SJW + 1) time quanta are allowed for re-synchronization.
TSEG1	[11:8]	rw	Time Segment Before Sample Point (TSEG1 + 1) time quanta is the user-defined nominal time between the end of the synchronization segment and the sample point. It includes the propagation segment, which takes into account signal propagation delays. The time segment may be lengthened due to re-synchronization. Valid values for TSEG1 are 2 to 15.
TSEG2	[14:12]	rw	Time Segment After Sample Point (TSEG2 + 1) time quanta is the user-defined nominal time between the sample point and the start of the next synchronization segment. It may be shortened due to re-synchronization. Valid values for TSEG2 are 1 to 7.

Controller Area Network Controller (MultiCAN+)

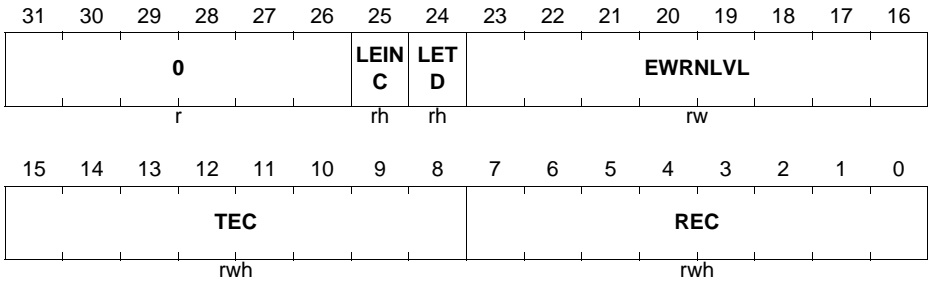
Field	Bits	Type	Description
DIV8	15	rw	Divide Prescaler Clock by 8 0_B A time quantum lasts (BRP+1) clock cycles. 1_B A time quantum lasts $8 \times$ (BRP+1) clock cycles.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

Controller Area Network Controller (MultiCAN+)

The Node Error Counter Register NECNTx contains the CAN receive and transmit error counter as well as some additional bits to ease error analysis. NECNTx can be written only if bit NCRx.CCE is set.

CAN_NECNTx (x = 0-1)

Node x Error Counter Register (214_H+x*100_H) **Reset Value: 0060 0000_H**



Field	Bits	Type	Description
REC	[7:0]	rwh	Receive Error Counter Bit field REC contains the value of the receive error counter of CAN node x.
TEC	[15:8]	rwh	Transmit Error Counter Bit field TEC contains the value of the transmit error counter of CAN node x.
EWRNLVL	[23:16]	rw	Error Warning Level Bit field EWRNLVL determines the threshold value (warning level, default 96) to be reached in order to set the corresponding error warning bit EWRN.
LETD	24	rh	Last Error Transfer Direction 0 _B The last error occurred while the CAN node x was receiver (REC has been incremented). 1 _B The last error occurred while the CAN node x was transmitter (TEC has been incremented).
LEINC	25	rh	Last Error Increment 0 _B The last error led to an error counter increment of 1. 1 _B The last error led to an error counter increment of 8.

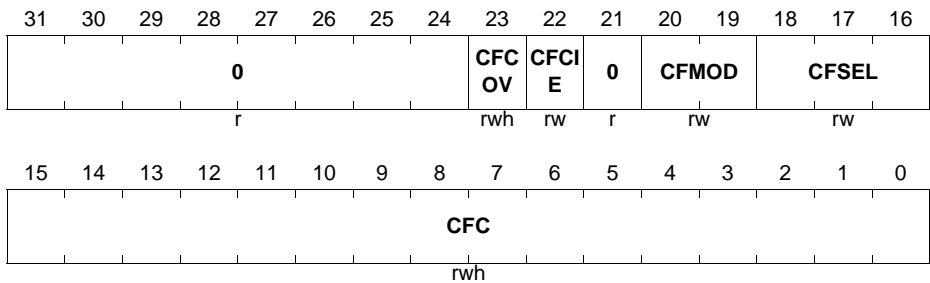
Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
0	[31:26]	r	Reserved Read as 0; should be written with 0.

The Node Frame Counter Register NFCRx contains the actual value of the frame counter as well as control and status bits of the frame counter.

CAN_NFCRx (x = 0-1)

Node x Frame Counter Register (218_H+x*100_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CFC	[15:0]	rwh	<p>CAN Frame Counter</p> <p>In Frame Count Mode (CFMOD = 00_B), this bit field contains the frame count value.</p> <p>In Time Stamp Mode (CFMOD = 01_B), this bit field contains the captured bit time count value, captured with the start of a new frame.</p> <p>In all Bit Timing Analysis Modes¹⁾ (CFMOD = 10_B), CFC always displays the number of f_{CLC} clock cycles (measurement result) minus 1. Example: a CFC value of 34 in measurement mode CFSEL = 000_B means that 35 f_{CLC} clock cycles have been elapsed between the most recent two dominant edges on the receive input.</p> <p>In Error Count Mode (CFMOD = 11_B), this bit field contains the total amount of error frames received or error detected by the node.</p>

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
CFSEL	[18:16]	rw	<p>CAN Frame Count Selection</p> <p>This bit field selects the function of the frame counter for the chosen frame count mode.</p> <p>Frame Count Mode</p> <p>Bit 0 If Bit 0 of CFSEL is set, then CFC is incremented each time a foreign frame (i.e. a frame not matching to a message object) has been received on the CAN bus.</p> <p>Bit 1 If Bit 1 of CFSEL is set, then CFC is incremented each time a frame matching to a message object has been received on the CAN bus.</p> <p>Bit 2 If Bit 2 of CFSEL is set, then CFC is incremented each time a frame has been transmitted successfully by the node.</p> <p>Time Stamp Mode</p> <p>The frame counter is incremented (internally) at the beginning of a new bit time. The value is sampled during the SOF bit of a new frame. The sampled value is visible in the CFC field.</p> <p>Bit Timing Mode</p> <p>The available bit timing measurement modes are shown in Table 18-10. If CFCIE is set, then an interrupt on request node x (where x is the CAN node number) is generated with a CFC update.</p> <p>Error Count Mode</p> <p>The frame counter is incremented when an error frame is received or an error is detected by the node. (001_B to 110_B) (see Table 18-9 for Encoding of the LEC Bit field). The configuration is don't care, in this mode.</p>

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
CFMOD	[20:19]	rw	<p>CAN Frame Counter Mode</p> <p>This bit field determines the operation mode of the frame counter.</p> <p>00_B Frame Count Mode: The frame counter is incremented upon the reception and transmission of frames.</p> <p>01_B Time Stamp Mode: The frame counter is used to count bit times.</p> <p>10_B Bit Timing Mode: The frame counter is used for analysis of the bit timing.</p> <p>11_B Error Count Mode: The frame counter is used for counting when an error frame is received or an error is detected by the node.</p>
CFCIE	22	rw	<p>CAN Frame Count Interrupt Enable</p> <p>CFCIE enables the CAN frame counter overflow interrupt of CAN node x.</p> <p>0_B CAN frame counter overflow interrupt is disabled.</p> <p>1_B CAN frame counter overflow interrupt is enabled.</p> <p>Bit field NIPRx.CFCINP selects the interrupt output line that is activated at this type of interrupt.</p>
CFCOV	23	rwh	<p>CAN Frame Counter Overflow Flag</p> <p>Flag CFCOV is set upon a frame counter overflow (transition from FFFF_H to 0000_H). In bit timing analysis mode, CFCOV is set upon an update of CFC. An interrupt request is generated if CFCIE = 1.</p> <p>0_B No overflow has occurred since last flag reset.</p> <p>1_B An overflow has occurred since last flag reset.</p> <p>CFCOV must be reset by software.</p>
0	21, [31:24]	r	<p>Reserved</p> <p>Read as 0; should be written with 0.</p>

1) The value of NFCRx.CFC is valid one module cycle later when NFCRx.CFCOV is set.

Bit Timing Analysis Modes

Table 18-10 Bit Timing Analysis Modes (CFMOD = 10)

CFSEL	Measurement
000 _B	Whenever a dominant edge (transition from 1 to 0) is monitored on the receive input, the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in CFC.
001 _B	Whenever a recessive edge (transition from 0 to 1) is monitored on the receive input the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in CFC.
010 _B	Whenever a dominant edge is received as a result of a transmitted dominant edge, the time (clock cycles) between both edges is stored in CFC.
011 _B	Whenever a recessive edge is received as a result of a transmitted recessive edge, the time (clock cycles) between both edges is stored in CFC.
100 _B	Whenever a dominant edge that qualifies for synchronization is monitored on the receive input, the time (measured in clock cycles) between this edge and the most recent sample point is stored in CFC.
101 _B	With each sample point, the time (measured in clock cycles) between the start of the new bit time and the start of the previous bit time is stored in CFC[11:0]. Additional information is written to CFC[15:12] at each sample point: CFC[15]: Transmit value of actual bit time CFC[14]: Receive sample value of actual bit time CFC[13:12]: CAN bus information (see Table 18-11)
110 _B	Reserved, do not use this combination.
111 _B	Reserved, do not use this combination.

Table 18-11 CAN Bus State Information

CFC[13:12]	CAN Bus State
00 _B	<p>NoBit The CAN bus is idle, performs bit (de-) stuffing or is in one of the following frame segments: SOF, SRR, CRC, delimiters, first 6 EOF bits, IFS.</p>
01 _B	<p>NewBit This code represents the first bit of a new frame segment. The current bit is the first bit in one of the following frame segments: Bit 10 (MSB) of standard ID (transmit only), RTR, reserved bits, IDE, DLC(MSB), bit 7 (MSB) in each data byte and the first bit of the ID extension.</p>
10 _B	<p>Bit This code represents a bit inside a frame segment with a length of more than one bit (not the first bit of those frame segments that is indicated by NewBit). The current bit is processed within one of the following frame segments: ID bits (except first bit of standard ID for transmission and first bit of ID extension), DLC (3 LSB) and bits 6-0 in each data byte.</p>
11 _B	<p>Done The current bit is in one of the following frame segments: Acknowledge slot, last bit of EOF, active/passive-error frame, overload frame. Two or more directly consecutive Done codes signal an Error Frame.</p>

Controller Area Network Controller (MultiCAN+)

18.5.3 Message Object Registers

The Message Object Control Register MOCTR_n and the Message Object Status Register MOSTAT_n are located at the same address offset within a message object address block (offset address 1C_H). The MOCTR_n is a write-only register that makes it possible to set/reset CAN transfer related control bits through software. Therefore the reset value is written as 0_H, even though the read part of the register has a different reset value.

CAN_MOCTR_z (z = 0-62)

Message Object z Control Register(101C_H+z*20_H)

Reset Value: 00000000_H

CAN_MOCTR₆₃

Message Object 63 Control Register (17FC_H)

Reset Value: 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				SET DIR	SET TXE N1	SET TXE N0	SET TXR Q	SET RXE N	SET RTS EL	SET MSG VAL	SET MSG LST	SET NEW DAT	SET RXU PD	SET TXP ND	SET RXP ND
w				w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				RES DIR	RES TXE N1	RES TXE N0	RES TXR Q	RES RXE N	RES RTS EL	RES MSG VAL	RES MSG LST	RES NEW DAT	RES RXU PD	RES TXP ND	RES RXP ND
w				w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RESRXPND, SETRXPND	0, 16	w	Reset/Set Receive Pending These bits control the set/reset condition for RXPND (see Table 18-12).
RESTXPND, SETTXPND	1, 17	w	Reset/Set Transmit Pending These bits control the set/reset condition for TXPND (see Table 18-12).
RESRXUPD, SETRXUPD	2, 18	w	Reset/Set Receive Updating These bits control the set/reset condition for RXUPD (see Table 18-12).
RESNEWDAT, SETNEWDAT	3, 19	w	Reset/Set New Data These bits control the set/reset condition for NEWDAT (see Table 18-12).

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
RESMSGST, SETMSGST	4, 20	w	Reset/Set Message Lost These bits control the set/reset condition for MSGST (see Table 18-12).
RESMSGVAL, SETMSGVAL	5, 21	w	Reset/Set Message Valid These bits control the set/reset condition for MSGVAL (see Table 18-12).
RESRTSEL, SETRTSEL	6, 22	w	Reset/Set Receive/Transmit Selected These bits control the set/reset condition for RTSEL (see Table 18-12).
RESRXEN, SETRXEN	7, 23	w	Reset/Set Receive Enable These bits control the set/reset condition for RXEN (see Table 18-12).
RESTXRQ, SETTXRQ	8, 24	w	Reset/Set Transmit Request These bits control the set/reset condition for TXRQ (see Table 18-12).
RESTXEN0, SETTXEN0	9, 25	w	Reset/Set Transmit Enable 0 These bits control the set/reset condition for TXEN0 (see Table 18-12).
RESTXEN1, SETTXEN1	10, 26	w	Reset/Set Transmit Enable 1 These bits control the set/reset condition for TXEN1 (see Table 18-12).
RESDIR, SETDIR	11, 27	w	Reset/Set Message Direction These bits control the set/reset condition for DIR (see Table 18-12).
0	[15:12], [31:28]	w	Reserved Should be written with 0.

Table 18-12 Reset/Set Conditions for Bits in Register MOCTRn

RESy Bit ¹⁾	SETy Bit	Action on Write
Write 0	Write 0	Leave element unchanged
	No write	
No write	Write 0	
Write 1	Write 1	

Controller Area Network Controller (MultiCAN+)

Table 18-12 Reset/Set Conditions for Bits in Register MOCTRn (cont'd)

RESy Bit¹⁾	SETy Bit	Action on Write
Write 1	Write 0	Reset element
	No write	
Write 0	Write 1	Set element
No write		

1) The parameter “y” stands for the second part of the bit name (“RXPND”, “TXPND”, ... up to “DIR”).

Controller Area Network Controller (MultiCAN+)

The MOSTATn is a read-only register that indicates message object list status information such as the number of the current message object predecessor and successor message object, as well as the list number to which the message object is assigned.

CAN_MOSTAT0

Message Object 0 Status Register (101C_H) **Reset Value: 0100 0000_H**

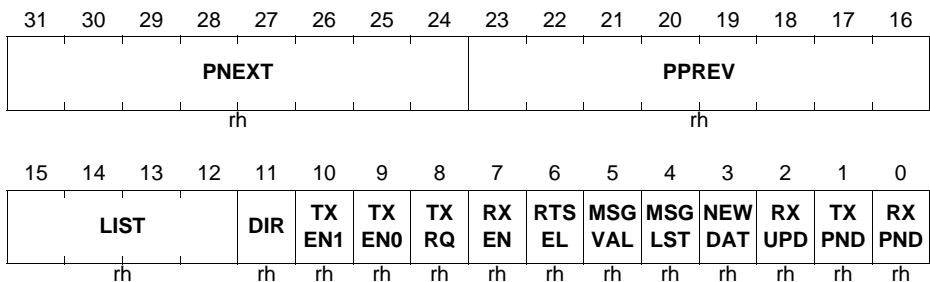
CAN_MOSTATn (n = 1-62)

Message Object n Status Register(101C_H+n*20_H)

Reset Value: ((n+1)*01000000_H)+((n-1)*00010000_H)

CAN_MOSTAT63

Message Object 63 Status Register (17FC_H) **Reset Value: 3F3E0000_H**



Field	Bits	Type	Description
RXPND	0	rh	<p>Receive Pending</p> <p>0_B No CAN message has been received. 1_B A CAN message has been received by the message object n, either directly or via gateway copy action. RXPND is set by hardware and must be reset by software.</p>
TXPND	1	rh	<p>Transmit Pending</p> <p>0_B No CAN message has been transmitted. 1_B A CAN message from message object n has been transmitted successfully over the CAN bus. TXPND is set by hardware and must be reset by software.</p>

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
RXUPD	2	rh	<p>Receive Updating</p> <p>0_B No receive update ongoing.</p> <p>1_B Message identifier, DLC, and data of the message object are currently updated.</p>
NEWDAT	3	rh	<p>New Data</p> <p>0_B No update of the message object n since last flag reset.</p> <p>1_B Message object n has been updated.</p> <p>NEWDAT is set by hardware after a received CAN frame has been stored in message object n.</p> <p>NEWDAT is cleared by hardware when a CAN transmission of message object n has been started.</p> <p>NEWDAT should be set by software after the new transmit data has been stored in message object n to prevent the automatic reset of TXRQ at the end of an ongoing transmission.</p>
MSGLST	4	rh	<p>Message Lost</p> <p>0_B No CAN message is lost.</p> <p>1_B A CAN message is lost because NEWDAT has become set again when it has already been set.</p>
MSGVAL	5	rh	<p>Message Valid</p> <p>0_B Message object n is not valid.</p> <p>1_B Message object n is valid.</p> <p>Only a valid message object takes part in CAN transfers.</p>

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
RTSEL	6	rh	<p>Receive/Transmit Selected</p> <p>0_B Message object n is not selected for receive or transmit operation.</p> <p>1_B Message object n is selected for receive or transmit operation.</p> <p>Frame Reception: RTSEL is set by hardware when message object n has been identified for storage of a CAN frame that is currently received. Before a received frame becomes finally stored in message object n, a check is performed to determine if RTSEL is set. Thus the CPU can suppress a scheduled frame delivery to this message object n by clearing RTSEL by software.</p> <p>Frame Transmission: RTSEL is set by hardware when message object n has been identified to be transmitted next. A check is performed to determine if RTSEL is still set before message object n is actually set up for transmission and bit NEWDAT is cleared. It is also checked that RTSEL is still set before its message object n is verified due to the successful transmission of a frame. RTSEL needs to be checked only when the context of message object n changes, and a conflict with an ongoing frame transfer shall be avoided. In all other cases, RTSEL can be ignored. RTSEL has no impact on message acceptance filtering. RTSEL is not cleared by hardware.</p>
RXEN	7	rh	<p>Receive Enable</p> <p>0_B Message object n is not enabled for frame reception.</p> <p>1_B Message object n is enabled for frame reception.</p> <p>RXEN is evaluated for receive acceptance filtering only.</p>

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
TXRQ	8	rh	<p>Transmit Request</p> <p>0_B No transmission of message object n is requested.</p> <p>1_B Transmission of message object n on the CAN bus is requested.</p> <p>The transmit request becomes valid only if TXRQ, TXEN0, TXEN1 and MSGVAL are set. TXRQ is set by hardware if a matching Remote Frame has been received correctly. TXRQ is reset by hardware if message object n has been transmitted successfully and NEWDAT is not set again by software.</p>
TXEN0	9	rh	<p>Transmit Enable 0</p> <p>0_B Message object n is not enabled for frame transmission.</p> <p>1_B Message object n is enabled for frame transmission.</p> <p>Message object n can be transmitted only if both bits, TXEN0 and TXEN1, are set.</p> <p>The user may clear TXEN0 in order to inhibit the transmission of a message that is currently updated, or to disable automatic response of Remote Frames.</p>
TXEN1	10	rh	<p>Transmit Enable 1</p> <p>0_B Message object n is not enabled for frame transmission.</p> <p>1_B Message object n is enabled for frame transmission.</p> <p>Message object n can be transmitted only if both bits, TXEN0 and TXEN1, are set.</p> <p>TXEN1 is used by the MultiCAN+ module for selecting the active message object in the Transmit FIFOs.</p>

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
DIR	11	rh	Message Direction 0_B Receive Object selected: With TXRQ = 1, a Remote Frame with the identifier of message object n is scheduled for transmission. On reception of a Data Frame with matching identifier, the message is stored in message object n. 1_B Transmit Object selected: If TXRQ = 1, message object n is scheduled for transmission of a Data Frame. On reception of a Remote Frame with matching identifier, bit TXRQ is set.
LIST	[15:12]	rh	List Allocation LIST indicates the number of the message list to which message object n is allocated. LIST is updated by hardware when the list allocation of the object is modified by a panel command.
PPREV	[23:16]	rh	Pointer to Previous Message Object PPREV holds the message object number of the previous message object in a message list structure.
PNEXT	[31:24]	rh	Pointer to Next Message Object PNEXT holds the message object number of the next message object in a message list structure.

Table 18-13 MOSTATn Reset Values

Message Object	PNEXT	PPREV	Reset Value
0	1	0	0100 0000 _H
1	2	0	0200 0000 _H
2	3	1	0301 0000 _H
3	4	2	0402 0000 _H
...
<u>63</u>	<u>63</u>	<u>62</u>	<u>3F3E</u> 0000 _H

The Message Object Interrupt Pointer Register MOIPRn holds the message interrupt pointers, the message pending number, and the frame counter value of message object n.

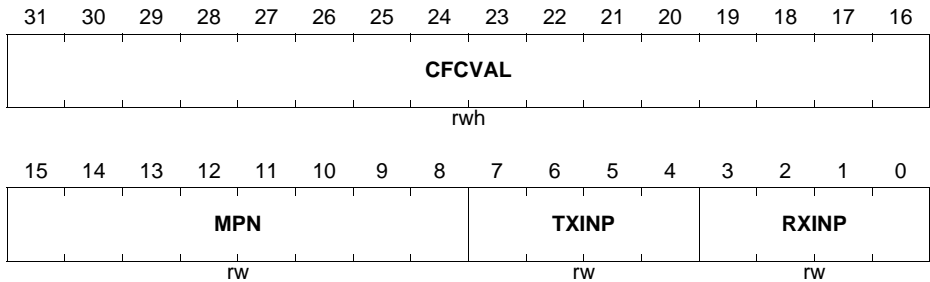
Controller Area Network Controller (MultiCAN+)

CAN_MOIPRn (n = 0-63)

Message Object n Interrupt Pointer Register

(1008_H+n*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
RXINP	[3:0]	rw	<p>Receive Interrupt Node Pointer</p> <p>RXINP selects the interrupt output line INT_O_m (m = 0-7) for a receive interrupt event of message object n. RXINP can also be taken for message pending bit selection (see Page 18-37).</p> <p>0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. ..._B ... 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.</p>
TXINP	[7:4]	rw	<p>Transmit Interrupt Node Pointer</p> <p>TXINP selects the interrupt output line INT_O_m (m = 0-7) for a transmit interrupt event of message object n. TXINP can also be taken for message pending bit selection (see Page 18-37).</p> <p>0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. ..._B ... 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.</p>
MPN	[15:8]	rw	<p>Message Pending Number</p> <p>This bit field selects the bit position of the bit in the Message Pending Register that is set upon a message object n receive/transmit interrupt.</p>

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
CFCVAL	[31:16]	rwh	CAN Frame Counter Value When a message is stored in message object n or message object n has been successfully transmitted, the CAN frame counter value NFCRx.CFC is then copied to CFCVAL.

Controller Area Network Controller (MultiCAN+)

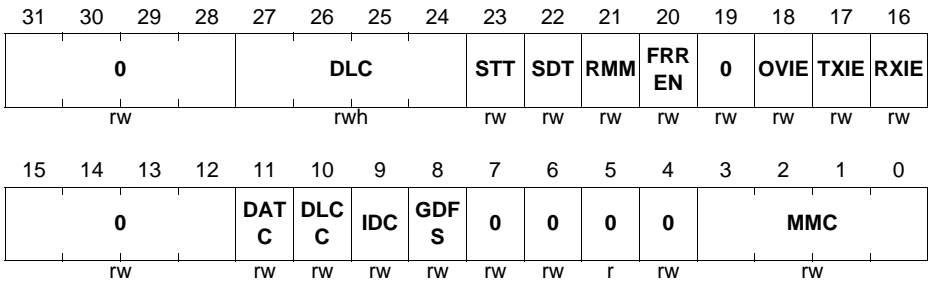
The Message Object Function Control Register MOFCRn contains bits that select and configure the function of the message object. It also holds the CAN data length code.

CAN_MOFCRn (n = 0-63)

Message Object n Function Control Register

(1000_H+n*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
MMC	[3:0]	rw	Message Mode Control MMC controls the message mode of message object n. 0000 _B Standard Message Object 0001 _B Receive FIFO Base Object 0010 _B Transmit FIFO Base Object 0011 _B Transmit FIFO Slave Object 0100 _B Gateway Source Object 0101 _B Do not use 0110 _B Do not use ... _B ... 1111 _B Do not use
0	4	rw	Reserved Shall be written with 0 _H .
0	5	rw	Reserved Shall be written with 0 _H .
0	6	rw	Reserved Shall be written with 0 _H . Setting a different value, will disable transmissions.

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
GDFS	8	rw	<p>Gateway Data Frame Send</p> <p>0_B TXRQ is unchanged in the destination object. 1_B TXRQ is set in the gateway destination object after the internal transfer from the gateway source to the gateway destination object. Applicable only to a gateway source object; ignored in other nodes.</p>
IDC	9	rw	<p>Identifier Copy</p> <p>0_B The identifier of the gateway source object is not copied. 1_B The identifier of the gateway source object (after storing the received frame in the source) is copied to the gateway destination object. Applicable only to a gateway source object; ignored in other nodes.</p>
DLCC	10	rw	<p>Data Length Code Copy</p> <p>0_B Data length code is not copied. 1_B Data length code of the gateway source object (after storing the received frame in the source) is copied to the gateway destination object. Applicable only to a gateway source object; ignored in other nodes.</p>
DATC	11	rw	<p>Data Copy</p> <p>0_B Data fields are not copied. 1_B Data fields in registers MODATALn and MODATAHn of the gateway source object (after storing the received frame in the source) are copied to the gateway destination. Applicable only to a gateway source object; ignored in other nodes.</p>
RXIE	16	rw	<p>Receive Interrupt Enable</p> <p>RXIE enables the message receive interrupt of message object n. This interrupt is generated after reception of a CAN message (independent of whether the CAN message is received directly or indirectly via a gateway action). 0_B Message receive interrupt is disabled. 1_B Message receive interrupt is enabled. Bit field MOIPRn.RXINP selects the interrupt output line which becomes activated at this type of interrupt.</p>

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
TXIE	17	rw	<p>Transmit Interrupt Enable</p> <p>TXIE enables the message transmit interrupt of message object n. This interrupt is generated after the transmission of a CAN message.</p> <p>0_B Message transmit interrupt is disabled. 1_B Message transmit interrupt is enabled. Bit field MOIPRn.TXINP selects the interrupt output line which becomes activated at this type of interrupt.</p>
OVIE	18	rw	<p>Overflow Interrupt Enable</p> <p>OVIE enables the FIFO full interrupt of message object n. This interrupt is generated when the pointer to the current message object (CUR) reaches the value of SEL in the FIFO/Gateway Pointer Register.</p> <p>0_B FIFO full interrupt is disabled. 1_B FIFO full interrupt is enabled. If message object n is a Receive FIFO base object, bit field MOIPRn.TXINP selects the interrupt output line which becomes activated at this type of interrupt. If message object n is a Transmit FIFO base object, bit field MOIPRn.RXINP selects the interrupt output line which becomes activated at this type of interrupt. For all other message object modes, bit OVIE has no effect.</p>
FRREN	20	rw	<p>Foreign Remote Request Enable</p> <p>Specifies whether the TXRQ bit is set in message object n or in a foreign message object referenced by the pointer CUR.</p> <p>0_B TXRQ of message object n is set on reception of a matching Remote Frame. 1_B TXRQ of the message object referenced by the pointer CUR is set on reception of a matching Remote Frame.</p>

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
RMM	21	rw	<p>Transmit Object Remote Monitoring</p> <p>0_B Remote monitoring is disabled: Identifier, IDE bit, and DLC of message object n remain unchanged upon the reception of a matching Remote Frame.</p> <p>1_B Remote monitoring is enabled: Identifier, IDE bit, and DLC of a matching Remote Frame are copied to transmit object n in order to monitor incoming Remote Frames. Bit RMM applies only to transmit objects and has no effect on receive objects.</p>
SDT	22	rw	<p>Single Data Transfer</p> <p>If SDT = 1 and message object n is not a FIFO base object, then MSGVAL is reset when this object has taken part in a successful data transfer (receive or transmit).</p> <p>If SDT = 1 and message object n is a FIFO base object, then MSGVAL is reset when the pointer to the current object CUR reaches the value of SEL in the FIFO/Gateway Pointer Register. With SDT = 0, bit MSGVAL is not affected.</p>
STT	23	rw	<p>Single Transmit Trial</p> <p>If this bit is set, then TXRQ is cleared on transmission start of message object n. Thus, no transmission retry is performed in case of transmission failure.</p>
DLC	[27:24]	rwh	<p>Data Length Code</p> <p>Bit field determines the number of data bytes for message object n. A value of DLC > 8 results in a data length of 8 data bytes. If a frame with DLC > 8 is received, the received value is stored in the message object.</p>
0	5, 7, [15:12], 19, [31:28]	rw	<p>Reserved</p> <p>Read as 0 after reset; value last written is read back; should be written with 0.</p>

Controller Area Network Controller (MultiCAN+)

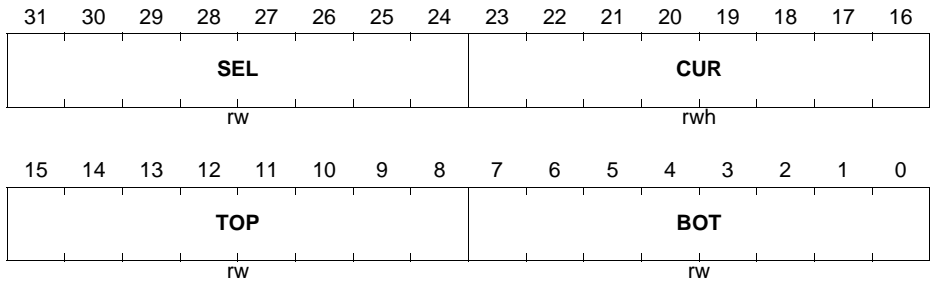
The Message Object FIFO/Gateway Pointer register MOFGPRn contains a set of message object link pointers that are used for FIFO and gateway operations.

CAN_MOFGPRn (n = 0-63)

Message Object n FIFO/Gateway Pointer Register

(1004_H+n*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
BOT	[7:0]	rw	Bottom Pointer Bit field BOT points to the first element in a FIFO structure.
TOP	[15:8]	rw	Top Pointer Bit field TOP points to the last element in a FIFO structure.
CUR	[23:16]	rwh	Current Object Pointer Bit field CUR points to the actual target object within a FIFO/Gateway structure. After a FIFO/gateway operation CUR is updated with the message number of the next message object in the list structure (given by PNEXT of the Message Object Status Register) until it reaches the FIFO top element (given by TOP) when it is reset to the bottom element (given by BOT).
SEL	[31:24]	rw	Object Select Pointer Bit field SEL is the second (software) pointer to complement the hardware pointer CUR in the FIFO structure. SEL is used for monitoring purposes (FIFO interrupt generation).

Controller Area Network Controller (MultiCAN+)

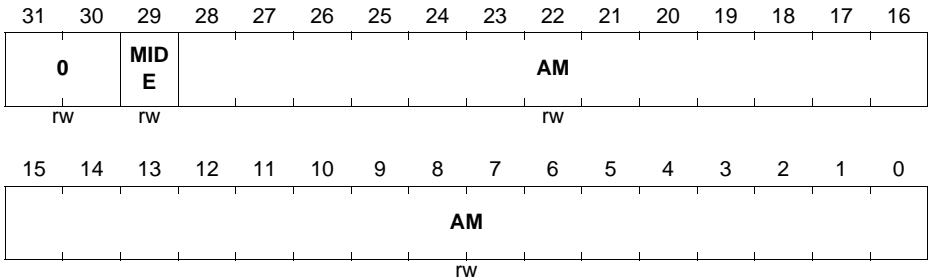
Message Object n Acceptance Mask Register MOAMRn contains the mask bits for the acceptance filtering of the message object n.

CAN_MOAMRn (n = 0-63)

Message Object n Acceptance Mask Register

($100C_H + n * 20_H$)

Reset Value: 3FFF FFFF_H



Field	Bits	Type	Description
AM	[28:0]	rw	Acceptance Mask for Message Identifier Bit field AM is the 29-bit mask for filtering incoming messages with standard identifiers (AM[28:18]) or extended identifiers (AM[28:0]). For standard identifiers, bits AM[17:0] are "don't care".
MIDE	29	rw	Acceptance Mask Bit for Message IDE Bit 0 _B Message object n accepts the reception of both, standard and extended frames. 1 _B Message object n receives frames only with matching IDE bit.
0	[31:30]	rw	Reserved Read as 0 after reset; value last written is read back; should be written with 0.

Controller Area Network Controller (MultiCAN+)

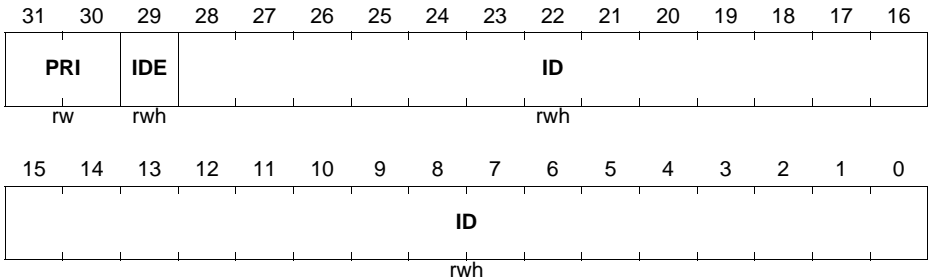
Message Object n Arbitration Register MOARn contains the CAN identifier of the message object.

CAN_MOARn (n = 0-63)

Message Object n Arbitration Register

(1018_H+n*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
ID	[28:0]	rwh	CAN Identifier of Message Object n Identifier of a standard message (ID[28:18]) or an extended message (ID[28:0]). For standard identifiers, bits ID[17:0] are “don’t care”.
IDE	29	rwh	Identifier Extension Bit of Message Object n 0 _B Message object n handles standard frames with 11-bit identifier. 1 _B Message object n handles extended frames with 29-bit identifier.

Controller Area Network Controller (MultiCAN+)

Field	Bits	Type	Description
PRI	[31:30]	rw	<p>Priority Class</p> <p>PRI assigns one of the four priority classes 0, 1, 2, 3 to message object n. A lower PRI number defines a higher priority. Message objects with lower PRI value always win acceptance filtering for frame reception and transmission over message objects with higher PRI value. Acceptance filtering based on identifier/mask and list position is performed only between message objects of the same priority class. PRI also determines the acceptance filtering method for transmission:</p> <p>00_B Reserved.</p> <p>01_B Transmit acceptance filtering is based on the list order. This means that message object n is considered for transmission only if there is no other message object with valid transmit request (MSGVAL & TXEN0 & TXEN1 = 1) somewhere before this object in the list.</p> <p>10_B Transmit acceptance filtering is based on the CAN identifier. This means, message object n is considered for transmission only if there is no other message object with higher priority identifier + IDE + DIR (with respect to CAN arbitration rules) somewhere in the list (see Table 18-14).</p> <p>11_B Transmit acceptance filtering is based on the list order (as PRI = 01_B).</p>

Controller Area Network Controller (MultiCAN+)

Transmit Priority of Msg. Objects based on CAN Arbitration Rules

Table 18-14 Transmit Priority of Msg. Objects Based on CAN Arbitration Rules

Settings of Arbitrarily Chosen Message Objects A and B, (A has higher transmit priority than B)	Comment
A.MOAR[28:18] < B.MOAR[28:18] (11-bit standard identifier of A less than 11-bit standard identifier of B)	Messages with lower standard identifier have higher priority than messages with higher standard identifier. MOAR[28] is the most significant bit (MSB) of the standard identifier. MOAR[18] is the least significant bit of the standard identifier.
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = 0 (send Standard Frame) B.MOAR.IDE = 1 (send Extended Frame)	Standard Frames have higher transmit priority than Extended Frames with equal standard identifier.
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = B.MOAR.IDE = 0 A.MOSTAT.DIR = 1 (send Data Frame) B.MOSTAT.DIR = 0 (send Remote Fame)	Standard Data Frames have higher transmit priority than standard Remote Frames with equal identifier.
A.MOAR[28:0] = B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 A.MOSTAT.DIR = 1 (send Data Frame) B.MOSTAT.DIR = 0 (send Remote Frame)	Extended Data Frames have higher transmit priority than Extended Remote Frames with equal identifier.
A.MOAR[28:0] < B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 (29-bit identifier)	Extended Frames with lower identifier have higher transmit priority than Extended Frames with higher identifier. MOAR[28] is the most significant bit (MSB) of the overall identifier (standard identifier MOAR[28:18] and identifier extension MOAR[17:0]). MOAR[0] is the least significant bit (LSB) of the overall identifier.

Controller Area Network Controller (MultiCAN+)

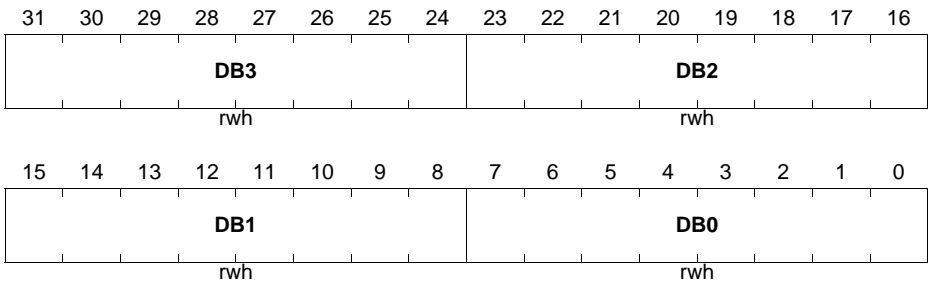
Message Object n Data Register Low MODATALn contains the lowest four data bytes of message object n. Unused data bytes are set to zero upon reception and ignored for transmission.

CAN_MODATALn (n = 0-63)

Message Object n Data Register Low

(1010_H+n*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DB0	[7:0]	rwh	Data Byte 0 of Message Object n
DB1	[15:8]	rwh	Data Byte 1 of Message Object n
DB2	[23:16]	rwh	Data Byte 2 of Message Object n
DB3	[31:24]	rwh	Data Byte 3 of Message Object n

Controller Area Network Controller (MultiCAN+)

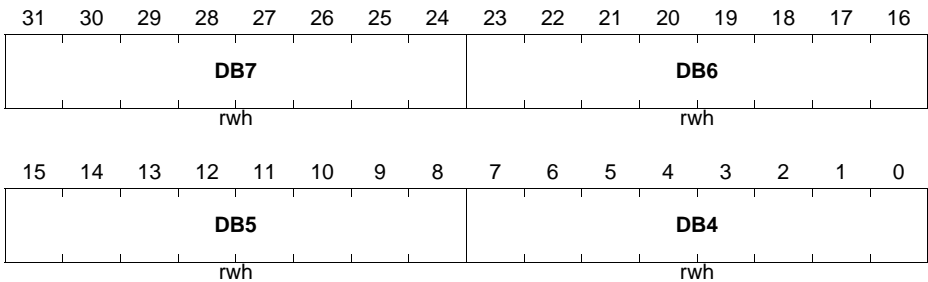
Message Object n Data Register High MODATAH contains the highest four data bytes of message object n. Unused data bytes are set to zero upon reception and ignored for transmission.

CAN_MODATAHn (n = 0-63)

Message Object n Data Register High

(1014_H+n*20_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
DB4	[7:0]	rwh	Data Byte 4 of Message Object n
DB5	[15:8]	rwh	Data Byte 5 of Message Object n
DB6	[23:16]	rwh	Data Byte 6 of Message Object n
DB7	[31:24]	rwh	Data Byte 7 of Message Object n

Controller Area Network Controller (MultiCAN+)

18.6 MultiCAN+ Module Implementation

This section describes CAN module interfaces with the clock control, port connections, interrupt control, and address decoding.

18.6.1 Interfaces of the MultiCAN+ Module

Figure 18-24 shows the XMC4300 specific implementation details and interconnections of the MultiCAN+ module. The I/O lines of the MultiCAN+ module (two I/O lines of each CAN node) are connected to the Ports listed in **Table 18-16**. The MultiCAN+ module is also supplied by clock control, interrupt control, and address decoding logic. MultiCAN+ interrupts can be directed to the DMA, CPU, CCU4 modules which are able to trigger DMA transfers and CCU4, CPU operations. Detailed tables see paragraph: **Connections with other modules**.

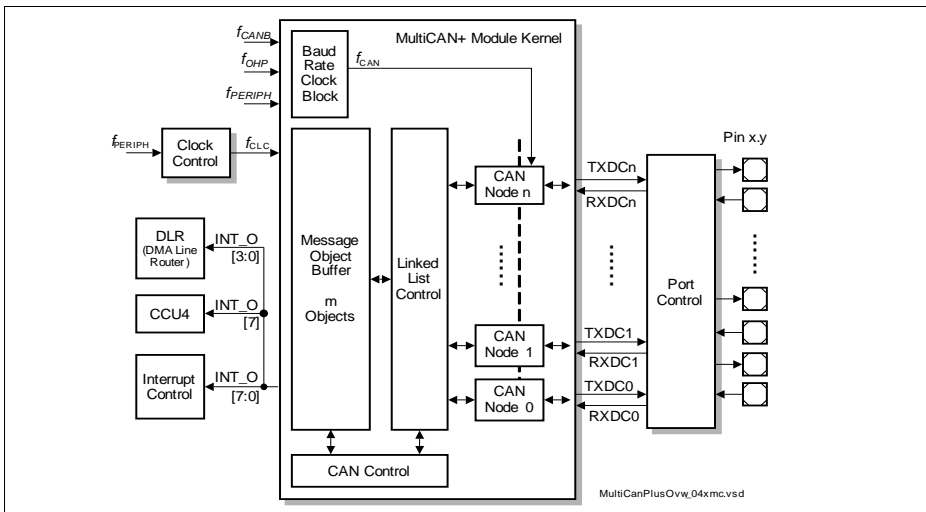


Figure 18-24 MultiCAN+ Module Implementation and Interconnections with $n := 1$ and $m := 64$ for XMC4300

Controller Area Network Controller (MultiCAN+)

18.6.2 MultiCAN+ Module External Registers

The registers listed in [Figure 18-25](#) are not included in the MultiCAN+ module kernel, some registers must be programmed for proper operation of the MultiCAN+ module.

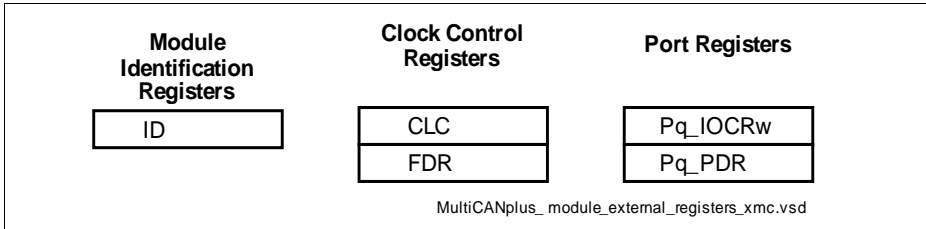


Figure 18-25 CAN Implementation-specific Special Function Registers

Table 18-15 MultiCAN+ Module External Registers

Short Name	Description	Offset Addr	Access Mode		Reset Class	Description see
			Read	Write		
Module Identification Registers						
ID	Module Identification Register	008 _H	U, PV	nBE	Application Reset	Page 18-62
Clock Control Registers						
CLC	Clock Control Register	000 _H	U, PV	PV,	Application Reset	Page 18-11 7
FDR	Fractional Divider Register	00C _H	U, PV	PV,	Application Reset	Page 18-11 8

18.6.3 Module Clock Generation

This chapter describes the way the module gets its clock.

18.6.3.1 Clock Selection

The bit timing machine and the rest of the MultiCAN+ module are separate frequency domains and can be driven by separate independent frequencies. The bit timing unit can be driven by the AHB bus clock or with the direct oscillator clock, and the rest of the chip is driven only by the AHB bus clock.

The purpose of supplying the bit timing unit with a direct oscillator clock is to avoid the clock jitter added by the PLL, necessary when the chip is driven by a low cost ceramic resonator instead of a high precision quartz crystal.

Selecting the clock source for the bit timing unit is done by programming the bit-field MCR.CLKSEL.

Enabling and disabling the clock of the module by using CLC.DISR affects always both frequency domains, so that when f_{CLC} is switched off, f_A is also switched off.

18.6.3.2 Fractional Divider

As shown in **Figure 18-26**, the clock signals for the MultiCAN+ module are generated and controlled by a clock control unit. This clock generation unit is responsible for the enable/disable control, the clock frequency adjustment, and the debug clock control. This unit includes two registers:

- CAN_CLC: generation of the module control clock f_{CLC}
- CAN_FDR: frequency control of the module timer f_{CAN}

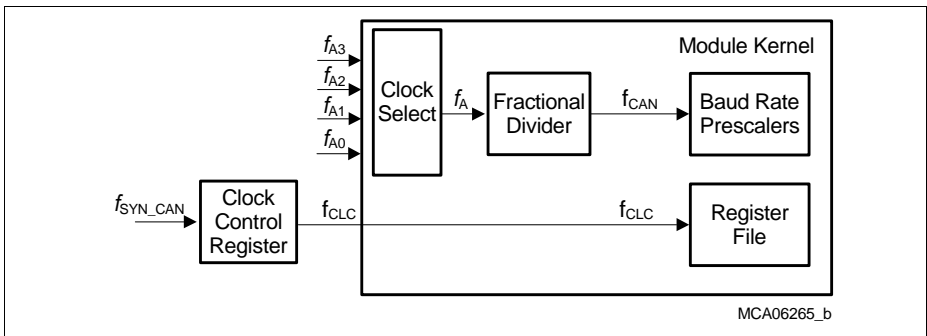


Figure 18-26 MultiCAN+ Module Clock Generation

The f_{SYN_CAN} is identical to f_{PERIPH} .

Controller Area Network Controller (MultiCAN+)

The module control clock f_{CLC} is used inside the MultiCAN+ module for control purposes such as clocking of control logic and register operations. The frequency of f_{CLC} is identical to the system clock frequency f_{PERIPH} . The clock control register CAN_CLC makes it possible to enable/disable f_{CLC} under certain conditions.

The module timer clock f_{CAN} is used inside the MultiCAN+ module as input clock for all timing relevant operations (e.g. bit timing). The settings in the CAN_FDR register determine the frequency of the module timer clock f_{CAN} according the following two formulas:

$$f_{CAN} = f_A \times \frac{1}{n} \quad \text{with } n = 1024 - \text{CAN_FDR.STEP} \quad (18.2)$$

$$f_{CAN} = f_A \times \frac{n}{1024} \quad \text{with } n = 0-1023 \quad (18.3)$$

Equation (18.2) applies to normal divider mode (CAN_FDR.DM = 01_B) of the fractional divider. **Equation (18.3)** applies to fractional divider mode (CAN_FDR.DM = 10_B).

Note: The CAN module is disabled after reset. In general, after reset, the module control clock f_{CLC} must be switched on (writing to register CAN_CLC) before the frequency of the module timer clock f_{CAN} is defined (writing to register CAN_FDR).

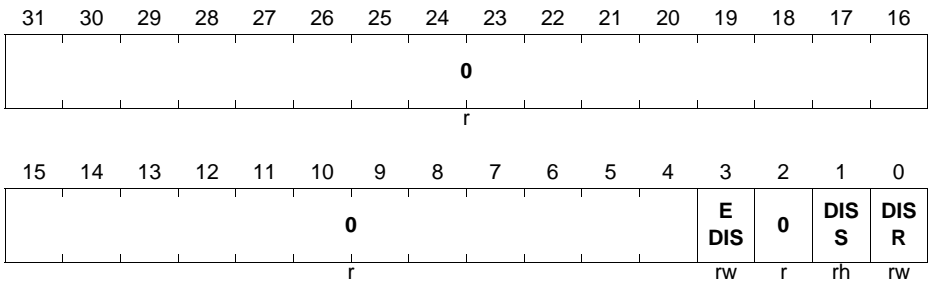
Controller Area Network Controller (MultiCAN+)

CAN Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI for the module. CLC controls the f_{CAN} module clock signal.

CLC

CAN Clock Control Register (000_H) Reset Value: 0000 0003_H



Field	Bits	Type	Description
DISR	0	rw	Module Disable Request Bit Used for enable/disable control of the module. Note that no register access is possible to any register while module is disabled.
DISS	1	rh	Module Disable Status Bit Bit indicates the current status of the module.
EDIS	3	rw	Sleep Mode Enable Control Used to control module's sleep mode.
0	[31:4], 2	r	Reserved Read as 0; should be written with 0.

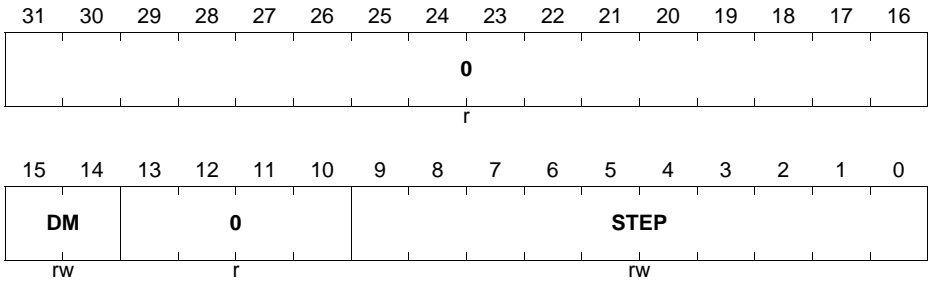
Note: The number of module clock cycles (wait states) which are required by the kernel to execute a read or write access depends on the selected CLC clock frequency.

The fractional divider register allows the programmer to control the clock rate of the module timer clock f_{CAN} .

Controller Area Network Controller (MultiCAN+)

FDR

CAN Fractional Divider Register (00C_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
STEP	[9:0]	rw	Step Value Reload or addition value for the result.
DM	[15:14]	rw	Divider Mode This bit field selects normal divider mode, fractional divider mode, and off-state.
0	[13:10], [31:16]	r	Reserved Read as 0; should be written with 0.

Controller Area Network Controller (MultiCAN+)

18.6.4 Port and I/O Line Control

The interconnections between the MultiCAN+ module and the port I/O lines are controlled in the port logic. Additionally to the port input selection, the following port control operations must be executed:

- Input/output function selection (IOCR registers)
- Pad driver characteristics selection for the outputs (PDR registers)

18.6.4.1 Input/Output Function Selection in Ports

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the MultiCAN+ module are controlled by the port input/output control registers, which are described in the datasheet. In case of discrepancies between datasheet and CAN chapter, the description in the datasheet is correct.

Table 18-16 shows the corresponding pins, sorted by node. Even though the table is pair wise, it is possible to select a different pairing for RXD and TXD. In addition the RXSEL value to be programmed for the Receive Pins is part of this table. For more information on RXSEL please see **Chapter 18.6.4.2**.

The listed CAN port pins are named CAN.Nn, with n = 0 .. 1.

Table 18-16 MultiCAN+ I/O Control Selection and Setup for XMC4300

Node	RXD	NPCR _x .RXSEL	TXD
CAN0	P1.5 / N0_RXDA	000 _B	P1.4 / N0_TXD
	P14.3 / N0_RXDB	001 _B	
CAN1	P1.13 / N1_RXDC	010 _B	P1.12 / N1_TXD
	P1.4 / N1_RXDD	011 _B	P1.5 / N1_TXD
	P2.6 / N1_RXDA	000 _B	P2.7 / N1_TXD
	CAN0INS / N1_RXDF	101 _B	

18.6.4.2 Node Receive Input Selection

Additionally to the I/O control selection, as defined in the datasheet, the selection of a CAN node's receive input line requires that bit field RXSEL in its node port control register NPCRx must be set according to **Table 18-16**. Values for NPCRx.RXSEL other than those of the table mentioned above will result in a recessive receive input for node x. As a hint A results in 0_H, B in 1_H until H resulting in the value of 7_H.

This feature allows, for example, a CAN node which operates in analyzer mode to monitor the receive operations of its neighbor CAN node.

Controller Area Network Controller (MultiCAN+)

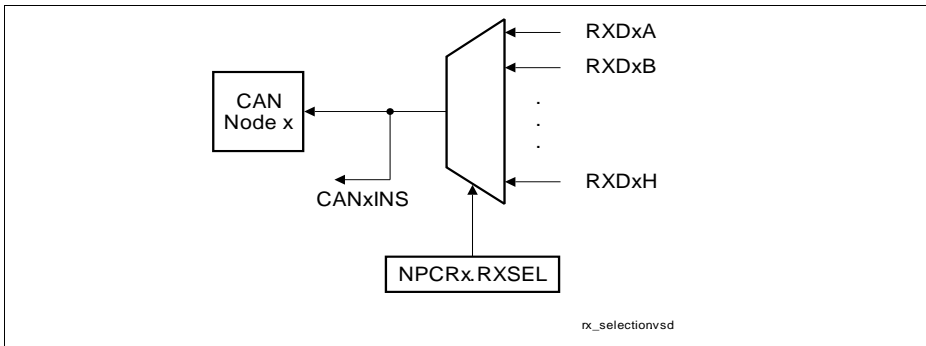


Figure 18-27 CAN Module Receive Input Selection

18.6.5 Connections with other modules

This chapter describes the connections to other modules.

18.6.5.1 Connections to Interrupt Router Inputs

The interrupt output lines INT_O0-7 are connected to the Interrupt Router module, see [Table 18-17](#).

Table 18-17 Interrupt Router Inputs

Interrupt Router Input	Connected to CAN Interrupt Output
SRC_CANINT0	INT_O0
SRC_CANINT1	INT_O1
SRC_CANINT2	INT_O2
SRC_CANINT3	INT_O3
SRC_CANINT4	INT_O4
SRC_CANINT5	INT_O5
SRC_CANINT6	INT_O6
SRC_CANINT7	INT_O7

18.6.5.2 Connections to Capcom4 Inputs

The interrupt output line INT_O7 of MultiCAN+ is connected to the CAPCOM4, see [Table 18-18](#).

Table 18-18 General Timer Module Inputs

CCU4 Input	Connected to CAN Interrupt Output
Timer input (CCU40.IN0H)	INT_07
Timer input (CCU41.IN0H)	

18.6.5.3 Connections to USIC Inputs

The internal signal CAN1INS is connected to the USIC module, see [Table 18-19](#).

Table 18-19 CAN-to-USIC Connections

USIC Input	Connected to CAN Internal Signal
U1C1_DX0E	CAN1INS

18.6.6 Interrupt Control

The interrupt control logic in the MultiCAN+ module uses an interrupt compressing scheme that allows high flexibility in interrupt processing. There are hardware and software interrupt sources available:

- CAN node interrupts:
 - Five different interrupt sources for each of the 2 CAN nodes = $5 * \underline{2}$ interrupt sources
- Message object interrupts:
 - Two interrupt source for each message object = $2 * \underline{64}$ interrupt sources
- One register (MITR) to initiate 16 interrupts via software

Each of the hardware initiated interrupt sources is controlled by a 4-bit interrupt pointer that directs the interrupt source to one of the 8 interrupt outputs INT_Om ($m = 0-7$). This makes it possible to connect more than one interrupt source (between one and all) to one interrupt output line. The interrupt wiring matrix shown in [Figure 18-28](#) is built up according to the following rules:

- Each output of the 4-bit interrupt pointer demultiplexer is connected to exactly one OR-gate input of the INT_Om line. The number “m” of the corresponding selected INT_Om interrupt output line is defined by the interrupt pointer value.
- Each INT_Om output line has an input OR gate which is connected to all interrupt pointer demultiplexer outputs which are selected by an identical 4-bit pointer value.

Controller Area Network Controller (MultiCAN+)

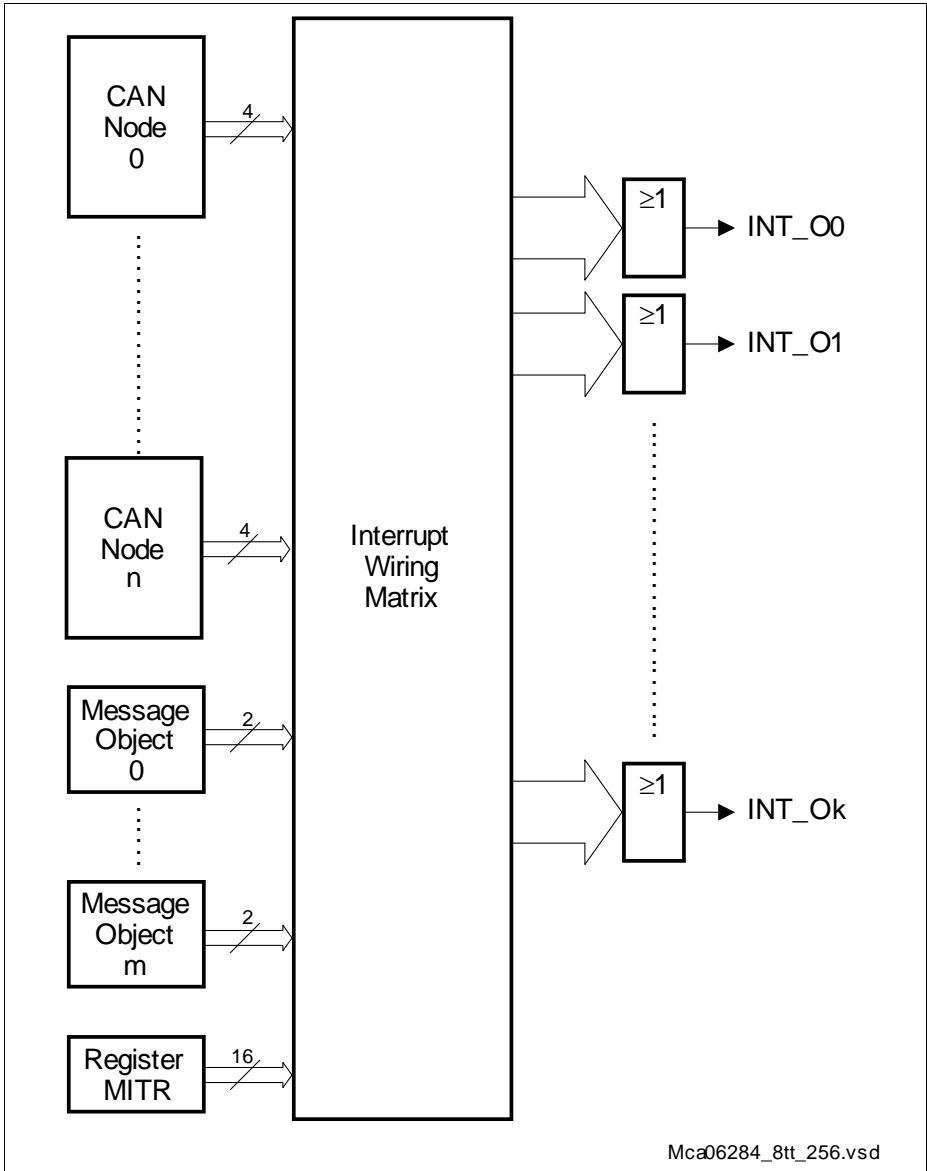


Figure 18-28 Interrupt Compressor $n = \underline{2}$, $m = \underline{64}$ and $k = \underline{8}$

Analog Frontend Peripherals

19 Versatile Analog-to-Digital Converter (VADC)

The XMC4300 provides a series of analog input channels connected to a cluster of Analog/Digital Converters using the Successive Approximation Register (SAR) principle to convert analog input values (voltages) to discrete digital values.

The number of analog input channels and ADCs depends on the chosen product type (please refer to **“Product-Specific Configuration”** on Page 19-132).

Table 19-1 Abbreviations used in ADC chapter

ADC	Analog to Digital Converter
DMA	Direct Memory Access (controller)
DNL	Differential Non-Linearity (error)
INL	Integral Non-Linearity (error)
LSB _n	Least Significant Bit: finest granularity of the analog value in digital format, represented by one least significant bit of the conversion result with n bits resolution (measurement range divided in 2 ⁿ equally distributed steps)
SCU	System Control Unit of the device
TUE	Total Unadjusted Error

19.1 Overview

Each converter of the ADC cluster can operate independent of the others, controlled by a dedicated set of registers and triggered by a dedicated group request source. The results of each channel can be stored in a dedicated channel-specific result register or in a group-specific result register.

A background request source can access all analog input channels that are not assigned to any group request source. These conversions are executed with low priority. The background request source can, therefore, be regarded as an additional background converter.

The Versatile Analog to Digital Converter module (VADC) of the XMC4300 comprises a set of converter blocks that can be operated either independently or via a common request source that emulates a background converter. Each converter block is equipped with a dedicated input multiplexer and dedicated request sources, which together build separate groups.

This basic structure supports application-oriented programming and operating while still providing general access to all resources. The almost identical converter groups allow a flexible assignment of functions to channels.

Versatile Analog-to-Digital Converter (VADC)

The basic module clock f_{ADC} is connected to the system clock signal f_{PERIPH} .

Feature List

The following features describe the functionality of the ADC cluster:

- Nominal analog supply voltage 3.3 V
- Input voltage range from 0 V up to analog supply voltage
- Standard (V_{AREF}) and alternate (CH0) reference voltage source selectable for each channel to support ratiometric measurements and different signal scales
- Up to 2 independent converters with up to 8 analog input channels
- External analog multiplexer control, including adjusted sample time and scan support
- Conversion speed and sample time adjustable to adapt to sensors and reference
- Conversion time below 1 μ s (depending on result width and sample time)
- Flexible source selection and arbitration
 - Programmable arbitrary conversion sequence (single or repeated)
 - Configurable auto scan conversion (single or repeated) on each converter
 - Configurable auto scan conversion (single or repeated) in the background (all converters)
 - Conversions triggered by software, timer events, or external events
 - Cancel-inject-restart mode for reduced conversion delay on priority channels
- Powerful result handling
 - Selectable result width of 8/10/12 bits
 - Fast Compare Mode
 - Independent result registers
 - Configurable limit checking against programmable border values
 - Data rate reduction through adding a selectable number of conversion results
 - FIR/IIR filter with selectable coefficients
- Flexible service request generation based on selectable events
- Built-in safety features
 - Broken wire detection with programmable default levels
 - Multiplexer test mode to verify signal path integrity
- Support of suspend and power saving modes

Note: Additional functions are available from the out of range comparator (see description in the SCU).

Versatile Analog-to-Digital Converter (VADC)

Table 19-2 VADC Applications

Use Case VADC	Application
Automatic scheduling of complex conversion sequences, including prioritization of time-critical conversions	Motor control, Power conversion
Effective result handling for bursts of high-speed conversions	Highly dynamic input signals
Synchronous sampling of up to 4 input signals	Multi-phase current measurement

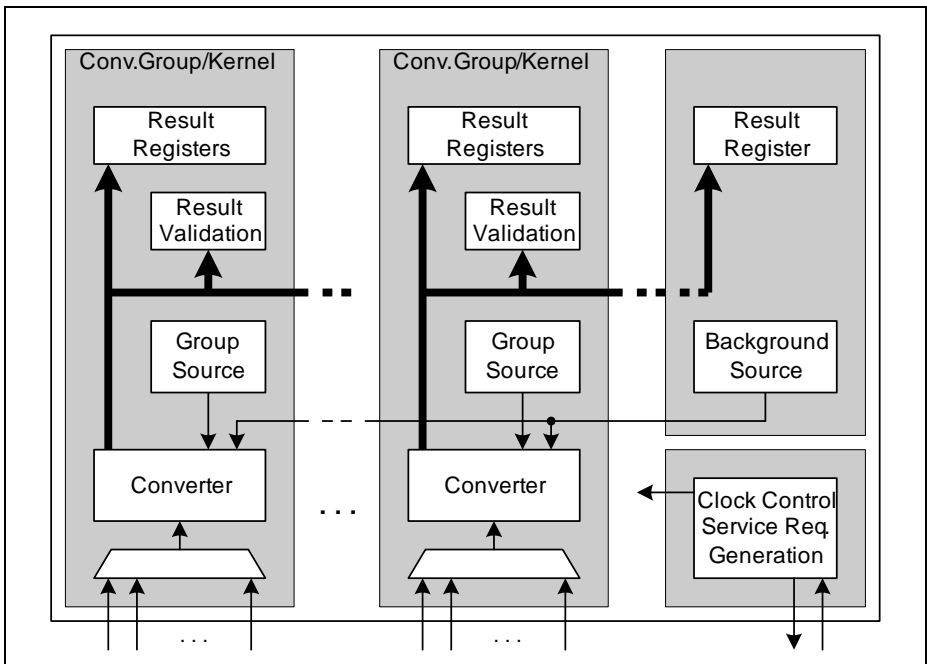


Figure 19-1 ADC Structure Overview

Versatile Analog-to-Digital Converter (VADC)

19.2 Introduction and Basic Structure

The Versatile Analog to Digital Converter module (VADC) of the XMC4300 comprises a set of converter blocks that can be operated either independently or via a common request source that emulates a background converter. Each converter block is equipped with a dedicated input multiplexer and dedicated request sources, which together build separate groups.

This basic structure supports application-oriented programming and operating while still providing general access to all resources. The almost identical converter groups allow a flexible assignment of functions to channels.

A set of functional units can be configured according to the requirements of a given application. These units build a path from the input signals to the digital results.

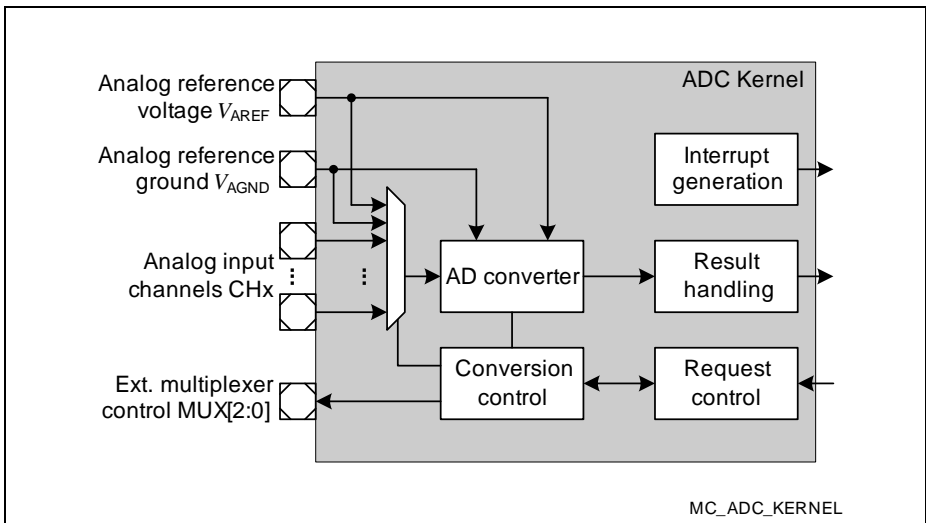


Figure 19-2 ADC Kernel Block Diagram

Versatile Analog-to-Digital Converter (VADC)**Conversion Modes and Request Sources**

Analog/Digital conversions can be requested by several request sources (2 group request sources and the background request source) and can be executed in several conversion modes. The request sources can be enabled concurrently with configurable priorities.

- **Fixed Channel Conversion (single or continuous)**

A specific channel source requests conversions of one selectable channel (once or repeatedly)

- **Auto Scan Conversion (single or continuous)**

A channel scan source (request source 1 or 2) requests auto scan conversions of a configurable linear sequence of all available channels (once or repeatedly)

- **Channel Sequence Conversion (single or continuous)**

A queued source (request source 0) requests a sequence of conversions of up to 8 arbitrarily selectable channels (once or repeatedly)

The conversion modes can be used concurrently by the available request sources, i.e. conversions in different modes can be enabled at the same time. Each source can be enabled separately and can be triggered by external events, such as edges of PWM or timer signals, or pin transitions.

Request Source Control

Because all request sources can be enabled at the same time, an arbiter resolves concurrent conversion requests from different sources. Each source can be triggered by external signals, by on-chip signals, or by software.

Requests with higher priority can either cancel a running lower-priority conversion (cancel-inject-repeat mode) or be converted immediately after the currently running conversion (wait-for-start mode). If the target result register has not been read, a conversion can be deferred (wait-for-read mode).

Certain channels can also be synchronized with other ADC kernels, so several signals can be converted in parallel.

Versatile Analog-to-Digital Converter (VADC)

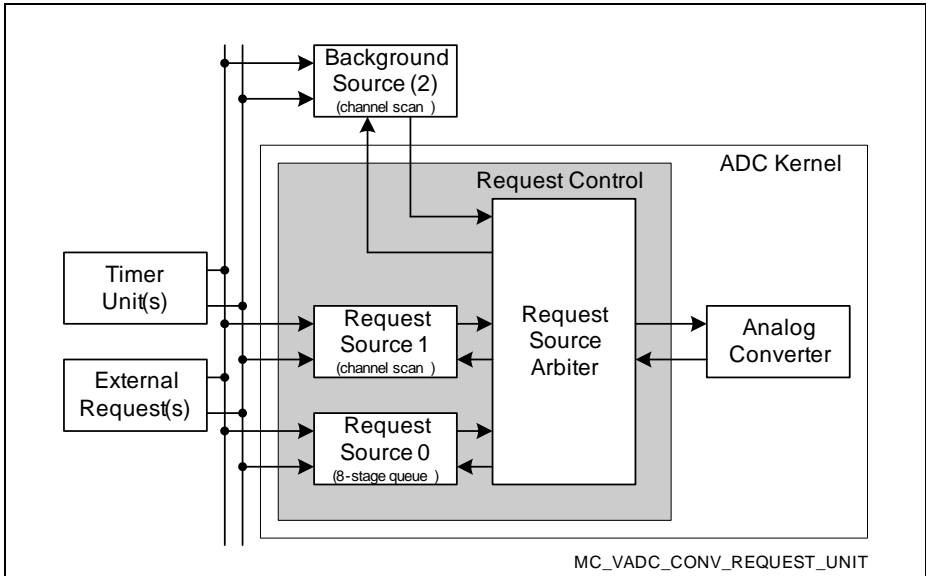


Figure 19-3 Conversion Request Unit

Input Channel Selection

The analog input multiplexer selects one of the available analog inputs (CH0 - CHx¹) to be converted. Three sources can select a linear sequence, an arbitrary sequence, or a specific channel. The priorities of these sources can be configured.

Additional external analog multiplexers can be controlled automatically, if more separate input channels are required than are built in.

Note: Not all analog input channels are necessarily available in all packages, due to pin limitations. Please refer to the implementation description in [Section 19.14](#).

Conversion Control

Conversion parameters, such as sample phase duration, reference voltage, or result resolution can be configured for 4 input classes (2 group-specific classes, 2 global classes). Each channel can be individually assigned to one of these input classes.

The input channels can, thus, be adjusted to the type of sensor (or other analog sources) connected to the ADC.

1) The availability of input channels depends on the package of the used product type. A summary can be found in [Section 19.14.2](#).

Versatile Analog-to-Digital Converter (VADC)

This unit also controls the built-in multiplexer and external analog multiplexers, if selected.

Analog/Digital Converter

The selected input channel is converted to a digital value by first sampling the voltage on the selected input and then generating the selected number of result bits.

For 12-bit conversions, post-calibration is executed after converting the channel.

For broken wire detection (see [Section 19.10.1](#)), the converter network can be preloaded before sampling the selected input channel.

Result Handling

The conversion results of each analog input channel can be directed to one of 16 group-specific result registers and one global result register to be stored there. A result register can be used by a group of channels or by a single channel.

The wait-for-read mode avoids data loss due to result overwrite by blocking a conversion until the previous result has been read.

Data reduction (e.g. for digital anti-aliasing filtering) can automatically add up to 4 conversion results before issuing a service request.

Alternatively, an FIR or IIR filter can be enabled that preprocesses the conversion results before sending them to the result register.

Also, result registers can be concatenated to build FIFO structures that store a number of conversion results without overwriting previous data. This increases the allowed CPU latency for retrieving conversion data from the ADC.

Service Request Generation

Several ADC events can issue service requests to CPU or DMA:

- **Source events** indicate the completion of a conversion sequence in the corresponding request source. This event can be used to trigger the setup of a new sequence.
- **Channel events** indicate the completion of a conversion for a certain channel. This can be combined with limit checking, so interrupt are generated only if the result is within a defined range of values.
- **Result events** indicate the availability of new result data in the corresponding result register. If data reduction mode is active, events are generated only after a complete accumulation sequence.

Each event can be assigned to one of eight service request nodes. This allows grouping the requests according to the requirements of the application.

Versatile Analog-to-Digital Converter (VADC)

Safety Features

Safety-aware applications are supported with mechanisms that help to ensure the integrity of a signal path.

Broken-wire-detection (BWD) preloads the converter network with a selectable level before sampling the input channel. The result will then reflect the preload value if the input signal is no more connected. If buffer capacitors are used, a certain number of conversions may be required to reach the failure indication level.

Pull Down Diagnostics (PDD) connects an additional strong pull-down device to an input channel. A subsequent conversion can then confirm the expected modified signal level. This allows to check the proper connection of a signal source (sensor) to the multiplexer.

Multiplexer Diagnostics (MD) connects a weak pull-up or pull-down device to an input channel. A subsequent conversion can then confirm the expected modified signal level. This allows to check the proper operation of the multiplexer.

Note: These pull-up/pull-down devices are controlled via the port logic.

Converter Diagnostics (CD) connects an alternate signal to the converter. A subsequent conversion can then confirm the proper operation of the converter.

Versatile Analog-to-Digital Converter (VADC)

19.3 Configuration of General Functions

While many parameters can be selected individually for each channel, source, or group, some adjustments are valid for the whole ADC cluster:

- Clock control
- Kernel synchronization
- External multiplexer control
- Test functions

19.3.1 General Clocking Scheme and Control

The A/D Converters of the XMC4300 are supplied with a global clock signal from the system f_{ADC} . Two clock signals are derived from this input and are distributed to all converters. The global configuration register defines common clock bases for all converters of the cluster. This ensures deterministic behavior of converters that shall operate in parallel.

The analog converter clock f_{ADC1} determines the performance of the converters and must be selected to comply with the specification given in the Data Sheet.

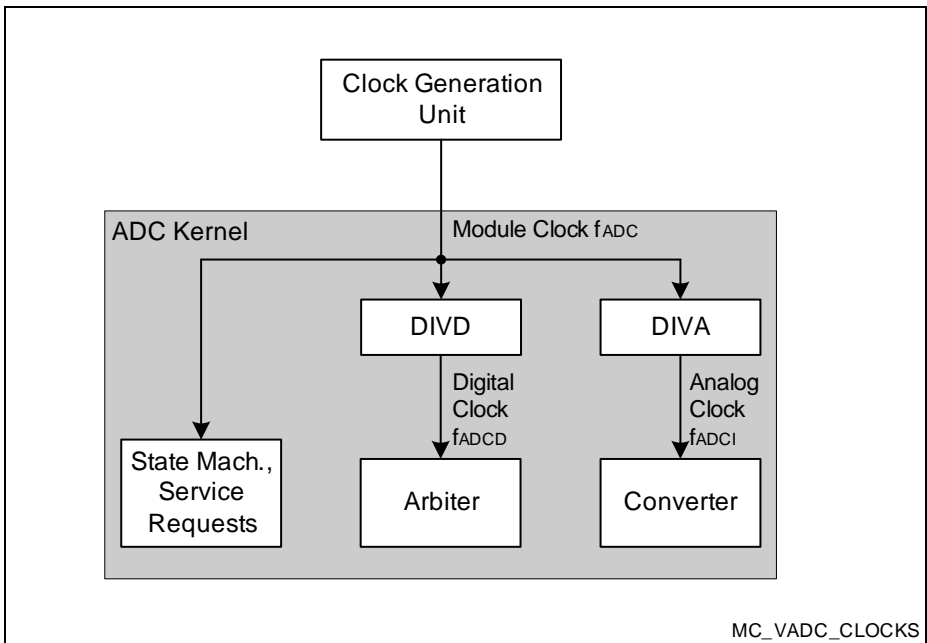


Figure 19-4 Clock Signal Summary

19.3.2 Priority Channel Assignment

Each channel of a group can be assigned to this group's request sources and is then regarded as a priority channel. An assigned priority channel can only be converted by its own group's request sources. A not assigned channel can also be converted by the background request source.

19.4 Module Activation and Power Saving

The analog converter of the ADC draws a permanent current during its operation. It can be deactivated between conversions to reduce the consumed overall energy.

The operating mode is determined by bitfield **GxARBCFG (x = 0 - 1)**.ANONS:

- ANONS = 11_B: **Normal Operation**
The converter is active, conversions are started immediately.
Requires no wakeup time.
- ANONS = 10_B or 01_B: **Reserved**
- ANONS = 00_B: **Converter switched Off** (default after reset)
The converter is switched off. Furthermore, digital logic blocks are set to their initial state. If the arbiter is currently running, it completes the actual arbitration round and then stops.
Before starting a conversion, select the active mode for ANONS.
Requires the wakeup time (see below).

Wakeup Time from Analog Powerdown

When the converter is activated, it needs a certain wakeup time to settle before a conversion can be properly executed. This wakeup time can be established by waiting the required period before starting a conversion, or by adding it to the intended sample time.

The wakeup time is approximately 15 μ s.

Exact numbers can be found in the respective Data Sheets.

Note: The wakeup time is also required after initially enabling the converter.

Calibration

Calibration automatically compensates deviations caused by process, temperature, and voltage variations. This ensures precise results throughout the operation time.

An initial start-up calibration is required once after a reset for all calibrated converters and is triggered globally. All calibrated converters must be enabled (ANONS = 11_B) before initiating the start-up calibration. Conversions may be started after the initial calibration sequence. This is indicated by bit CAL = 0_B.

After that, postcalibration cycles will compensate the effects of drifting parameters.

19.5 Conversion Request Generation

The conversion request unit of a group autonomously handles the generation of conversion requests. Three request sources (2 group-specific sources and the background source) can generate requests for the conversion of an analog channel. The arbiter resolves concurrent requests and selects the channel to be converted next.

Upon a trigger event, the request source requests the conversion of a certain analog input channel or a sequence of channels.

- **Software triggers**
directly activate the respective request source.
- **External triggers**
synchronize the request source activation with external events, such as a trigger pulse from a timer generating a PWM signal or from a port pin.

Application software selects the trigger, the channel(s) to be converted, and the request source priority. A request source can also be activated directly by software without requiring an external trigger.

The arbiter regularly scans the request sources for pending conversion requests and selects the conversion request with the highest priority. This conversion request is then forwarded to the converter to start the conversion of the requested channel.

Each request source can operate in single-shot or in continuous mode:

- **In single-shot mode,**
the programmed conversion (sequence) is requested once after being triggered. A subsequent conversion (sequence) must be triggered again.
- **In continuous mode,**
the programmed conversion (sequence) is automatically requested repeatedly after being triggered once.

For each request source, external triggers are generated from one of 16 selectable trigger inputs (REQTRx[P:A]) and from one of 16 selectable gating inputs (REQGTx[P:A]). The available trigger signals for the XMC4300 are listed in [Section 19.14.3](#).

Note: [Figure 19-3 “Conversion Request Unit” on Page 19-6](#) summarizes the request sources.

Versatile Analog-to-Digital Converter (VADC)

Two types of requests sources are available:

- **A queued source** can issue conversion requests for an arbitrary sequence of input channels. The channel numbers for this sequence can be freely programmed¹⁾. This supports application-specific conversion sequences that cannot be covered by a channel scan source. Also, multiple conversions of the same channel within a sequence are supported.

A queued source converts a series of input channels permanently or on a regular time base. For example, if programmed with medium priority, some input channels can be converted upon a specified event (e.g. synchronized to a PWM). Conversions of lower priority sources are suspended in the meantime.

Request source 0 is a group-specific 8-stage queued source.

- **A channel scan source** can issue conversion requests for a coherent sequence of input channels. This sequence begins with the highest enabled channel number and continues towards lower channel numbers. All available channels¹⁾ can be enabled for the scan sequence. Each channel is converted once per sequence.

A scan source converts a series of input channels permanently or on a regular time base. For example, if programmed with low priority, some input channels can be scanned in a background task to update information that is not time-critical.

- Request source 1 is a group-specific channel scan source.
- Request source 2 is a global channel scan source (background source).

The background source can request all channels of all groups.

1) The availability of input channels depends on the package of the used product type. A summary can be found in [Section 19.14.2](#).

The background source can only request non-priority channels, i.e. channels that are not selected in registers GxCHASS. Priority channels are reserved for the group-specific request sources 0 and 1.

19.5.1 Queued Request Source Handling

A queued request source supports short conversion sequences (up to 8) of arbitrary channels (contrary to a scan request source with a fixed conversion order for the enabled channels). The programmed sequence is stored in a queue buffer (based on a FIFO mechanism). The requested channel numbers are entered via the queue input, while queue stage 0 defines the channel to be converted next.

A conversion request is only issued to the request source arbiter if a valid entry is stored in queue stage 0.

If the arbiter aborts a conversion triggered by a queued request source due to higher priority requests, the corresponding conversion parameters are automatically saved in the backup stage. This ensures that an aborted conversion is not lost but takes part in the next arbitration round (before stage 0).

The trigger and gating unit generates trigger events from the selected external (outside the ADC) trigger and gating signals. For example, a timer unit can issue a request signal to synchronize conversions to PWM events.

Trigger events start a queued sequence and can be generated either via software or via the selected hardware triggers. The occurrence of a trigger event is indicated by bit QSRx.EV. This flag is cleared when the corresponding conversion is started or by writing to bit QMRx.CEV.

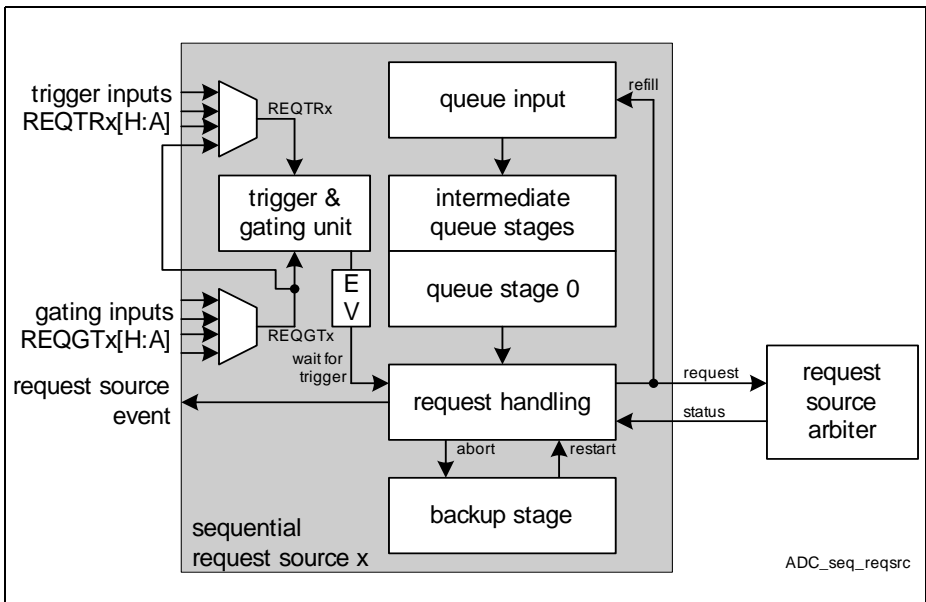


Figure 19-5 Queued Request Source

Versatile Analog-to-Digital Converter (VADC)

A sequence is defined by entering conversion requests into the queue input register (**GxQINR0 (x = 0 - 1)**). Each entry selects the channel to be converted and can enable an external trigger, generation of an interrupt, and an automatic refill (i.e. copy this entry to the top of the queue after conversion). The entries are stored in the queue buffer stages.

The content of stage 0 (**GxQ0R0 (x = 0 - 1)**) selects the channel to be converted next. When the requested conversion is started, the contents of this queue stage is invalidated and copied to the backup stage. Then the next queue entry can be handled (if available).

Note: The contents of the queue stages cannot be modified directly, but only by writing to the queue input or by flushing the queue.

*The current status of the queue is shown in register **GxQSR0 (x = 0 - 1)**.*

If all queue entries have automatic refill selected, the defined conversion sequence can be repeated without re-programming.

Properties of the Queued Request Source

Queued request source 0 provides 8 buffer stages and can handle sequences of up to 8 input channel entries. It supports short application-specific conversion sequences, especially for timing-critical sequences containing also multiple conversions of the same channel.

Queued Source Operation

Configure the queued request source by executing the following actions:

- Define the sequence by writing the entries to the queue input **GxQINR0 (x = 0 - 1)**. Initialize the complete sequence before enabling the request source, because with enabled refill feature, software writes to QINRx are not allowed.
- If hardware trigger or gating is desired, select the appropriate trigger and gating inputs and the proper transitions by programming **GxQCTRL0 (x = 0 - 1)**. Enable the trigger and select the gating mode by programming bitfield ENGT in register **GxQMR0 (x = 0 - 1)**.¹⁾
- Enable the corresponding arbitration slot (0) to accept conversion requests from the queued source (see register **GxARBPR (x = 0 - 1)**).

Start a queued sequence by generating a trigger event:

- If a hardware trigger is selected and enabled, generate the configured transition at the selected input signal, e.g. from a timer or an input pin.
- Generate a software trigger event by setting **GxQMR0.TREV = 1**.

¹⁾ If PDOUT signals from the ERU are used, initialize the ERU accordingly before enabling the gate inputs to avoid an unexpected signal transitions.

Versatile Analog-to-Digital Converter (VADC)

- Write a new entry to the queue input of an empty queue. This leads to a (new) valid queue entry that is forwarded to queue stage 0 and starts a conversion request (if enabled by GxQMR0.ENGTT and without waiting for an external trigger).

Note: If the refill mechanism is activated, a processed entry is automatically reloaded into the queue. This permanently repeats the respective sequence (autoscan). In this case, do not write to the queue input while the queued source is running. Write operations to a completely filled queue are ignored.

Stop or abort an ongoing queued sequence by executing the following actions:

- If external gating is enabled, switch the gating signal to the defined inactive level. This does not modify the queue entries, but only prevents issuing conversion requests to the arbiter.
- Disable the corresponding arbitration slot (0) in the arbiter. This does not modify the queue entries, but only prevents the arbiter from accepting requests from the request handling block.
- Disable the queued source by clearing bitfield ENGT = 00_B.
 - Invalidate the next pending queue entry by setting bit GxQMR0.CLRV = 1. If the backup stage contains a valid entry, this one is invalidated, otherwise stage 0 is invalidated.
 - Remove all entries from the queue by setting bit GxQMR0.FLUSH = 1.

Queue Request Source Events and Service Requests

A request source event of a queued source occurs when a conversion is finished. A source event service request can be generated based on a request source event according to the structure shown in [Figure 19-6](#). If a request source event is detected, it sets the corresponding indication flag in register **GxSEFLAG (x = 0 - 1)**. These flags can also be set by writing a 1 to the corresponding bit position, whereas writing 0 has no effect. The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register **GxSEFCLR (x = 0 - 1)**.

The interrupt enable bit is taken from stage 0 for a normal sequential conversion, or from the backup stage for a repeated conversion after an abort.

The service request output line SR_x that is selected by the request source event interrupt node pointer bitfields in register **GxSEVNP (x = 0 - 1)** becomes activated each time the related request source event is detected (and enabled by GxQ0R0.ENS1, or GxQBUR0.ENS1 respectively) or the related bit position in register **GxSEFLAG (x = 0 - 1)** is written with a 1 (this write action simulates a request source event).

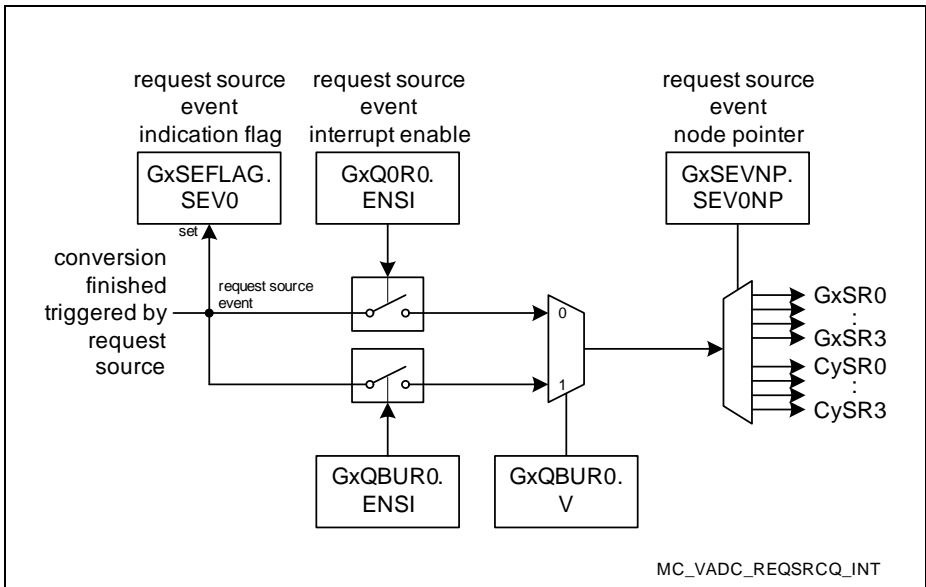


Figure 19-6 Interrupt Generation of a Queued Request Source

19.5.2 Channel Scan Request Source Handling

The VADC provides two types of channel scan sources:

- **Source 1: Group scan source**
This scan source can request all channels of the corresponding group.
- **Source 2: Background scan source**
This scan source can request all channels of all groups.
Priority channels selected in registers **GxCHASS (x = 0 - 1)** cannot take part in background conversion sequences.

Both sources operate in the same way and provide the same register interface. The background source provides more request/pending bits because it can request all channels of all groups.

Each analog input channel can be included in or excluded from the scan sequence by setting or clearing the corresponding channel select bit in register **GxASSEL (x = 0 - 1)** or **BRSELx (x = 0 - 1)**. The programmed register value remains unchanged by an ongoing scan sequence. The scan sequence starts with the highest enabled channel number and continues towards lower channel numbers.

Upon a load event, the request pattern is transferred to the pending bits in register **GxASPND (x = 0 - 1)** or **BRSPNDx (x = 0 - 1)**. The pending conversion requests indicate

Versatile Analog-to-Digital Converter (VADC)

which input channels are to be converted in an ongoing scan sequence. Each conversion start that was triggered by the scan request source, automatically clears the corresponding pending bit. If the last conversion triggered by the scan source is finished and all pending bits are cleared, the current scan sequence is considered finished and a request source event is generated.

A conversion request is only issued to the request source arbiter if at least one pending bit is set.

If the arbiter aborts a conversion triggered by the scan request source due to higher priority requests, the corresponding pending bit is automatically set. This ensures that an aborted conversion is not lost but takes part in the next arbitration round.

The trigger and gating unit generates load events from the selected external (outside the ADC) trigger and gating signals. For example, a timer unit can issue a request signal to synchronize conversions to PWM events.

Load events start a scan sequence and can be generated either via software or via the selected hardware triggers. The request source event can also generate an automatic load event, so the programmed sequence is automatically repeated.

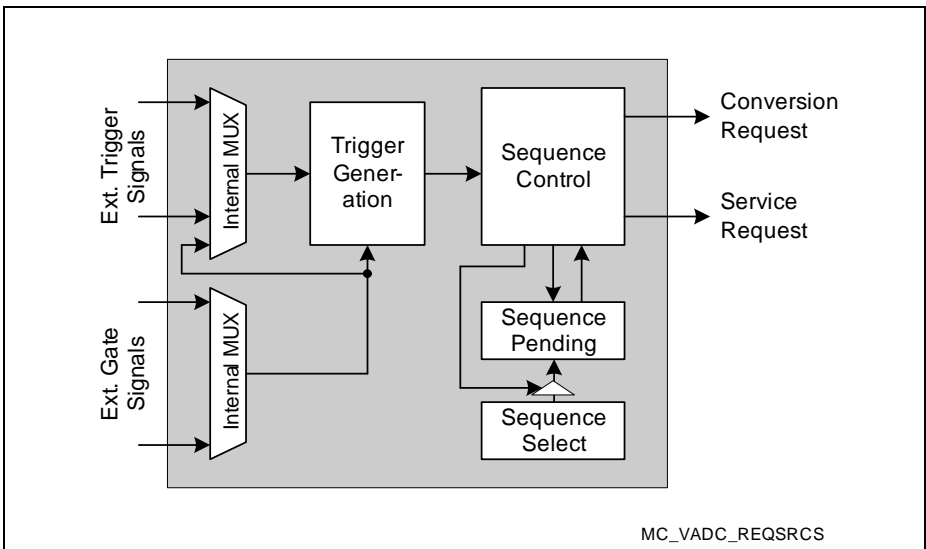


Figure 19-7 Scan Request Source

Scan Source Operation

Configure the scan request source by executing the following actions:

- Select the input channels for the sequence by programming **GxASSEL (x = 0 - 1)** or **BRSELx (x = 0 - 1)**
- If hardware trigger or gating is desired, select the appropriate trigger and gating inputs and the proper signal transitions by programming **GxASCTRL (x = 0 - 1)** or **BRCTRL**. Enable the trigger and select the gating mode by programming **GxASMR (x = 0 - 1)** or **BRSMR**.¹⁾
- Define the load event operation (handling of pending bits, autoscan mode) by programming **GxASMR (x = 0 - 1)** or **BRSMR**.
A load event with bit LDM = 0 copies the content of **GxASSEL (x = 0 - 1)** or **BRSELx (x = 0 - 1)** to **GxASPND (x = 0 - 1)** or **BRSPNDx (x = 0 - 1)** (overwrite mode). This starts a new scan sequence and aborts any pending conversions from a previous scan sequence.
A load event with bit LDM = 1 OR-combines the content of **GxASSEL (x = 0 - 1)** or **BRSELx (x = 0 - 1)** to **GxASPND (x = 0 - 1)** or **BRSPNDx (x = 0 - 1)** (combine mode). This starts a scan sequence that includes pending conversions from a previous scan sequence.
- Enable the corresponding arbitration slot (1) to accept conversion requests from the channel scan source (see register **GxARBPR (x = 0 - 1)**).

Start a channel scan sequence by generating a load event:

- If a hardware trigger is selected and enabled, generate the configured transition at the selected input signal, e.g. from a timer or an input pin.
- Generate a software load event by setting LDEV = 1 (**GxASMR (x = 0 - 1)** or **BRSMR**).
- Generate a load event by writing the scan pattern directly to the pending bits in **GxASPND (x = 0 - 1)** or **BRSPNDx (x = 0 - 1)**. The pattern is copied to **GxASSEL (x = 0 - 1)** or **BRSELx (x = 0 - 1)** and a load event is generated automatically.
In this case, a scan sequence can be defined and started with a single data write action, e.g. under PEC control (provided that the pattern fits into one register).

Note: If autoscan is enabled, a load event is generated automatically each time a request source event occurs when the scan sequence has finished. This permanently repeats the defined scan sequence (autoscan).

Stop or abort an ongoing scan sequence by executing the following actions:

- If external gating is enabled, switch the gating signal to the defined inactive level. This does not modify the conversion pending bits, but only prevents issuing conversion requests to the arbiter.

¹⁾ If PDOUT signals from the ERU are used, initialize the ERU accordingly before enabling the gate inputs to avoid un expected signal transitions.

Versatile Analog-to-Digital Converter (VADC)

- Disable the corresponding arbitration slot (1 or 2) in the arbiter. This does not modify the contents of the conversion pending bits, but only prevents the arbiter from accepting requests from the request handling block.
- Disable the channel scan source by clearing bitfield ENGT = 00_B. Clear the pending request bits by setting bit CLRPND = 1 (**GxASMR (x = 0 - 1)** or **BRSMR**).

Scan Request Source Events and Service Requests

A request source event of a scan source occurs if the last conversion of a scan sequence is finished (all pending bits = 0). A request source event interrupt can be generated based on a request source event. If a request source event is detected, it sets the corresponding indication flag in register **GxSEFLAG (x = 0 - 1)**. These flags can also be set by writing a 1 to the corresponding bit position, whereas writing 0 has no effect.

The service request output SR_x that is selected by the request source event interrupt node pointer bitfields in register **GxSEVNP (x = 0 - 1)** becomes activated each time the related request source event is detected (and enabled by ENSI) or the related bit position in register **GxSEFLAG (x = 0 - 1)** is written with a 1 (this write action simulates a request source event).

The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register **GxSEFCLR (x = 0 - 1)**.¹⁾

1) Please refer to "[Service Request Generation](#)" on [Page 19-56](#).

19.6 Request Source Arbitration

The request source arbiter regularly polls the request sources, one after the other, for pending conversion requests. Each request source is assigned to a certain time slot within an arbitration round, called arbitration slot. The duration of an arbitration slot is user-configurable via register **GLOBCFG**.

The priority of each request source is user-configurable via register **GxARBPR (x = 0 - 1)**, so the arbiter can select the next channel to be converted, in the case of concurrent requests from multiple sources, according to the application requirements.

An unused arbitration slot is considered empty and does not take part in the arbitration. After reset, all slots are disabled and must be enabled (register **GxARBPR (x = 0 - 1)**) to take part in the arbitration process.

Figure 19-8 summarizes the arbitration sequence. An arbitration round consists of one arbitration slot for each available request source. The synchronization source is always evaluated in the last slot and has a higher priority than all other sources. At the end of each arbitration round, the arbiter has determined the highest priority conversion request.

If a conversion is started in an arbitration round, this arbitration round does not deliver an arbitration winner. In the XMC4300, the following request sources are available:

- Arbitration slot 0: **Group Queued source**, 8-stage sequences in arbitrary order
- Arbitration slot 1: **Group Scan source**, sequences in defined order within group
- Arbitration slot 2: **Background Scan source**, sequences in defined order, all groups
- Last arbitration slot: **Synchronization source**, synchronized conversion requests from another ADC kernel (always handled with the highest priority in a synchronization slave kernel).

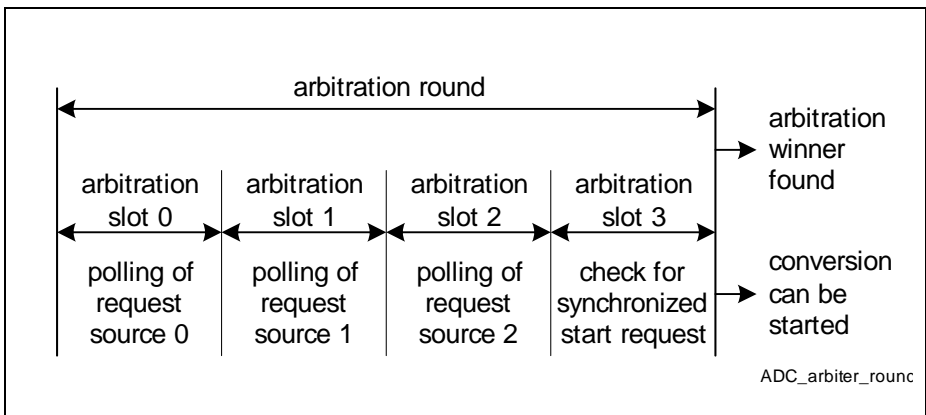


Figure 19-8 Arbitration Round with 4 Arbitration Slots

19.6.1 Arbiter Operation and Configuration

The timing of the arbiter (i.e. of an arbitration round) is determined by the number of arbitration slots within an arbitration round and by the duration of an arbitration slot.

An arbitration round consist of 4...20 arbitration slots (defined by bitfield **GxARBCFG (x = 0 - 1).ARBRND**). 4 slots are sufficient for the XMC4300, more can be programmed to obtain the same arbiter timing for different products.

The duration of an arbitration slot is configurable $t_{\text{Slot}} = (\text{DIVD}+1) / f_{\text{ADC}}$.

The duration of an arbitration round, therefore, is $t_{\text{ARB}} = 4 \times t_{\text{Slot}}$.

The period of the arbitration round introduces a timing granularity to detect an incoming conversion request signal and the earliest point to start the related conversion. This granularity can introduce a jitter of maximum one arbitration round. The jitter can be reduced by minimizing the period of an arbitration round.

To achieve a reproducible reaction time (constant delay without jitter) between the trigger event of a conversion request (e.g. by a timer unit or due to an external event) and the start of the related conversion, mainly the following two options exist. For both options, the converter has to be idle and other conversion requests must not be pending for at least one arbiter round before the trigger event occurs:

- If bit **GxARBCFG (x = 0 - 1).ARBM = 0**, the **arbiter runs permanently**. In this mode, synchronized conversions of more than one ADC kernel are possible.¹⁾
The trigger for a conversion request has to be generated synchronously to the arbiter timing. Incoming triggers should have exactly n-times the granularity of the arbiter (n = 1, 2, 3,...). In order to allow some flexibility, the duration of an arbitration slot can be programmed in cycles of f_{ADC} .
- If bit **GxARBCFG (x = 0 - 1).ARBM = 1**, the **arbiter stops after an arbitration round** when no conversion request have been found pending any more. The arbiter is started again if at least one enabled request source indicates a pending conversion request. The trigger for a conversion request does not need to be synchronous to the arbiter timing.

In this mode, parallel conversions are not possible for synchronization slave kernels.

Each request source has a configurable priority, so the arbiter can resolve concurrent conversion requests from different sources. The request with the highest priority is selected for conversion. These priorities can be adapted to the requirements of a given application (see register **GxARBPR (x = 0 - 1)**).

The **Conversion Start Mode** determines the handling of the conversion request that has won the arbitration.

1) For more information, please refer to **“Synchronization of Conversions” on Page 19-47**.

19.6.2 Conversion Start Mode

When the arbiter has selected the request to be converted next, the handling of this channel depends on the current activity of the converter:

- Converter is currently idle: the conversion of the arbitration winner is started immediately.
- Current conversion has same or higher priority: the current conversion is completed, the conversion of the arbitration winner is started after that.
- Current conversion has lower priority: the action is user-configurable:

- **Wait-for-start mode:** the current conversion is completed, the conversion of the arbitration winner is started after that. This mode provides maximum throughput, but can produce a jitter for the higher priority conversion.

Example in [Figure 19-9](#):

Conversion A is requested (t1) and started (t2). Conversion B is then requested (t3), but started only after completion of conversion A (t4).

- **Cancel-inject-repeat mode:** the current conversion is aborted, the conversion of the arbitration winner is started after the abortion ($3 f_{ADC}$ cycles).

The aborted conversion request is restored in the corresponding request source and takes part again in the next arbitration round. This mode provides minimum jitter for the higher priority conversions, but reduces the overall throughput.

Example in [Figure 19-9](#):

Conversion A is requested (t6) and started (t7). Conversion B is then requested (t8) and started (t9), while conversion A is aborted but requested again. When conversion B is complete (t10), conversion A is restarted.

Exception: If both requests target the same result register with wait-for-read mode active (see [Section 19.8.3](#)), the current conversion cannot be aborted.

Note: A cancelled conversion can be repeated automatically in each case, or it can be discarded if it was cancelled. This is selected for each source by bit RPTDIS in the corresponding source's mode register.

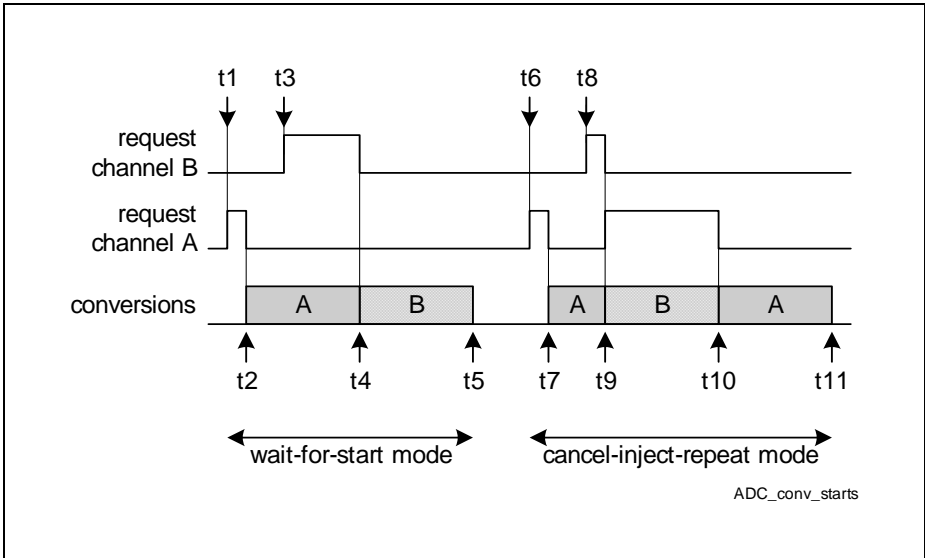


Figure 19-9 Conversion Start Modes

The conversion start mode can be individually programmed for each request source by bits in register **GxARBPR (x = 0 - 1)** and is applied to all channels requested by the source. In this example, channel A is issued by a request source with a lower priority than the request source requesting the conversion of channel B.

19.7 Analog Input Channel Configuration

For each analog input channel a number of parameters can be configured that control the conversion of this channel. The channel control registers define the following parameters:

- **Channel Parameters:** The sample time for this channel and the data width of the result are defined via input classes. Each channel can select one of two classes of its own group or one of two global classes.
- **Reference selection:** an alternate reference voltage can be selected for most channels (exceptions are marked in [Section 19.14.2](#))
- **Result target:** The conversion result values are stored either in a group-specific result register or in the global result register. The group-specific result registers are selected in several ways:
 - channel-specific, selected by bitfield RESREG in register **G0CHCTRY** ($y = 0 - 7$) etc., with bitfield SRCRESREG = 0000_B
 - source-specific, selected by bitfield SRCRESREG in register **GxQCTRL0** ($x = 0 - 1$), **GxASCTRL** ($x = 0 - 1$) or **BRCTRL** with bitfield SRCRESREG $\neq 0000_B$
- **Result position:** The result values can be stored left-aligned or right-aligned. The exact position depends also on the configured result width and on the data accumulation mode.
See also [Figure 19-16 “Result Storage Options” on Page 19-36](#).
- **Compare with Standard Conversions (Limit Checking):** Channel events can be generated whenever a new result value becomes available. Channel event generation can be restricted to values that lie inside or outside a user-configurable band.
In Fast Compare Mode, channel events can be generated depending on the transitions of the (1-bit) result.
- **Broken Wire Detection:** This safety feature can detect a missing connection to an analog signal source (sensor).
- **Synchronization of Conversions:** Synchronized conversions are executed at the same time on several converters.

The **Alias Feature** redirects conversion requests for channels CH0 and/or CH1 to other channels.

19.7.1 Channel Parameters

Each analog input channel is configured by its associated channel control register.

Note: For the safety feature “Broken Wire Detection”, refer to [Section 19.10.1](#).

The following features can be defined for each channel:

- The conversion class defines the result width and the sample time
- Generation of channel events and the result value band, if used
- Target of the result defining the target register and the position within the register

Versatile Analog-to-Digital Converter (VADC)

The group-specific input class registers define the sample time and data conversion mode for each channel of the respective group that selects them via bitfield ICLSEL in its channel control register GxCHCTR_x.

The global input class registers define the sample time and data conversion mode for each channel of any group that selects them via bitfield ICLSEL in its channel control register GxCHCTR_x.

19.7.2 Conversion Timing

The total time required for a conversion depends on several user-definable factors:

- The ADC conversion clock frequency, where $f_{\text{ADCI}} = f_{\text{ADC}} / (\text{DIVA}+1)^{1)}$
- The selected sample time, where $t_{\text{S}} = (2 + \text{STC}) \times t_{\text{ADCI}}$
(STC = additional sample time, see also [Table 19-9](#))
- The selected operating mode (normal conversion / fast compare mode)
- The result width N (8/10/12 bits) for normal conversions
- The post-calibration time PC, if selected (PC = 2, otherwise 0)
- Synchronization steps done at module clock speed

The conversion time is the sum of sample time, conversion steps, and synchronization. It can be computed with the following formulas:

Standard conversions: $t_{\text{CN}} = (2 + \text{STC} + \text{N} + \text{PC}) \times t_{\text{ADCI}} + 2 \times t_{\text{ADC}}$

Fast compare mode: $t_{\text{CN}} = (2 + \text{STC} + 2) \times t_{\text{ADCI}} + 2 \times t_{\text{ADC}}$

The frequency at which conversions are triggered also depends on several configurable factors:

- The selected conversion time, according to the input class definitions. For conversions using an external multiplexer, also the extended sample times count.
- Delays induced by cancelled conversions that must be repeated.
- Delays due to equidistant sampling of other channels.
- The configured arbitration cycle time.
- The frequency of external trigger signals, if enabled.

Timing Examples

System assumptions:

$f_{\text{ADC}} = 120 \text{ MHz}$ i.e. $t_{\text{ADC}} = 8.3 \text{ ns}$, $\text{DIVA} = 3$, $f_{\text{ADCI}} = 30 \text{ MHz}$ i.e. $t_{\text{ADCI}} = 33.3 \text{ ns}$

According to the given formulas the following minimum conversion times can be achieved:

12-bit calibrated conversion:

$$t_{\text{CN12C}} = (2 + 12 + 2) \times t_{\text{ADCI}} + 2 \times t_{\text{ADC}} = 16 \times 33.3 \text{ ns} + 2 \times 8.3 \text{ ns} = 550 \text{ ns}$$

10-bit uncalibrated conversion:

$$t_{\text{CN10}} = (2 + 10) \times t_{\text{ADCI}} + 2 \times t_{\text{ADC}} = 12 \times 33.3 \text{ ns} + 2 \times 8.3 \text{ ns} = 417 \text{ ns}$$

Fast comparison:

$$t_{\text{FCM}} = (2 + 2) \times t_{\text{ADCI}} + 2 \times t_{\text{ADC}} = 4 \times 33.3 \text{ ns} + 2 \times 8.3 \text{ ns} = 150 \text{ ns}$$

1) The minimum prescaler factor for calibrated converters is 2.

19.7.3 Alias Feature

The Alias Feature redirects conversion requests for channels CH0 and/or CH1 to other channel numbers. This feature can be used to trigger conversions of the same input channel by independent events and to store the conversion results in different result registers.

- The same signal can be measured twice without the need to read out the conversion result to avoid data loss. This allows triggering both conversions quickly one after the other and being independent from CPU/DMA service request latency.
- The sensor signal is connected to only one analog input (instead of two analog inputs). This saves input pins in low-cost applications and only the leakage of one input has to be considered in the error calculation.
- Even if the analog input CH0 is used as alternative reference (see [Figure 19-10](#)), the internal trigger and data handling features for channel CH0 can be used.
- The channel settings for both conversions can be different (boundary values, service requests, etc.).

In typical low-cost AC-drive applications, only one common current sensor is used to determine the phase currents. Depending on the applied PWM pattern, the measured value has different meanings and the sample points have to be precisely located in the PWM period. [Figure 19-10](#) shows an example where the sensor signal is connected to one input channel (CHx) but two conversions are triggered for two different channels (CHx and CH0). With the alias feature, a conversion request for CH0 leads to a conversion of the analog input CHx instead of CH0, but taking into account the settings for CH0. Although the same analog input (CHx) has been measured, the conversion results can be stored and read out from the result registers RESx (conversion triggered for CHx) and RESy (conversion triggered for CH0). Additionally, different interrupts or limit boundaries can be selected, enabled or disabled.

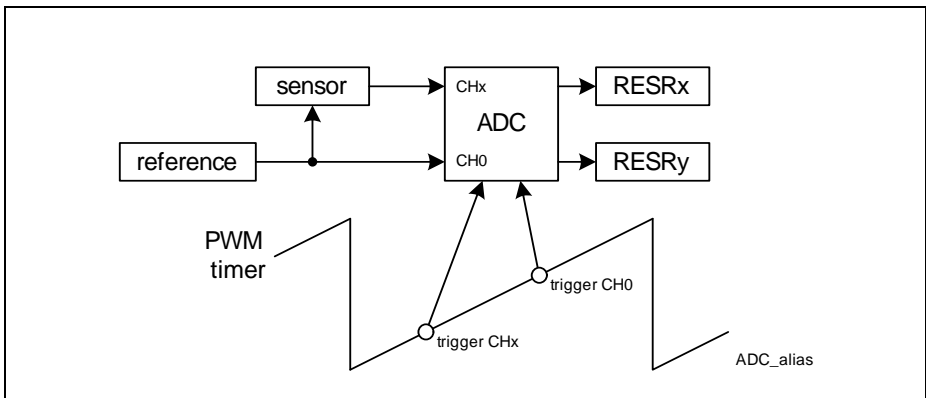


Figure 19-10 Alias Feature

19.7.4 Conversion Modes

A conversion can be executed in several ways. The conversion mode is selected according to the requested resolution of the digital result and according to the acceptable conversion time ([Section 19.7.2](#)).

Use bitfield CMS/CME in register **GxICLASS0 (x = 0 - 1)** etc. to select a mode.

Standard Conversions

A standard conversion returns a result value with a predefined resolution. 8-bit, 10-bit, and 12-bit resolution can be selected.

These result values can be accumulated, filtered, or used for digital limit checking and determination of extrema.

Note: The calibrated converters can operate with and without post-calibration.

Fast Compare Mode

In Fast Compare Mode, the selected input voltage is directly compared with a digital value that is stored in the corresponding result register. This compare operation returns a binary result indicating if the compared input voltage is above or below the given reference value. This result is generated quickly and thus supports monitoring of boundary values.

Fast Compare Mode uses a 10-bit compare value stored left-aligned at bit position 11. Separate positive and negative delta values define an arbitrary hysteresis band.

Selecting Compare Values

Values for digital or analog compare operations can be selected from several sources. The separate GxBOUND registers provide software-defined compare values. Compare values can also be taken from a result register where it can be provided by another channel building a reference.

In Fast Compare Mode, the result registers provide the compare value while the bitfields of local GxBOUND registers define positive and negative delta values.

Versatile Analog-to-Digital Converter (VADC)

19.7.5 Compare with Standard Conversions (Limit Checking)

The limit checking mechanism can automatically compare each digital conversion result to an upper and a lower boundary value. A channel event can then be generated when the result of a conversion/comparison is inside or outside a user-defined band (see bitfield CHEVMODE and [Figure 19-11](#)).

This feature supports automatic range monitoring and minimizes the CPU load by issuing service requests only under certain predefined conditions.

Note: Channel events can also be generated for each result value (ignoring the band) or they can be suppressed completely.

The boundary values to which results are compared can be selected from several sources (see register GxCHCTRY).

While bitfield BNDSELX = 0000_B, bitfields BNDSELU and BNDSELL select the valid upper/lower boundary value either from the group-specific boundary register **GxBOUND (x = 0 - 1)** or from the global boundary register **GLOBBOUND**. The group boundary register can be selected for each channel of the respective group, the global boundary register can be selected by each available channel.

Otherwise, the compare values are taken from result registers, where bitfield BNDSELX selects the upper boundary value (GxRES1 ... GxRES15), the concatenated bitfields BNDSELU||BNDSELL select the lower boundary value (GxRES0 ... GxRES15).

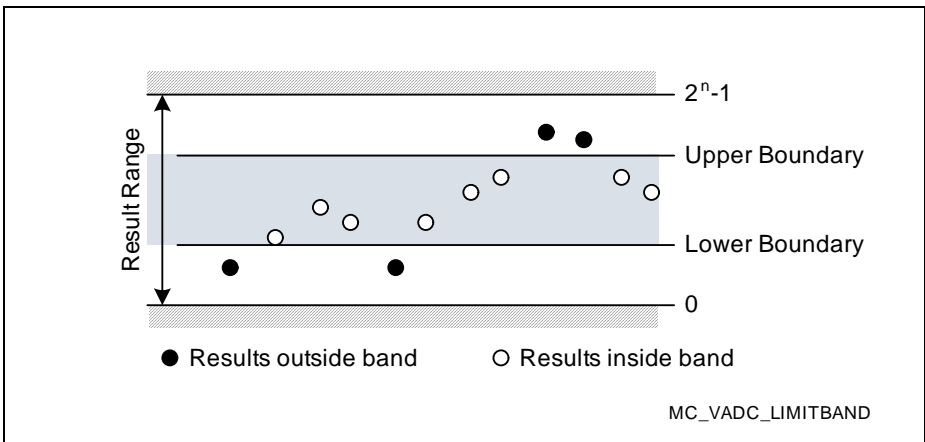


Figure 19-11 Result Monitoring through Limit Checking

Versatile Analog-to-Digital Converter (VADC)

A result value is considered inside the defined band when both of the following conditions are true:

- the value is less than or equal to the selected upper boundary
- the value is greater than or equal to the selected lower boundary

The result range can also be divided into two areas:

To select the lower part as valid band, set the lower boundary to the minimum value (000_{H}) and set the upper boundary to the highest intended value.

To select the upper part as valid band, set the upper boundary to the maximum value (FFF_{H}) and set the lower boundary to the lowest intended value.

Finding Extrema (Peak Detection)

The limit checking mechanism uses standard conversions and, therefore, always can provide the actual conversion result that was used for comparison. Combining this with a special FIFO mode, that only updates the corresponding FIFO stage if the result was above (or below) the current value of the stage, provides the usual conversion results and at the same time stores the highest (or lowest) result of a conversion sequence. For this operation the FIFO stage below the standard result must be selected as the compare value. Mode selection is done via bitfield FEN in register GxRCRy.

Before starting a peak detection sequence, write a reasonable start value to the result bitfield in the peak result register (e.g. 0000_{H} to find the maximum and FFFF_{H} to find the minimum).

Versatile Analog-to-Digital Converter (VADC)

19.7.6 Utilizing Fast Compare Mode

In Fast Compare Mode, the input signal is directly compared to a value in the associated result register. This comparison just provides a binary result (above/below). If the exact result value is not required, this saves conversion time. A channel event can then be generated when the input signal becomes higher (or lower) than the compare value (see bitfield CHEVMODE and [Figure 19-12](#)).

The compare value in Fast Compare Mode is taken from the result register. Bitfields BOUNDARY1 and BOUNDARY0 in register **GxBOUND (x = 0 - 1)** define delta limits in this case. These deltas are added to (or subtracted from) the original compare value and allow defining an arbitrary hysteresis band.

The actual used compare value depends on the Fast Compare Result FCR (see registers **GORES_y (y = 0 - 15)**, etc.):

- GxRES_y.FCR = 0: reference value + upper delta
(GxRES_y.RESULT + GxBOUND.BOUNDARY0)
- GxRES_y.FCR = 1: reference value - lower delta
(GxRES_y.RESULT - GxBOUND.BOUNDARY1)

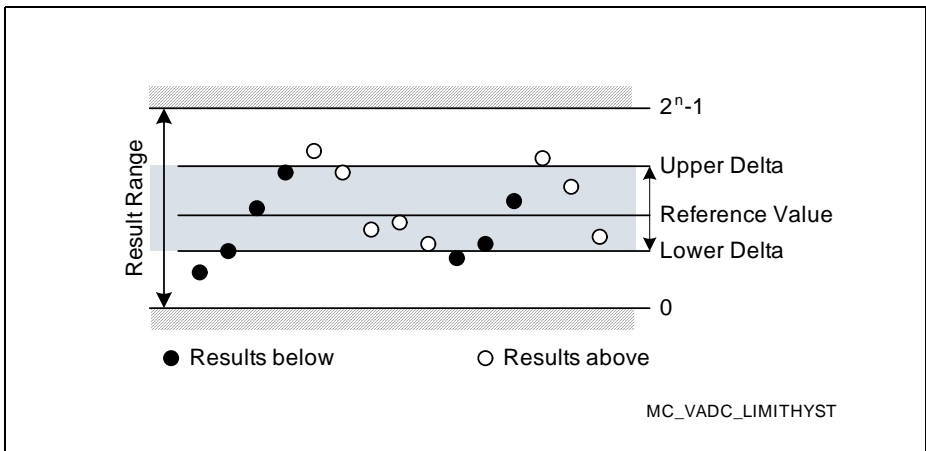


Figure 19-12 Result Monitoring through Compare with Hysteresis

19.7.7 Boundary Flag Control

Both limit checking mechanisms can be configured to automatically control the boundary flags. These boundary flags are also available as control signals for other modules. The flags can be set or cleared when the defined level is exceeded and the polarity of the output signal can be selected. A gate signal can be selected to enable the boundary flag operation while the gate is active.

Each boundary flag is available at group-specific output lines. Node pointers additionally route them to one of four boundary signals or one of the associated common service request lines, see register **GxBFLNP (x = 0 - 1)**.

For standard conversions, a boundary flag will be set/cleared when the conversion result is above the defined band, and will be cleared/set when the conversion result is below the defined band.

The band between the two boundary values defines a hysteresis for setting/clearing the boundary flags.

Using this feature on three channels that monitor linear hall elements can produce signals to feed the three hall position inputs of a unit that generates the corresponding PWM control signals.

In Fast Compare Mode, a boundary flag reflects the result of the comparisons, i.e. it will be set/cleared when the compared signal level is above the compare value, and will be cleared/set when the signal level is below the compare value. The delta values define a hysteresis band around the compare value.

Note: Clear register GxBOUND (i.e. the deltas) if a hysteresis is not wanted.

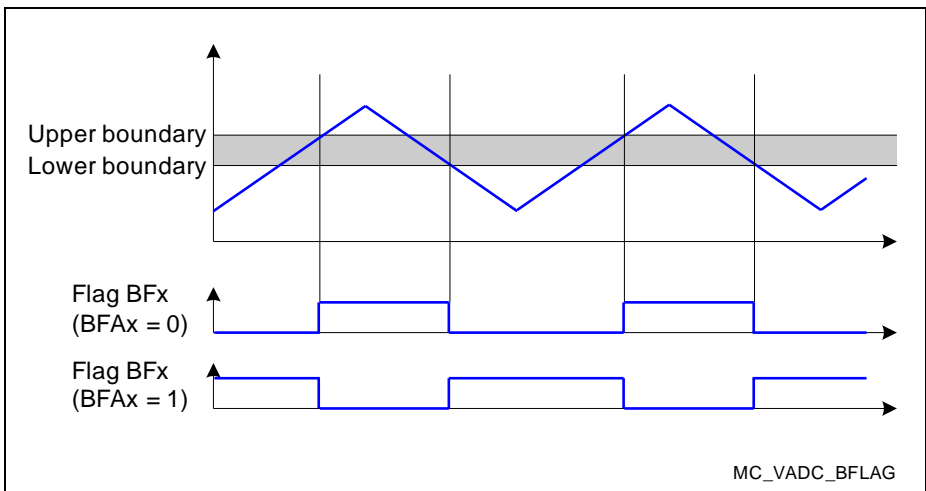


Figure 19-13 Boundary Flag Switching

Versatile Analog-to-Digital Converter (VADC)

Boundary flags can be switched by each compare operation, or the influence of compare operations can be restricted to the active phases of the corresponding request source gate signal (see **GxBFLC (x = 0 - 1)**).

Note: If a boundary flag is used together with Fast Compare Mode, it is recommended not to direct results from other channels to the corresponding result register.

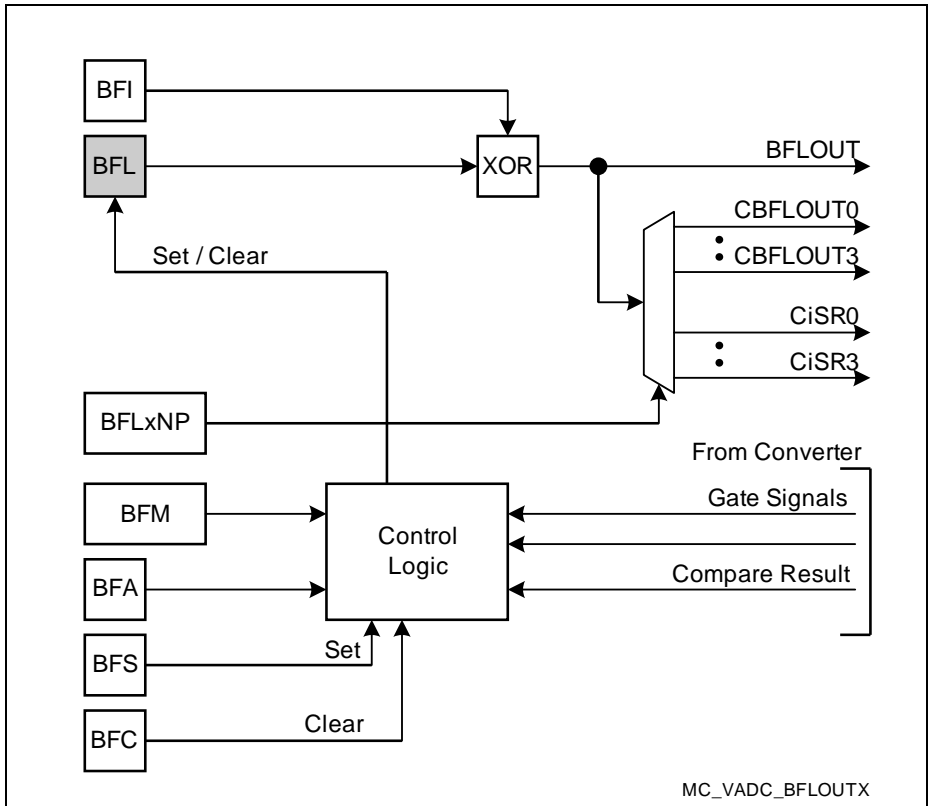


Figure 19-14 Boundary Flag Control

A boundary flag BFLy is assigned to result register GxRESy and thus to an arbitrary channel.

19.8 Conversion Result Handling

The A/D converters can preprocess the conversions result data to a certain extent before storing them for retrieval by the CPU or a DMA channel. This supports the subsequent handling of result data by the application software.

Conversion result handling comprises the following functions:

- **Storage of Conversion Results** to user-configurable registers
- **Data Alignment** according to result width and endianness
- **Wait-for-Read Mode** to avoid loss of data
- **Result Event Generation**
- Data reduction or anti-aliasing filtering (see [Section 19.8.6](#))

19.8.1 Storage of Conversion Results

The conversion result values of a certain group can be stored in one of the 16 associated group result registers or in the common global result register (can be used, for example, for the channels of the background source (see [Selecting a Result Register](#)).

This structure provides different locations for the conversion results of different sets of channels. Depending on the application needs (data reduction, auto-scan, alias feature, etc.), the user can distribute the conversion results to minimize CPU load and/or optimize the performance of DMA transfers.

Each result register has an individual data valid flag (VF) associated with it. This flag indicates when “new” valid data has been stored in the corresponding result register and can be read out.

For standard conversions, result values are available in bitfield RESULT. Conversions in Fast Compare Mode use bitfield RESULT for the reference value, so the result of the operation is stored in bit FCR.

Result registers can be read via two different views. These views use different addresses but access the same register data:

- When a result register is read via the **application view**, the corresponding valid flag is automatically cleared when the result is read. This provides an easy handshake between result generation and retrieval. This also supports wait-for-read mode.
- When a result register is read via the **debug view**, the corresponding valid flag remains unchanged when the result is read. This supports debugging by delivering the result value without disturbing the handshake with the application.

The application can retrieve conversion results through several result registers:

- Group result register:
Returns the result value and the channel number
- Global result register:
Returns the result value and the channel number and the group number

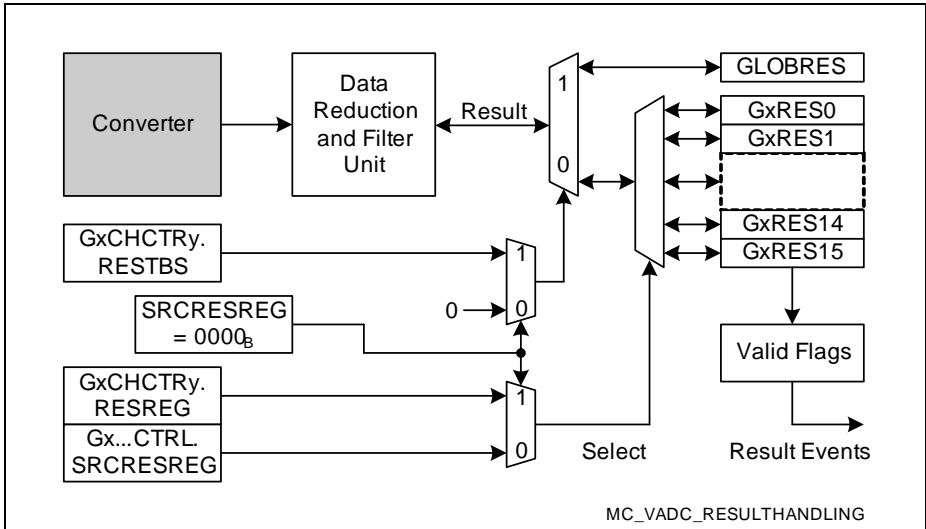


Figure 19-15 Conversion Result Storage

Selecting a Result Register

Conversion results are stored in result registers that can be assigned by the user according to the requirements of the application. The following bitfields direct the results to a register:

- SRCRESREG in register **GxQCTRL0 (x = 0 - 1)**, **GxASCTRL (x = 0 - 1)** or **BRCTRL**
Selects the group-specific result register GxRES1 ... GxRES15 when source-specific result registers are used
- RESTBS in register **G0CHCTry (y = 0 - 7)** etc.
Selects the global result register for results requested by the background source
- RESREG in register **G0CHCTry (y = 0 - 7)** etc.
Selects the group-specific result register GxRES0 ... GxRES15 when channel-specific result registers are used

Using source-specific result registers allows separating results from the same channel that are requested by different request sources. Usually these request sources are used by different tasks and are triggered at different times.

Versatile Analog-to-Digital Converter (VADC)

19.8.2 Data Alignment

The position of a conversion result value within the selected result register depends on 3 configurations (summary in **Figure 19-16**):

- The selected result width (12/10/8 bits, selected by the conversion mode)
- The selected result position (Left/Right-aligned)
- The selected data accumulation mode (data reduction)

These options provide the conversion results in a way that minimizes data handling for the application software.

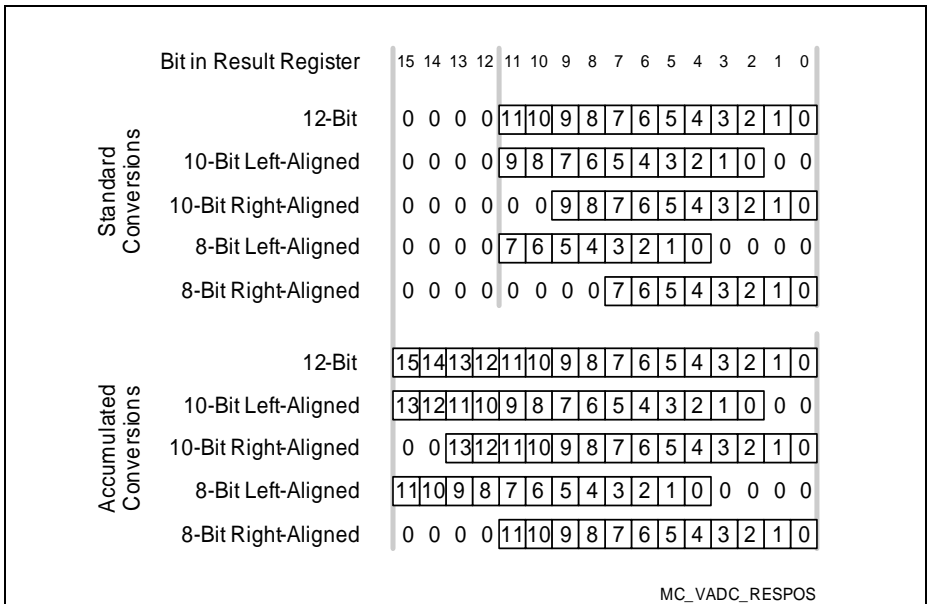


Figure 19-16 Result Storage Options

Bitfield RESULT can be written by software to provide the reference value for Fast Compare Mode. In this mode, bits 11-2 are evaluated, the other bits are ignored.

19.8.3 Wait-for-Read Mode

The wait-for-read mode prevents data loss due to overwriting a result register with a new conversion result before the CPU (or DMA) has read the previous data. For example, auto-scan conversion sequences or other sequences with “relaxed” timing requirements may use a common result register. However, the results come from different input channels, so an overwrite would destroy the result from the previous conversion¹⁾.

Wait-for-read mode automatically suspends the start of a conversion for this channel from this source until the current result has been read. So a conversion or a conversion sequence can be requested by a hardware or software trigger, while each conversion is only started after the result of the previous one has been read. This automatically aligns the conversion sequence with the CPU/DMA capability to read the formerly converted result (latency).

If wait-for-read mode is enabled for a result register (bit GxRCRy.WFR = 1), a request source does not generate a conversion request while the targeted result register contains valid data (indicated by the valid flag VF = 1) or if a currently running conversion targets the same result register.

If two request sources target the same result register with wait-for-read mode selected, a higher priority source cannot interrupt a lower priority conversion request started before the higher priority source has requested its conversion. Cancel-inject-repeat mode does not work in this case. In particular, this must be regarded if one of the involved sources is the background source (which usually has lowest priority). If the higher priority request targets a different result register, the lower priority conversion can be cancelled and repeated afterwards.

Note: Wait-for-read mode is ignored for synchronized conversions of synchronization slaves (see [Section 19.9](#)).

1) Repeated conversions of a single channel that use a separate result register will not destroy other results, but rather update their own previous result value. This way, always the actual signal data is available in the result register.

Versatile Analog-to-Digital Converter (VADC)

19.8.4 Result FIFO Buffer

Result registers can either be used as direct target for conversion results or they can be concatenated with other result registers of the same ADC group to form a result FIFO buffer (first-in-first-out buffer mechanism). A result FIFO stores several measurement results that can be read out later with a “relaxed” CPU response timing. It is possible to set up more than one FIFO buffer structure with the available result registers.

Result FIFO structures of two or more registers are built by concatenating result registers to their following “neighbor” result register (with next higher index, see [Figure 19-17](#)). This is enabled by setting bitfield GxRCRy.FEN = 01_B.

Conversion results are stored to the register with the highest index of a FIFO structure. Software reads the values from the FIFO register with the lowest index.

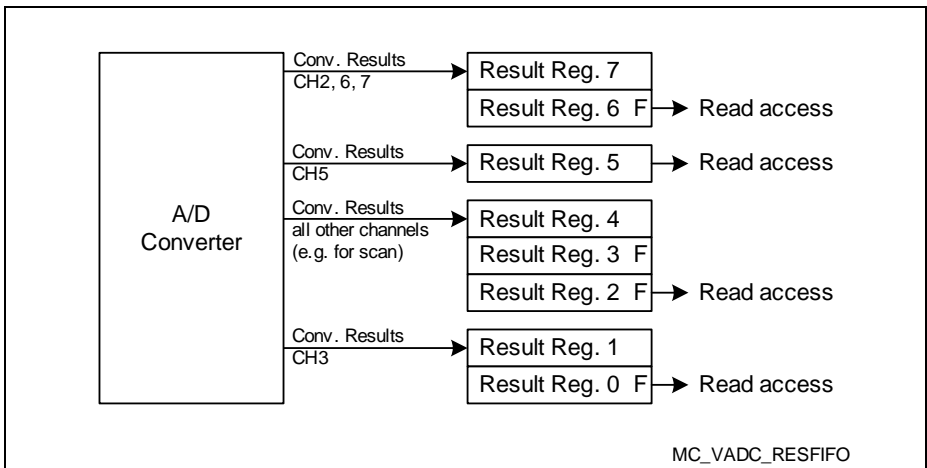


Figure 19-17 Result FIFO Buffers

In the example shown the result registers have been configured in the following way:

- 2-stage buffer consisting of result registers 7-6
- dedicated result register 5
- 3-stage buffer consisting of result registers 4-3-2
- 2-stage buffer consisting of result registers 1-0

Table 19-3 summarizes the required configuration of result registers if they are combined to build result FIFO buffers.

Table 19-3 Properties of Result FIFO Registers

Function	Input Stage	Intermed. Stage	Output Stage
Result target	YES	no	no
Application read	no	no	YES
Data reduction mode	YES	no	no
Wait-for-read mode	YES	no	no
Result event interrupt	no	no	YES
FIFO enable (FEN)	00 _B	01 _B	01 _B
Registers in example	7, 4, 1	3	6, 2, 0

Note: If enabled, a result interrupt is generated for each data word in the FIFO.

19.8.5 Result Event Generation

A result event can be generated when a new value is stored to a result register. Result events can be restricted due to data accumulation and be generated only if the accumulation is complete.

Result events can also be suppressed completely.

19.8.6 Data Modification

The data resulting from conversions can be automatically modified before being used by an application. Several options can be selected (bitfield DMM in register **G0RCRy (y = 0 - 15)** etc.) which reduce the CPU/DMA load required to unload and/or process the conversion data.

- **Standard Data Reduction Mode (for GxRES0 ... GxRES15):**
Accumulates 2, 3, or 4 result values within each result register before generating a result interrupt. This can remove some noise from the input signal.
- **Result Filtering Mode (FIR, for GxRES7, GxRES15):**
Applies a 3rd order Finite Impulse Response Filter (FIR) with selectable coefficients to the conversion results for the selected result register.
- **Result Filtering Mode (IIR, for GxRES7, GxRES15):**
Applies a 1st order Infinite Impulse Response Filter (IIR) with selectable coefficients to the conversion results for the selected result register.
- **Difference Mode (for GxRES1 ... GxRES15):**
Subtracts the contents of result register GxRES0 from the conversion results for the selected result register. Bitfield DRCTR is not used in this mode.

Table 19-4 Function of Bitfield DRCTR

DRCTR	Standard Data Reduction Mode (DMM = 00 _B)	DRCTR	Result Filtering Mode (DMM = 01 _B) ¹⁾
0000 _B	Data Reduction disabled	0000 _B	FIR filter: a=2, b=1, c=0
0001 _B	Accumulate 2 result values	0001 _B	FIR filter: a=1, b=2, c=0
0010 _B	Accumulate 3 result values	0010 _B	FIR filter: a=2, b=0, c=1
0011 _B	Accumulate 4 result values	0011 _B	FIR filter: a=1, b=1, c=1
0100 _B	Reserved	0100 _B	FIR filter: a=1, b=0, c=2
0101 _B	Reserved	0101 _B	FIR filter: a=3, b=1, c=0
0110 _B	Reserved	0110 _B	FIR filter: a=2, b=2, c=0
0111 _B	Reserved	0111 _B	FIR filter: a=1, b=3, c=0
1000 _B	Reserved	1000 _B	FIR filter: a=3, b=0, c=1
1001 _B	Reserved	1001 _B	FIR filter: a=2, b=1, c=1
1010 _B	Reserved	1010 _B	FIR filter: a=1, b=2, c=1
1011 _B	Reserved	1011 _B	FIR filter: a=2, b=0, c=2
1100 _B	Reserved	1100 _B	FIR filter: a=1, b=1, c=2
1101 _B	Reserved	1101 _B	FIR filter: a=1, b=0, c=3

Versatile Analog-to-Digital Converter (VADC)

Table 19-4 Function of Bitfield DRCTR (cont'd)

DRCTR	Standard Data Reduction Mode (DMM = 00_B)	DRCTR	Result Filtering Mode (DMM = 01_B)¹⁾
1110 _B	Reserved	1110 _B	IIR filter: a=2, b=2
1111 _B	Reserved	1111 _B	IIR filter: a=3, b=4

1) The filter registers are cleared while bitfield DMM ≠ 01_B.

Versatile Analog-to-Digital Converter (VADC)

Standard Data Reduction Mode

The data reduction mode can be used as digital filter for anti-aliasing or decimation purposes. It accumulates a maximum of 4 conversion values to generate a final result.

Each result register can be individually enabled for data reduction, controlled by bitfield DRCTR in registers **G0CHCTry (y = 0 - 7)**. The data reduction counter DRC indicates the actual status of the accumulation.

Note: Conversions for other result registers can be inserted between conversions to be accumulated.

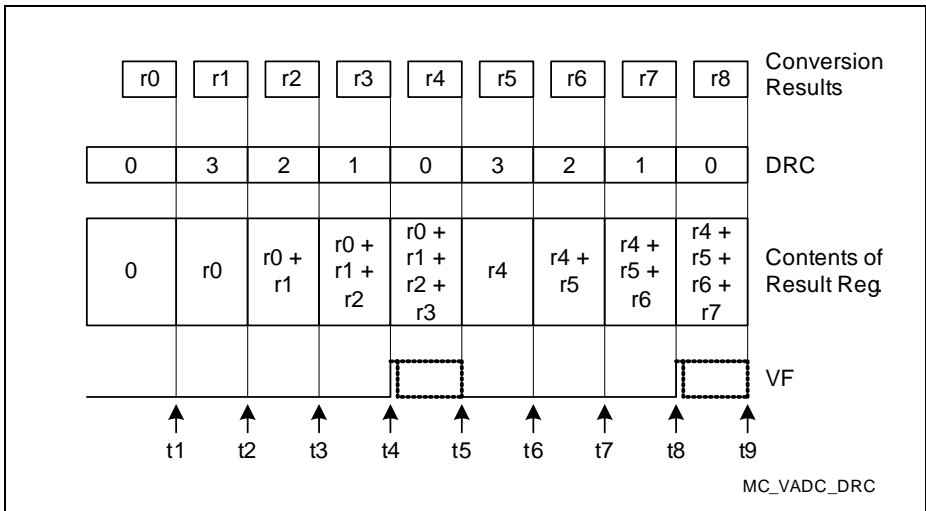


Figure 19-18 Standard Data Reduction Filter

This example shows a data reduction sequence of 4 accumulated conversion results. Eight conversion results (r0 ... r7) are accumulated and produce 2 final results.

When a conversion is complete and stores data to a result register that has data reduction mode enabled, the data handling is controlled by the data reduction counter DRC:

- If DRC = 0 (t1, t5, t9 in the example), the conversion result is stored to the register. DRC is loaded with the contents of bitfield DRCTR (i.e. the accumulation begins).
- If DRC > 0 (t2, t3, t4 and t6, t7, t8 in the example), the conversion result is added to the value in the result register. DRC is decremented by 1.
- If DRC becomes 0, either decremented from 1 (t4 and t8 in the example) or loaded from DRCTR, the valid bit for the respective result register is set and a result register event occurs.

Versatile Analog-to-Digital Converter (VADC)

The final result must be read before the next data reduction sequence starts (before t5 or t9 in the example). This automatically clears the valid flag.

*Note: Software can clear the data reduction counter DRC by clearing the corresponding valid Flag (via **GxVFR (x = 0 - 1)**).*

The response time to read the final data reduction results can be increased by associating the adjacent result register to build a result FIFO (see **Figure 19-19**). In this case, the final result of a data reduction sequence is loaded to the adjacent register. The value can be read from this register until the next data reduction sequence is finished (t8 in the 2nd example).

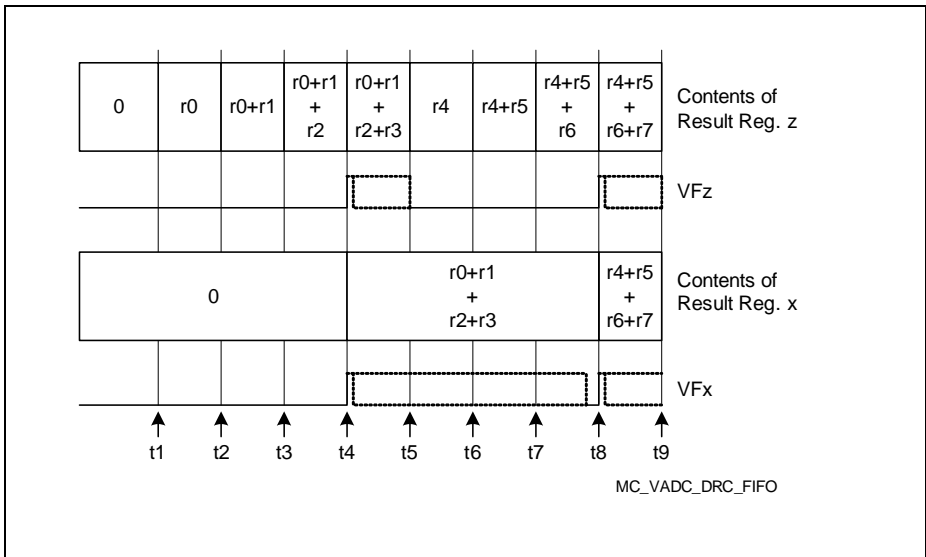


Figure 19-19 Standard Data Reduction Filter with FIFO Enabled

Versatile Analog-to-Digital Converter (VADC)

Finite Impulse Response Filter Mode (FIR)

The FIR filter (see [Figure 19-20](#)) provides 2 result buffers for intermediate results (RB1, RB2) and 3 configurable tap coefficients (a, b, c).

The conversion result and the intermediate result buffer values are added weighted with their respective coefficients to form the final value for the result register. Several predefined sets of coefficients can be selected via bitfield DRCTR (coding listed in [Table 19-4](#)) in registers **G0RESy (y = 0 - 15)** and **GLOBRES**. These coefficients lead to a gain of 3 or 4 to the ADC result producing a 14-bit value. The valid flag (VF) is activated for each sample after activation, i.e. for each sample generates a valid result.

Note: Conversions for other result registers can be inserted between conversions to be filtered.

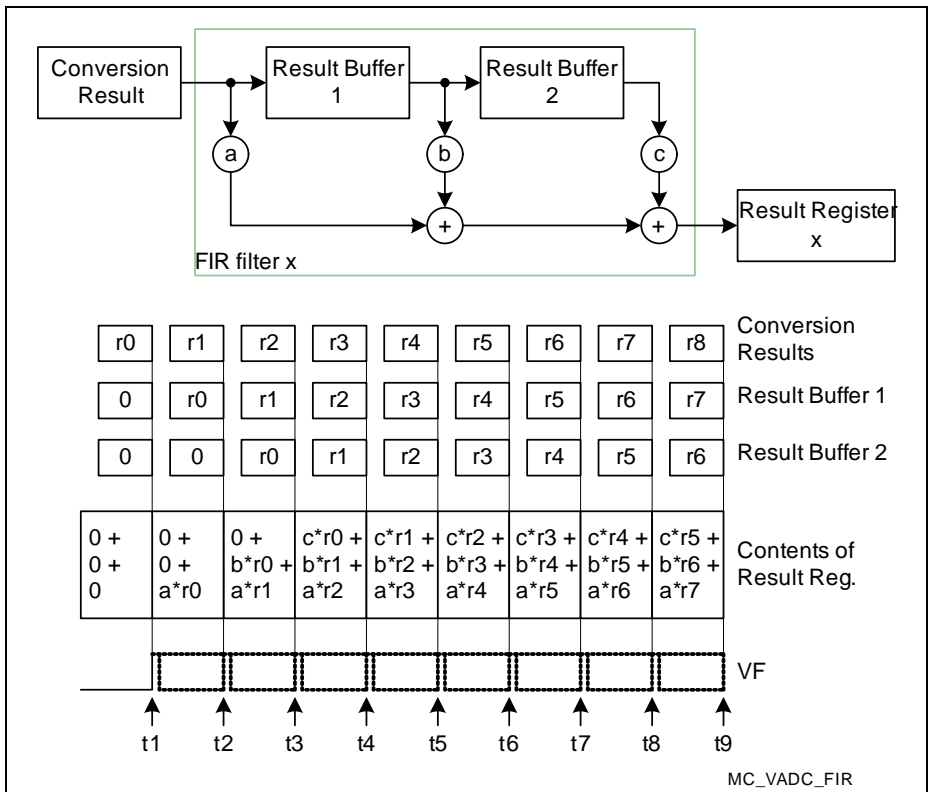


Figure 19-20 FIR Filter Structure

Note: The filter registers are cleared while bitfield DMM ≠ 01_B.

Versatile Analog-to-Digital Converter (VADC)

Infinite Impulse Response Filter Mode (IIR)

The IIR filter (see [Figure 19-21](#)) provides a result buffer (RB) and 2 configurable coefficients (a, b). It represents a first order low-pass filter.

The conversion result, weighted with the respective coefficient, and a fraction of the previous result are added to form the final value for the result register. Several predefined sets of coefficients can be selected via bitfield DRCTR (coding listed in [Table 19-4](#)) in registers **G0RESy (y = 0 - 15)** and **GLOBRES**. These coefficients lead to a gain of 4 to the ADC result producing a 14-bit value. The valid flag (VF) is activated for each sample after activation, i.e. for each sample generates a valid result.

Note: Conversions for other result registers can be inserted between conversions to be filtered.

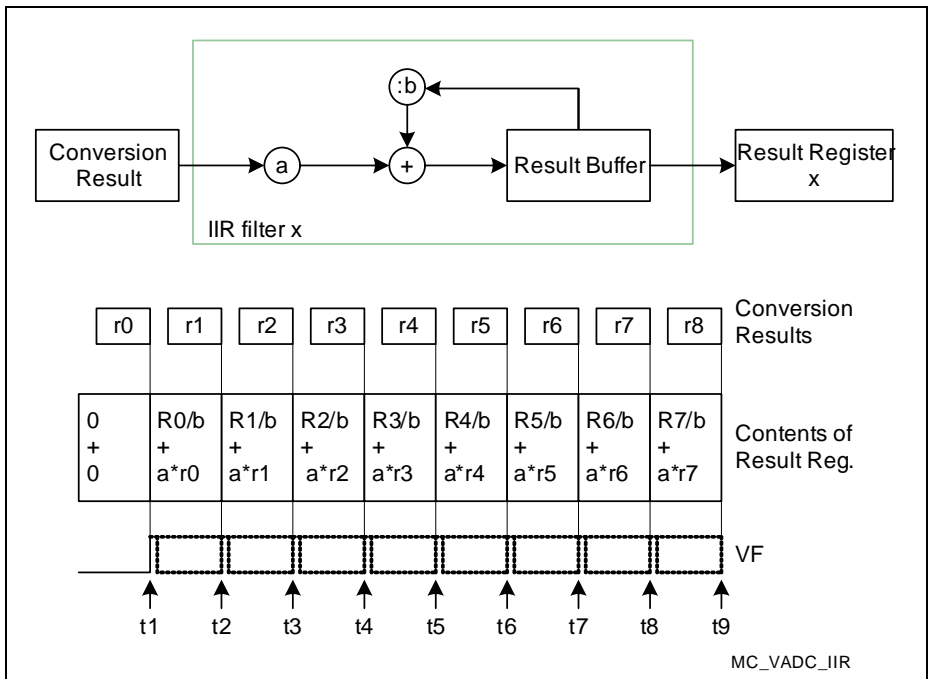


Figure 19-21 IIR Filter Structure

Note: The filter registers are cleared while bitfield DMM ≠ 01_B.

Difference Mode

Subtracting the contents of result register 0 from the actual result puts the results of the respective channel in relation to another signal. No software action is required.

The reference channel must store its result(s) into result register 0. The reference value can be determined once and then be used for a series of conversions, or it can be converted before each related conversion.

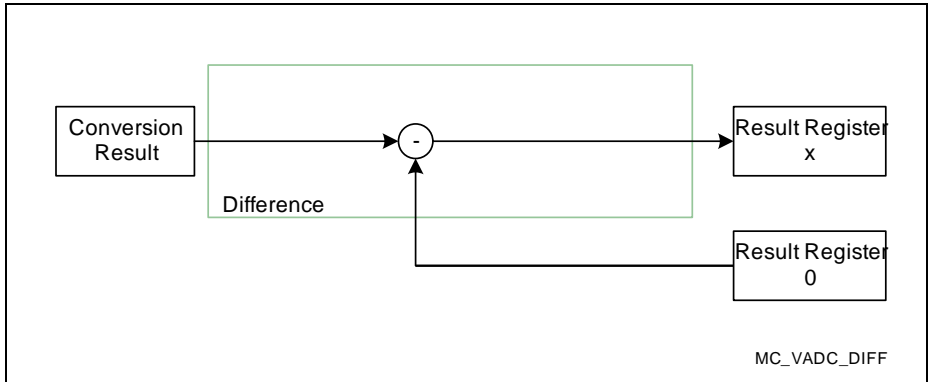


Figure 19-22 Result Difference

19.9 Synchronization of Conversions

The conversions of an ADC kernel can be scheduled either self-timed according to the kernel's configuration or triggered by external (outside the ADC) signals:

Synchronized Conversions for Parallel Sampling support parallel conversion of channels within a synchronization group¹⁾. This optimizes e.g. the control of electrical drives.

Equidistant Sampling supports conversions in a fixed raster with minimum jitter. This optimizes e.g. filter algorithms or audio applications.

19.9.1 Synchronized Conversions for Parallel Sampling

Several independent ADC kernels¹⁾ implemented in the XMC4300 can be synchronized for simultaneous measurements of analog input channels. While no parallel conversion is requested, the kernels can work independently.

The synchronization mechanism for parallel conversions ensures that the sample phases of the related channels start simultaneously. Synchronized kernels convert the same channel that is requested by the master. Different values for the resolution and the sample phase length of each kernel for a parallel conversion are supported.

A parallel conversion can be requested individually for each input channel (one or more). In the example shown in **Figure 19-23**, input channels CH3 of the ADC kernels 0 and 1 are converted synchronously, whereas other input channels do not lead to parallel conversions.

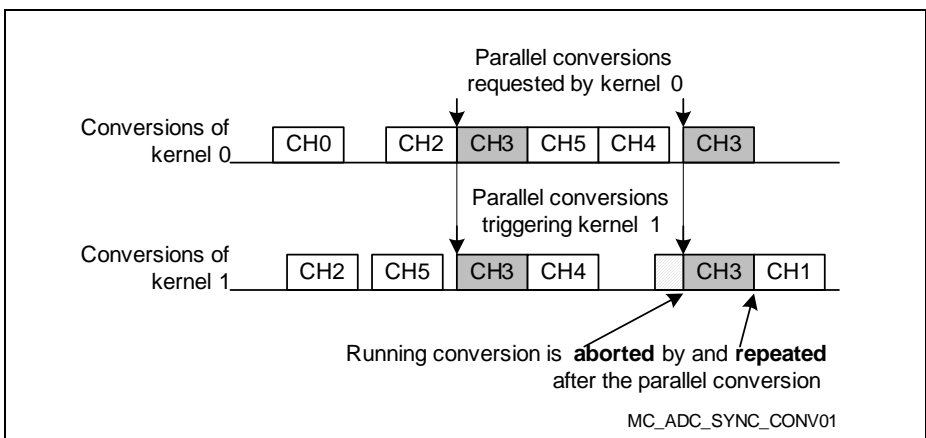


Figure 19-23 Parallel Conversions

1) For a summary, please refer to **"Synchronization Groups in the XMC4300"** on **Page 19-133**.

Versatile Analog-to-Digital Converter (VADC)

One kernel operates as synchronization master, the other kernel(s) operate(s) as synchronization slave. Each kernel can play either role. The arbiters of all involved kernels must run synchronously. This is achieved by switching off all involved kernels before the initialization and switching on the master kernel at the end of the initialization sequence.

Master and slave kernels form a “conversion group” to control parallel sampling:

- The arbiters must run permanently (bits **GxARBCFG (x = 0 - 1).ARBM** = 0). Initialize the slave before the master to have the arbiters run synchronously. Set the master’s **GxARBCFG.ANONC** at the end of the initialization.
- **The synchronization master** controls the slave(s) by providing the control information **GxARBCFG (x = 0 - 1).ANONS** (see **Figure 19-24**) and the requested channel number.
 - Bitfield **GxSYNCTR (x = 0 - 1).STSEL = 00_B** selects the master’s ANON information as the source of the ANON information for all kernels of the synchronization group.¹⁾
 - The ready signals indicate when a slave kernel is ready to start the sample phase of a parallel conversion. Bit **GxSYNCTR (x = 0 - 1).EVALRy = 1** enables the control by the ready signal (in the example kernel 1 is the slave, so **EVALR1 = 1**).
 - The master requests a synchronized conversion of a certain channel (**SYNC = 1** in the corresponding channel control register **GxCHCTRy**), which is also requested in the connected slave ADC kernel(s).
 - Wait-for-read mode is supported for the master.
- **The synchronization slave** reacts to incoming synchronized conversion requests from the master. While no synchronized conversions are requested, the slave kernel can execute “local” conversions.
 - Bitfield **GxSYNCTR (x = 0 - 1).STSEL = 01_B/10_B/11_B** selects the master’s ANON information as the source of the ANON information for all kernels of the synchronization group¹⁾ (in the example kernel 0 is the master, so **STSEL = 01_B**).
 - The ready signals indicate when the master kernel and the other slave kernels are ready to start the sample phase of a parallel conversion. Bit **GxSYNCTR (x = 0 - 1).EVALRy = 1** enables the control by the ready signal (in the example kernel 0 is the master, so **EVALR1 = 1**).
 - The slave timing must be configured according to the master timing (**ARBRND** in register **GxARBCFG (x = 0 - 1)**) to enable parallel conversions.
 - A parallel conversion request is always handled with highest priority and cancel-inject-repeat mode.
 - Wait-for-read mode is ignored in the slave. Previous results may be overwritten, in particular, if the same result register is used by other conversions.

1) **STSEL = 00_B** selects the own ANON information. The other control inputs (**STSEL = 01_B/10_B/11_B**) are connected to the other kernels of a synchronization group in ascending order (see also **Table 19-11 “Synchronization Groups in the XMC4300” on Page 19-133**).

Versatile Analog-to-Digital Converter (VADC)

- Once started, a parallel conversion cannot be aborted.

Note: Synchronized conversions request the same channel number, defined by the master. Using the alias feature (see [Section 19.7.3](#)), analog signals from different input channels can be converted. This is advantageous if e.g. CH0 is used as alternate reference.

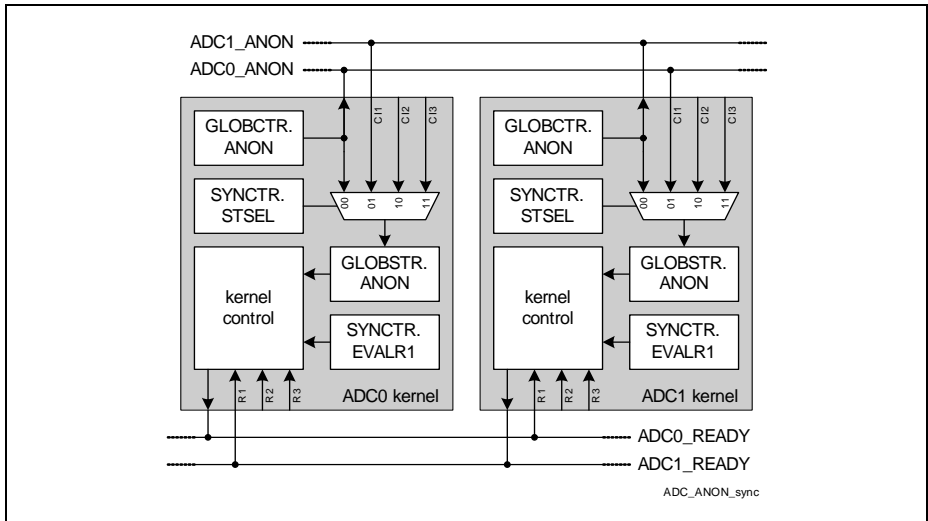


Figure 19-24 Synchronization via ANON and Ready Signals

19.9.2 Equidistant Sampling

To optimize the input data e.g. for filter or audio applications, conversions can be executed in a fixed timing raster. Conversions for equidistant sampling are triggered by an external signal (e.g. a timer). To generate the trigger signal synchronous to the arbiter, the ADC provides an output signal (ARBCNT) that is activated once per arbitration round and serves as timing base for the trigger timer. In this case, the arbiter must run permanently (**GxARBCFG (x = 0 - 1).ARBM = 0**). If the timer has an independent time base, the arbiter can be stopped while no requests are pending. The preface time (see **Figure 19-25**) must be longer than one arbitration round and the highest possible conversion time.

Select timer mode (TMEN = 1 in register **GxQCTRL0 (x = 0 - 1)** or **GxASCTRL (x = 0 - 1)**) for the intended source of equidistant conversions. In timer mode, a request of this source is triggered and arbitrated, but only started when the trigger signal is removed (see **Figure 19-25**) and the converter is idle.

To ensure that the converter is idle and the start of conversion can be controlled by the trigger signal, the equidistant conversion requests must receive highest priority. The preface time between request trigger and conversion start must be long enough for a currently active conversion to finish.

The frequency of signal REQTRx defines the sampling rate and its high time defines the preface time interval where the corresponding request source takes part in the arbitration.

Depending on the used request source, equidistant sampling is also supported for a sequence of channels. It is also possible to do equidistant sampling for more than one request source in parallel if the preface times and the equidistant conversions do not overlap.

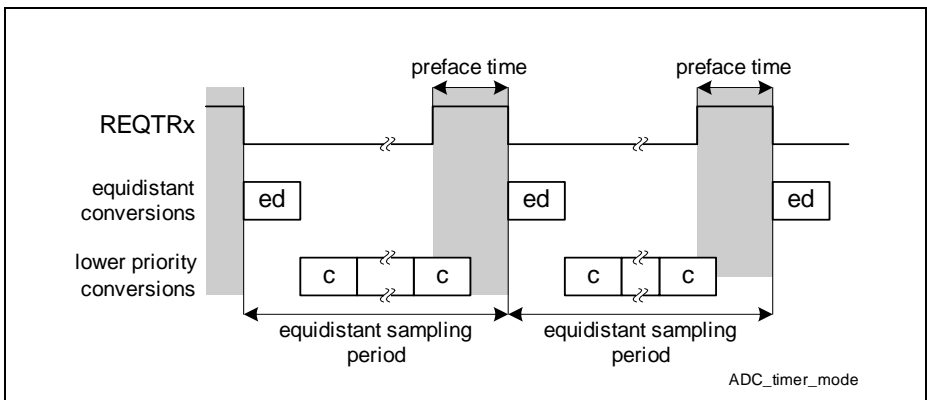


Figure 19-25 Timer Mode for Equidistant Sampling

19.10 Safety Features

Several test features can be enabled to verify the validity of the analog input signals of an application. These test features aim at different sections of the signal flow:

- **Broken Wire Detection** validates the connection from the sensor to the input pin,
- **Multiplexer Diagnostics** validates the operation of the internal analog input multiplexer,
- **Converter Diagnostics** validates the operation of the Analog/Digital converter itself.

19.10.1 Broken Wire Detection

To test the proper connection of an external analog sensor to its input pin, the converter's capacitor can be precharged to a selectable value before the regular sample phase. If the connection to the sensor is interrupted the subsequent conversion value will rather represent the precharged value than the expected sensor result. By using a precharge voltage outside the expected result range (broken wire detection preferably uses V_{AGND} and/or V_{AREF}) a valid measurement (sensor connected) can be distinguished from a failure (sensor detached).

While broken wire detection is disabled, the converter's capacitor is precharged to $V_{AREF}/2$.

Note: The duration of the complete conversion is increased by the preparation phase (same as the sample phase) if the broken wire detection is enabled. This influences the timing of conversion sequences.

Broken wire detection can be enabled for each channel separately by bitfield BW DEN in the corresponding channel control register (**G0CHCTRY (y = 0 - 7)**). This bitfield also selects the level for the preparation phase.

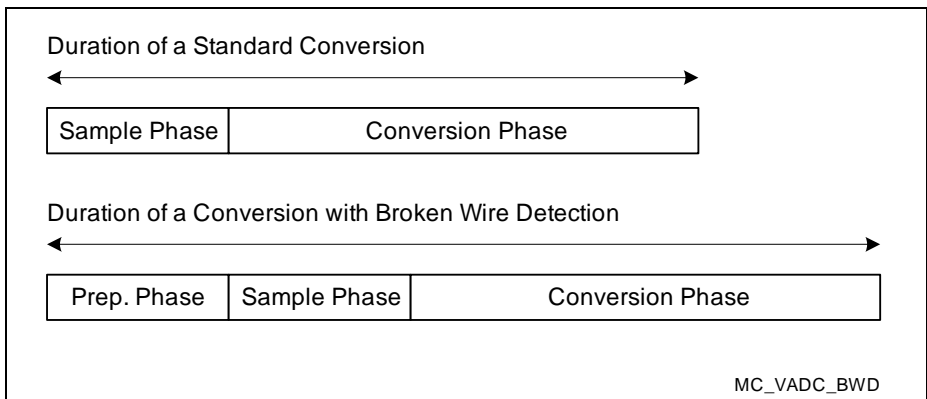


Figure 19-26 Broken Wire Detection

19.10.2 Signal Path Test Modes

Additional test structures can be activated to test the signal path from the sensor to the input pin and the internal signal path from the input pin through the multiplexer to the converter. These test structures apply additional loads to the signal path (see summary in [Figure 19-27](#)).

Multiplexer Diagnostics

To test the proper operation of the internal analog input multiplexer, additional pull-up and/or pull-down devices can be connected to a channel. In combination with a known external input signal this test function shows if the multiplexer connects any pin to the converter input and if this is the correct pin. These pull-up/pull-down devices are controlled via the port logic.

Pull-Down Diagnostics

One single input channel provides a further strong pull-down (R_{PDD}) that can be activated to verify the external connection to a sensor.

Converter Diagnostics

To test the proper operation of the converter itself, several signals can be connected to the converter input. The test signals can be connected to the converter input either instead of the standard input signal or in parallel to the standard input signal.

The test signal can be selected from four different signals as shown in [Figure 19-27](#).

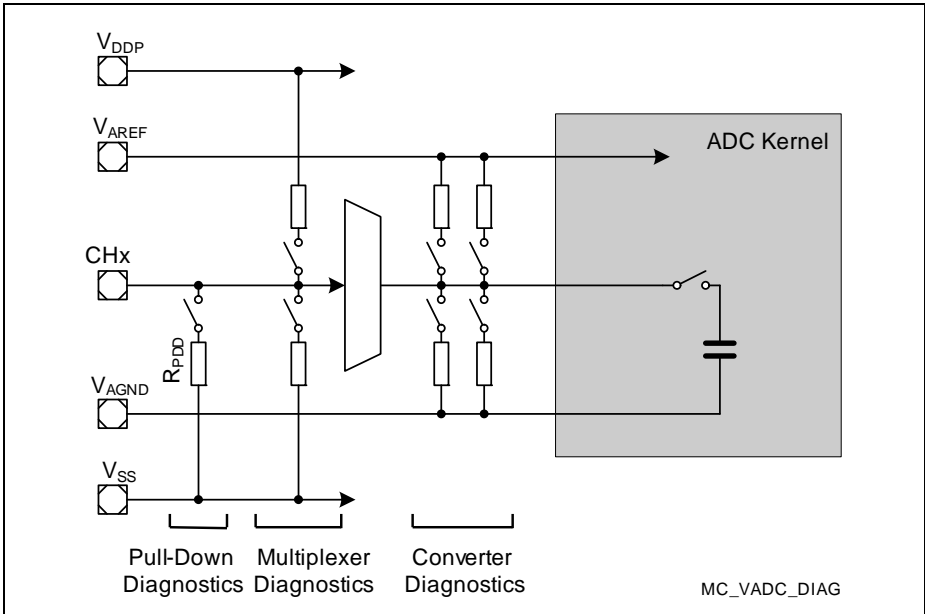


Figure 19-27 Signal Path Test

19.10.3 Configuration of Test Functions

The pull-up and pull-down devices for the test functions can be enabled individually under software control. Various test levels can be applied controlling the devices in an adequate way. Because these test functions interfere with the normal operation of the A/D Converters, they are controlled by a separate register set or by port registers.

Not all test options are available for each channel. Selecting an unavailable function has no effect.

19.11 External Multiplexer Control

The number of analog input channels can be increased by connecting external analog multiplexers to an input channel. The ADC can be configured to control these external multiplexers automatically.

For each available EMUX interface (see register **EMUXSEL**) one channel can be selected for this operating mode. The ADC supports 1-out-of-8 multiplexers with several control options:

- **Sequence mode** automatically converts all configured external channels when the selected channel is encountered. In the example in **Figure 19-28** the following conversions are done: --4-32-31-30-2-1-0--4-32-31-30-2-1-0--...
- **Single-step mode** converts one external channel of the configured sequence when the selected channel is encountered. In the example in **Figure 19-28** the following conversions are done: --4-32-2-1-0--4-31-2-1-0--4-30-2-1-0--4-32-... (Single-step mode works best with one channel)
- **Steady mode** converts the configured external channel when the selected channel is encountered. In the example in **Figure 19-28** the following conversions are done: --4-32-2-1-0--4-32-2-1-0--4-32-2-1-0--...

*Note: The example in **Figure 19-28** has an external multiplexer connected to channel CH3. The start selection value EMUXSET is assumed as 2.*

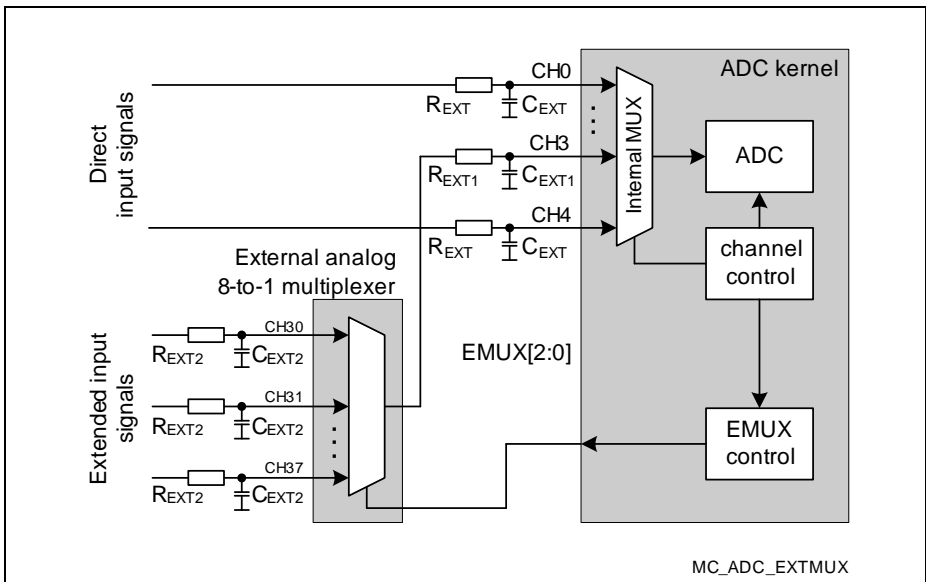


Figure 19-28 External Analog Multiplexer Example

Versatile Analog-to-Digital Converter (VADC)

Bitfield EMUXACT determines the control information sent to the external multiplexer. In single-step mode, EMUXACT is updated after each conversion of an enabled channel. If EMUXACT = 000_B it is reloaded from bitfield EMUXSET, otherwise it is decremented by 1.

Additional external channels may have different properties due to the modified signal path. Local filters may be used at the additional inputs (R_{EXT2} - C_{EXT2} on CH3x in [Figure 19-28](#)). For applications where the external multiplexer is located far from the ADC analog input, it is recommended to add an RC filter directly at the analog input of the ADC (R_{EXT1} - C_{EXT1} on CH3 in [Figure 19-28](#)).

Note: Each RC filter limits the bandwidth of the analog input signal.

Conversions for external channels, therefore, use the alternate conversion mode setting CME. This automatically selects a different conversion mode if required.

Switching the external multiplexer usually requires an additional settling time for the input signal. Therefore, the alternate sample time setting STCE is applied each time the external channel is changed. This automatically fulfills the different sampling time requirements in this case.

In each group an arbitrary channel can be assigned to external multiplexer control (register **GxEMUXCTR** ($x = 0 - 1$)). Each available port interface selects the group whose control lines are output (register **EMUXSEL**).

Control Signals

The external channel number that controls the external multiplexer can be output in standard binary format or Gray-coded. Gray code avoids intermediate multiplexer switching when selecting a sequence of channels, because only one bit changes at a time. [Table 19-5](#) indicates the resulting codes.

Table 19-5 EMUX Control Signal Coding

Channel	0	1	2	3	4	5	6	7
Binary	000 _B	001 _B	010 _B	011 _B	100 _B	101 _B	110 _B	111 _B
Gray	000	001	011	010	110	111	101	100

Operation Without External Multiplexer

If no external multiplexers are used in an application, the reset values of the control registers provide the appropriate setup.

EMUXMODE = 00_B disables the automatic EMUX control.

Since the control output signals are alternate port output signals, they are only visible at the respective pins if explicitly selected.

19.12 Service Request Generation

Each A/D Converter can activate up to 4 group-specific service request output signals and up to 4 shared service request output signals to issue an interrupt or to trigger a DMA channel. Two common service request groups are available, see [Table 19-10 “General Converter Configuration in the XMC4300” on Page 19-132](#).

Several events can be assigned to each service request output. Service requests can be generated by three types of events:

- **Request source events:** indicate that a request source completed the requested conversion sequence and the application software can initiate further actions.
For a scan source (group or background), the event is generated when the complete defined set of channels (pending bits) has been converted.
For a group queue source, the event is generated according to the programming, i.e. when a channel with enabled source interrupt has been converted or when an invalid entry is encountered.
- **Channel events:** indicate that a conversion is finished. Optionally, channel events can be restricted to result values within a programmable value range. This offloads the CPU/DMA from background tasks, i.e. a service request is only activated if the specified conversion result range is met or exceeded.
- **Result events:** indicate a new valid result in a result register. Usually, this triggers a read action by the CPU (or DMA). Optionally, result events can be generated only at a reduced rate if data reduction is active.
For example, a single DMA channel can read the results for a complete auto-scan sequence, if all channels of the sequence target the same result register and the transfers are triggered by result events.

Each ADC event is indicated by a dedicated flag that can be cleared by software. If a service request is enabled for a certain event, the service request is generated for each event, independent of the status of the corresponding event indication flag. This ensures efficient DMA handling of ADC events (the ADC event can generate a service request without the need to clear the indication flag).

Event flag registers indicate all types of events that occur during the ADC's operation. Software can set each flag by writing a 1 to the respective position in register GxCEFLAG/GxRFLAG to trigger an event. Software can clear each flag by writing a 1 to the respective position in register GxCEFCLR/GxREFCLR. If enabled, service requests are generated for each occurrence of an event, even if the associated flag remains set.

Node Pointer Registers

Requests from each event source can be directed to a set of service request nodes via associated node pointers. Requests from several sources can be directed to the same node; in this case, they are ORed to the service request output signal.

Software Service Request Activation

Each service request can be activated via software by setting the corresponding bit in register **GxSRACT (x = 0 - 1)**. This can be used for evaluation and testing purposes.

Note: For shared service request lines see common groups in [Table 19-10](#).

Versatile Analog-to-Digital Converter (VADC)

19.13 Registers

The Versatile ADC is built from a series of converter blocks that are controlled in an identical way. This makes programming versatile and scalable. The corresponding registers, therefore, have an individual offset assigned (see [Table 19-7](#)). The exact register location is obtained by adding the respective register offset to the base address (see [Table 19-6](#)) of the corresponding group.

Due to the regular group structure, several registers appear within each group. Other registers are provided for each channel. This is indicated in the register overview table by placeholders:

- $X###_H$ means: $x \times 0400_H + 0###_H$, for $x = 0 - 1$
- $###Y_H$ means: $###0_H + y \times 0004_H$, for $y = 0 - N$ (depends on register type)

Table 19-6 Registers Address Space

Module	Base Address	End Address	Note
VADC	4000 4000 _H	4000 7FFF _H	

Table 19-7 Registers Overview

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Page Num.
			Read	Write	
ID	Module Identification Register	0008 _H	U, PV	BE	19-61
CLC	Clock Control Register	0000 _H	U, PV	PV	19-63
OCS	OCDS Control and Status Register	0028 _H	U, PV	PV	19-64
GLOBCFG	Global Configuration Register	0080 _H	U, PV	U, PV	19-66
GxARBCFG	Arbitration Configuration Register	X480 _H	U, PV	U, PV	19-68
GxARBPR	Arbitration Priority Register	X484 _H	U, PV	U, PV	19-70
GxCHASS	Channel Assignment Register, Group x	X488 _H	U, PV	U, PV	19-67
GxQCTRL0	Queue 0 Source Control Register, Group x	X500 _H	U, PV	U, PV	19-72
GxQMR0	Queue 0 Mode Register, Group x	X504 _H	U, PV	U, PV	19-74
GxQSR0	Queue 0 Status Register, Group x	X508 _H	U, PV	U, PV	19-76
GxQINR0	Queue 0 Input Register, Group x	X510 _H	U, PV	U, PV	19-78
GxQ0R0	Queue 0 Register 0, Group x	X50C _H	U, PV	U, PV	19-80
GxQBUR0	Queue 0 Backup Register, Group x	X510 _H	U, PV	U, PV	19-82

Versatile Analog-to-Digital Converter (VADC)
Table 19-7 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Page Num.
			Read	Write	
GxASCTRL	Autoscan Source Control Register, Group x	X520 _H	U, PV	U, PV	19-84
GxASMR	Autoscan Source Mode Register, Group x	X524 _H	U, PV	U, PV	19-86
GxASSEL	Autoscan Source Channel Select Register, Group x	X528 _H	U, PV	U, PV	19-88
GxASPND	Autoscan Source Pending Register, Group x	X52C _H	U, PV	U, PV	19-89
BRCTRL	Background Request Source Control Register	0200 _H	U, PV	U, PV	19-90
BRSMR	Background Request Source Mode Register	0204 _H	U, PV	U, PV	19-92
BRSELx	Background Request Source Channel Select Register, Group x	018Y _H	U, PV	U, PV	19-94
BRSPNDx	Background Request Source Channel Pending Register, Group x	01CY _H	U, PV	U, PV	19-95
GxCHCTry	Channel x Control Register	X60Y _H	U, PV	U, PV	19-96
GxICLASS0	Input Class Register 0, Group x	X4A0 _H	U, PV	U, PV	19-98
GxICLASS1	Input Class Register 1, Group x	X4A4 _H	U, PV	U, PV	19-98
GLOBICLASS0	Input Class Register 0, Global	00A0 _H	U, PV	U, PV	19-98
GLOBICLASS1	Input Class Register 1, Global	00A4 _H	U, PV	U, PV	19-98
GxALIAS	Alias Register	X4B0 _H	U, PV	U, PV	19-109
GxBOUND	Boundary Select Register, Group x	X4B8 _H	U, PV	U, PV	19-110
GLOBBOUND	Global Boundary Select Register	00B8 _H	U, PV	U, PV	19-110
GxBFL	Boundary Flag Register, Group x	X4C8 _H	U, PV	U, PV	19-111
GxBFLS	Boundary Flag Software Register, Group x	X4CC _H	U, SV	U, SV P	19-112
GxBFLC	Boundary Flag Control Register, Group x	X4D0 _H	U, SV	U, SV P	19-113

Versatile Analog-to-Digital Converter (VADC)
Table 19-7 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Page Num.
			Read	Write	
GxBFLNP	Boundary Flag Node Pointer Register, Group x	X4D4 _H	U, SV	U, SV P	19-114
GxRCRy	Group x Result Control Register y	X68Y _H	U, PV	U, PV	19-101
GxRESy	Group x Result Register y	X70Y _H	U, PV	U, PV	19-103
GxRESyDy	Group x Result Register y (debug view)	X78Y _H	U, PV	U, PV	19-104
GLOBRCR	Global Result Control Register	0280 _H	U, PV	U, PV	19-106
GLOBRES	Global Result Register	0300 _H	U, PV	U, PV	19-107
GLOBRESD	Global Result Register (debug view)	0380 _H	U, PV	U, PV	19-107
GxVFR	Valid Flag Register, Group x	X5F8 _H	U, PV	U, PV	19-108
GxSYNCTR	Synchronization Control Register	X4C0 _H	U, PV	U, PV	19-115
GLOBTF	Global Test Functions Register	0160 _H	U, PV	U, PV	19-116
GxEMUXCTR	External Multiplexer Control Register, Group x	X5F0 _H	U, PV	U, PV	19-117
EMUXSEL	External Multiplexer Select Register	03F0 _H	U, PV	U, PV	19-120
GxSEFLAG	Source Event Flag Register, Group x	X588 _H	U, PV	U, PV	19-120
GxCEFLAG	Channel Event Flag Register, Group x	X580 _H	U, PV	U, PV	19-121
GxREFLAG	Result Event Flag Register, Group x	X584 _H	U, PV	U, PV	19-122
GxSEFCLR	Source Event Flag Clear Register, Group x	X598 _H	U, PV	U, PV	19-122

Versatile Analog-to-Digital Converter (VADC)

Table 19-7 Registers Overview (cont'd)

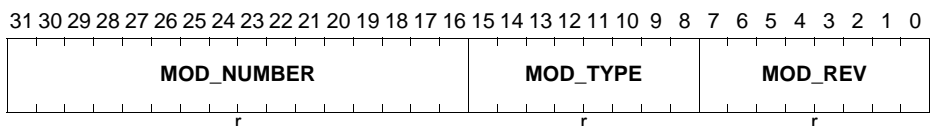
Register Short Name	Register Long Name	Offset Addr.	Access Mode		Page Num.
			Read	Write	
GxCEFCLR	Channel Event Flag Clear Register, Group x	X590 _H	U, PV	U, PV	19-123
GxREFCLR	Result Event Flag Clear Register, Group x	X594 _H	U, PV	U, PV	19-124
GLOBEFLAG	Global Event Flag Register	00E0 _H	U, PV	U, PV	19-124
GxSEVNP	Source Event Node Pointer Register, Group x	X5C0 _H	U, PV	U, PV	19-125
GxCEVNP0	Channel Event Node Pointer Register 0, Group x	X5A0 _H	U, PV	U, PV	19-126
GxREVNP0	Result Event Node Pointer Register 0, Group x	X5B0 _H	U, PV	U, PV	19-127
GxREVNP1	Result Event Node Pointer Register 1, Group x	X5B4 _H	U, PV	U, PV	19-128
GLOBEVNP	Global Event Node Pointer Register	0140 _H	U, PV	U, PV	19-129
GxSRACT	Service Request Software Activation Trigger, Group x	X5C8 _H	U, PV	U, PV	19-131

19.13.1 Module Identification

The module identification register indicates the version of the ADC module that is used in the XMC4300.

ID

Module Identification Register (0008_H) **Reset Value: 00C5 C0XX_H**



Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
MOD_REV	[7:0]	r	Module Revision Indicates the revision number of the implementation. This information depends on the design step.
MOD_TYPE	[15:8]	r	Module Type This internal marker is fixed to C0 _H .
MOD_NUMBER	[31:16]	r	Module Number Indicates the module identification number (00C5 _H = SARADC).

Versatile Analog-to-Digital Converter (VADC)

19.13.2 System Registers

A set of standardized registers provides general access to the module and controls basic system functions.

The Clock Control Register **CLC** allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. Register **CLC** controls the module clock signal and the reactivity to the sleep mode signal.

CLC

Clock Control Register

(0000_H)

Reset Value: 0000 0003_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	E DIS	0	DIS S	DIS R
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
DISR	0	rw	Module Disable Request Bit Used for enable/disable control of the module. Also the analog section is disabled by clearing ANONS. 0 _B On request: enable the module clock 1 _B Off request: stop the module clock
DISS	1	r	Module Disable Status Bit 0 _B Module clock is enabled 1 _B Off: module is not clocked
0	2	r	Reserved, write 0, read as 0
EDIS	3	rw	Sleep Mode Enable Control Used to control module's reaction to sleep mode. 0 _B Sleep mode request is enabled and functional 1 _B Module disregards the sleep mode control signal
0	[31:4]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The OCDS control and status register OCS controls the module's behavior in suspend mode (used for debugging) and includes the module-related control bits for the OCDS Trigger Bus (OTGB).

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

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

If OCDS is being disabled, the OCS register value will not change.

When OCDS is disabled the OCS suspend control is ineffective.

Write access is 32 bit wide only and requires Supervisor Mode.

OCS

OCDS Control and Status Register (0028_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	SUS STA	SUS _P	SUS				0	0	0	0	0	0	0	0
r	r	rh	w	rw				r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	TG _P	TGB	TGS	
r	r	r	r	r	r	r	r	r	r	r	r	w	rw	rw	

Field	Bits	Type	Description
TGS	[1:0]	rw	Trigger Set for OTGB0/1 00 _B No Trigger Set output 01 _B Trigger Set 1: TS16_SSIG, input sample signals 10 _B Reserved 11 _B Reserved
TGB	2	rw	OTGB0/1 Bus Select 0 _B Trigger Set is output on OTGB0 1 _B Trigger Set is output on OTGB1
TG_P	3	w	TGS, TGB Write Protection TGS and TGB are only written when TG_P is 1, otherwise unchanged. Read as 0.
0	[23:4]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
SUS	[27:24]	rw	OCDS Suspend Control Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0000 _B Will not suspend 0001 _B Hard suspend: Clock is switched off immediately. 0010 _B Soft suspend mode 0: Stop conversions after the currently running one is completed and its result has been stored. No change for the arbiter. 0011 _B Soft suspend mode 1: Stop conversions after the currently running one is completed and its result has been stored. Stop arbiter after the current arbitration round. others: Reserved
SUS_P	28	w	SUS Write Protection SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	Suspend State 0 _B Module is not (yet) suspended 1 _B Module is suspended
0	[31:30]	r	Reserved, write 0, read as 0

Table 19-8 TS16_SSIG Trigger Set VADC

Bits	Name	Description
[3:0]	GxSAMPLE	Input signal sample phase of converter group x (x = 3-0)
[15:4]	0	Reserved

Note: The SAMPLE signals can be used as gate/trigger inputs for the adjacent groups. These outputs are enabled when bitfield TGS = 01_B and bit TGB = 0_B.

Versatile Analog-to-Digital Converter (VADC)

19.13.3 General Registers

The global configuration register provides global control and configuration options that are valid for all converters of the cluster.

Version with 2 groups.

GLOBCFG

Global Configuration Register (0080_H) **Reset Value: 0000 000F_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
SU CAL	0	0	0	0	0	0	0	0	0	0	0	0	0	DP CAL	DP CAL	
														1	0	
w	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
DIV WC	0	0	0	0	0	DIVD		DC MSB	0	0	DIVA					
w	r	r	r	r	r											

Field	Bits	Type	Description
DIVA	[4:0]	rw	Divider Factor for the Analog Internal Clock Defines the frequency of the basic converter clock f_{ADCI} (base clock for conversion and sample phase). 00 _H $f_{ADCI} = f_{ADC} / 2$ 01 _H $f_{ADCI} = f_{ADC} / 2$ 02 _H $f_{ADCI} = f_{ADC} / 3$... 1F _H $f_{ADCI} = f_{ADC} / 32$
0	[6:5]	r	Reserved, write 0, read as 0
DCMSB	7	rw	Double Clock for the MSB Conversion Selects an additional clock cycle for the conversion step of the MSB. ¹⁾ 0 _B 1 clock cycles for the MSB (standard) 1 _B Reserved
DIVD	[9:8]	rw	Divider Factor for the Arbiter Clock Defines the frequency of the arbiter clock f_{ADCD} . 00 _B $f_{ADCD} = f_{ADC}$ 01 _B $f_{ADCD} = f_{ADC} / 2$ 10 _B $f_{ADCD} = f_{ADC} / 3$ 11 _B $f_{ADCD} = f_{ADC} / 4$

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
0	[14:10]	r	Reserved, write 0, read as 0
DIVWC	15	w	Write Control for Divider Parameters 0_B No write access to divider parameters 1_B Bitfields DIVA, DCMSB, DIVD can be written
DPCALx (x = 0 - 1)	x+16	rw	Disable Post-Calibration 0_B Automatic post-calibration after each conversion of group x 1_B No post-calibration
0	[30:18]	r	Reserved, write 0, read as 0
SUCAL	31	w	Start-Up Calibration The 0-1 transition of bit SUCAL initiates the start-up calibration phase of all calibrated analog converters. 0_B No action 1_B Initiate the start-up calibration phase (indication in bit GxARBCFG.CAL)

1) Please also refer to section **"Conversion Timing"** on Page 19-26.

GxCHASS (x = 0 - 1)

Channel Assignment Register, Group x

$$(x * 0400_H + 0488_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	ASS CH 7	ASS CH 6	ASS CH 5	ASS CH 4	ASS CH 3	ASS CH 2	ASS CH 1	ASS CH 0
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ASSCHy (y = 0 - 7)	y	rw	Assignment for Channel y 0_B Channel y can be a background channel converted with lowest priority 1_B Channel y is a priority channel within group x

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
0	[31:8]	r	Reserved, write 0, read as 0

19.13.4 Arbitration and Source Registers

The Arbitration Configuration Register selects the timing and the behavior of the arbiter.

GxARBCFG (x = 0 - 1)

Arbitration Configuration Register, Group x

$$(x * 0400_H + 0480_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SAMPLE	BU SY	0	CAL	0	0	0	0	0	0	0	0	0	0	ANONS	
rh	rh	r	rh	r	r	r	r	r	r	r	r	r	r	r	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	ARB M	0	ARBRND		0	0	ANONC	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
ANONC	[1:0]	rw	Analog Converter Control Defines the value of bitfield ANONS in a stand-alone converter or a converter in master mode. Coding see ANONS or Section 19.4 .
0	[3:2]	r	Reserved, write 0, read as 0
ARBRND	[5:4]	rw	Arbitration Round Length Defines the number of arbitration slots per arb. round (arbitration round length = t_{ARB}). ¹⁾ 00 _B 4 arbitration slots per round ($t_{ARB} = 4 / f_{ADCD}$) 01 _B 8 arbitration slots per round ($t_{ARB} = 8 / f_{ADCD}$) 10 _B 16 arbitration slots per round ($t_{ARB} = 16 / f_{ADCD}$) 11 _B 20 arbitration slots per round ($t_{ARB} = 20 / f_{ADCD}$)
0	6	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
ARBM	7	rw	<p>Arbitration Mode</p> <p>0_B The arbiter runs permanently. This setting is required for a synchronization slave (see Section 19.9.1) and for equidistant sampling using the signal ARBCNT (see Section 19.9.2).</p> <p>1_B The arbiter only runs if at least one conversion request of an enabled request source is pending. This setting ensures a reproducible latency from an incoming request to the conversion start, if the converter is idle. Synchronized conversions are not supported.</p>
0	[15:8]	r	Reserved, write 0, read as 0
ANONS	[17:16]	rh	<p>Analog Converter Control Status</p> <p>Defined by bitfield ANONC in a stand-alone kernel or a kernel in master mode. In slave mode, this bitfield is defined by bitfield ANONC of the respective master kernel. See also Section 19.4.</p> <p>00_B Analog converter off 01_B Reserved 10_B Reserved 11_B Normal operation (permanently on)</p>
0	[27:18]	r	Reserved, write 0, read as 0
CAL	28	rh	<p>Start-Up Calibration Active Indication</p> <p>Indicates the start-up calibration phase of the corresponding analog converter.</p> <p>0_B Completed or not yet started 1_B Start-up calibration phase is active</p> <p><i>Note: Start conversions only after the start-up calibration phase is complete.</i></p>
0	29	r	Reserved, write 0, read as 0
BUSY	30	rh	<p>Converter Busy Flag</p> <p>0_B Not busy 1_B Converter is busy with a conversion</p>

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
SAMPLE	31	rh	Sample Phase Flag 0 _B Converting or idle 1 _B Input signal is currently sampled

1) The default setting of 4 arbitration slots is sufficient for correct arbitration. The duration of an arbitration round can be increased if required to synchronize requests.

The Arbitration Priority Register defines the request source priority and the conversion start mode for each request source.

Note: Only change priority and conversion start mode settings of a request source while this request source is disabled, and a currently running conversion requested by this source is finished.

GxARBPR (x = 0 - 1)

Arbitration Priority Register, Group x

$$(x * 0400_H + 0484_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	AS EN2	AS EN1	AS EN0	0	0	0	0	0	0	0	0
r	r	r	r	r	rw	rw	rw	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	CSM 2	0	PRIO 2	CSM 1	0	PRIO 1	CSM 0	0	PRIO 0	0	PRIO 0	0
r	r	r	r	rw	r	rw	rw	r	rw	rw	r	rw	r	rw	r

Field	Bits	Type	Description
PRIO0, PRIO1, PRIO2	[1:0], [5:4], [9:8]	rw	Priority of Request Source x Arbitration priority of request source x (in slot x) 00 _B Lowest priority is selected. ... 11 _B Highest priority is selected.
CSM0, CSM1, CSM2	3, 7, 11	rw	Conversion Start Mode of Request Source x 0 _B Wait-for-start mode 1 _B Cancel-inject-repeat mode, i.e. this source can cancel conversion of other sources.

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
0	2, 6, 10, [23:12]	r	Reserved, write 0, read as 0
ASENy (y = 0 - 2)	24 + y	rw	<p>Arbitration Slot y Enable Enables the associated arbitration slot of an arbiter round. The request source bits are not modified by write actions to ASENy.</p> <p>0_B The corresponding arbitration slot is disabled and considered as empty. Pending conversion requests from the associated request source are disregarded.</p> <p>1_B The corresponding arbitration slot is enabled. Pending conversion requests from the associated request source are arbitrated.</p>
0	[31:27]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The control register of the queue source selects the external gate and/or trigger signals. Write control bits allow separate control of each function with a simple write access.

GxQCTRL0 (x = 0 - 1)

Queue 0 Source Control Register, Group x

$$(x * 0400_H + 0500_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TM WC	0	0	TM EN	0	0	0	0	GT WC	0	0	GT LVL			GT SEL	
w	r	r	rw	r	r	r	r	w	r	r	rh			rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XT WC	XT MODE	XT LVL			XT SEL			0	0	0	0			SRCRESREG	
w	rw	rh			rw			r	r	r	r			rw	

Field	Bits	Type	Description
SRCRESREG	[3:0]	rw	Source-specific Result Register 0000 _B Use GxCHCTRY.RESREG to select a group result register 0001 _B Store result in group result register GxRES1 ... 1111 _B Store result in group result register GxRES15
0	[7:4]	r	Reserved, write 0, read as 0
XTSEL	[11:8]	rw	External Trigger Input Selection The connected trigger input signals are listed in Table 19-13 “Digital Connections in the XMC4300” on Page 19-135 <i>Note: XTSEL = 1111_B uses the selected gate input as trigger source (ENG_T must be 0X_B).</i>
XTLVL	12	rh	External Trigger Level Current level of the selected trigger input
XTMODE	[14:13]	rw	Trigger Operating Mode 00 _B No external trigger 01 _B Trigger event upon a falling edge 10 _B Trigger event upon a rising edge 11 _B Trigger event upon any edge

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
XTWC	15	w	Write Control for Trigger Configuration 0 _B No write access to trigger configuration 1 _B Bitfields XTMODE and XTSEL can be written
GTSEL	[19:16]	rw	Gate Input Selection The connected gate input signals are listed in Table 19-13 “Digital Connections in the XMC4300” on Page 19-135
GTLVL	20	rh	Gate Input Level Current level of the selected gate input
0	[22:21]	r	Reserved, write 0, read as 0
GTWC	23	w	Write Control for Gate Configuration 0 _B No write access to gate configuration 1 _B Bitfield GTSEL can be written
0	[27:24]	r	Reserved, write 0, read as 0
TMEN	28	rw	Timer Mode Enable 0 _B No timer mode: standard gating mechanism can be used 1 _B Timer mode for equidistant sampling enabled: standard gating mechanism must be disabled
0	[30:29]	r	Reserved, write 0, read as 0
TMWC	31	w	Write Control for Timer Mode 0 _B No write access to timer mode 1 _B Bitfield TMEN can be written

Versatile Analog-to-Digital Converter (VADC)

The Queue Mode Register configures the operating mode of a queued request source.

GxQMR0 (x = 0 - 1)

Queue 0 Mode Register, Group x

(x * 0400_H + 0504_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RPT DIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	CEV	FLU SH	TR EV	CLR V	0	0	0	0	0	EN TR	ENGT	
r	r	r	r	w	w	w	w	r	r	r	r	r	rw	rw	

Field	Bits	Type	Description
ENGT	[1:0]	rw	<p>Enable Gate</p> <p>Selects the gating functionality for source 0/2.</p> <p>00_B No conversion requests are issued</p> <p>01_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register</p> <p>10_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register and REQGTx = 1</p> <p>11_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register and REQGTx = 0</p> <p><i>Note: REQGTx is the selected gating signal.</i></p>
ENTR	2	rw	<p>Enable External Trigger</p> <p>0_B External trigger disabled</p> <p>1_B The selected edge at the selected trigger input signal REQTR generates the trigger event</p>
0	[7:3]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
CLR_V	8	w	Clear Valid Bit 0_B No action 1_B The next pending valid queue entry in the sequence and the event flag EV are cleared. If there is a valid entry in the queue backup register (QBUR.V = 1), this entry is cleared, otherwise the entry in queue register 0 is cleared.
TREV	9	w	Trigger Event 0_B No action 1_B Generate a trigger event by software
FLUSH	10	w	Flush Queue 0_B No action 1_B Clear all queue entries (including backup stage) and the event flag EV. The queue contains no more valid entry.
CEV	11	w	Clear Event Flag 0_B No action 1_B Clear bit EV
0	[15:12]	r	Reserved, write 0, read as 0
RPTDIS	16	rw	Repeat Disable 0_B A cancelled conversion is repeated 1_B A cancelled conversion is discarded
0	[31:17]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The Queue Status Register indicates the current status of the queued source. The filling level and the empty information refer to the queue intermediate stages (if available) and to the queue register 0. An aborted conversion stored in the backup stage is not indicated by these bits (therefore, see QBURx.V).

GxQSR0 (x = 0 - 1)

Queue 0 Status Register, Group x

$$(x * 0400_H + 0508_H)$$

Reset Value: 0000 0020_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	EV	REQ GT	0	EMP TY	0	FILL			
r	r	r	r	r	r	r	rh	rh	r	rh	r	rh			

Field	Bits	Type	Description
FILL	[3:0]	rh	<p>Filling Level for Queue 2</p> <p>Indicates the number of valid queue entries. It is incremented each time a new entry is written to QINRx or by an enabled refill mechanism. It is decremented each time a requested conversion has been started. A new entry is ignored if the filling level has reached its maximum value.</p> <p>0000_B There is 1 (if EMPTY = 0) or no (if EMPTY = 1) valid entry in the queue 0001_B There are 2 valid entries in the queue 0010_B There are 3 valid entries in the queue ... 0111_B There are 8 valid entries in the queue others: Reserved</p>
0	4	r	Reserved, write 0, read as 0
EMPTY	5	rh	<p>Queue Empty</p> <p>0_B There are valid entries in the queue (see FILL) 1_B No valid entries (queue is empty)</p>
0	6	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
REQGT	7	rh	Request Gate Level Monitors the level at the selected REQGT input. 0 _B The gate input is low 1 _B The gate input is high
EV	8	rh	Event Detected Indicates that an event has been detected while at least one valid entry has been in the queue (queue register 0 or backup stage). Once set, this bit is cleared automatically when the requested conversion is started. 0 _B No trigger event 1 _B A trigger event has been detected
0	[31:9]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The Queue Input Register is the entry point for conversion requests of a queued request source.

GxQINR0 (x = 0 - 1)

Queue 0 Input Register, Group x

$$(x * 0400_H + 0510_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	EX TR	EN SI	RF	REQCHNR				
r	r	r	r	r	r	r	r	w	w	w	w				

Field	Bits	Type	Description
REQCHNR	[4:0]	w	Request Channel Number Defines the channel number to be converted
RF	5	w	Refill 0 _B No refill: this queue entry is converted once and then invalidated 1 _B Automatic refill: this queue entry is automatically reloaded into QINRx when the related conversion is started
ENSI	6	w	Enable Source Interrupt 0 _B No request source interrupt 1 _B A request source event interrupt is generated upon a request source event (related conversion is finished)
EXTR	7	w	External Trigger Enables the external trigger functionality. 0 _B A valid queue entry immediately leads to a conversion request. 1 _B A valid queue entry waits for a trigger event to occur before issuing a conversion request.
0	[31:8]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

*Note: Registers QINRx share addresses with registers QBURx.
Write operations target the control bits in register QINRx. Read operations return the status bits from register QBURx.*

Versatile Analog-to-Digital Converter (VADC)

The queue registers 0 monitor the status of the pending request (queue stage 0).

GxQ0R0 (x = 0 - 1)

Queue 0 Register 0, Group x (x * 0400_H + 050C_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	V	EX TR	EN SI	RF	REQCHNR				
r	r	r	r	r	r	r	rh	rh	rh	rh	rh				

Field	Bits	Type	Description
REQCHNR	[4:0]	rh	Request Channel Number Stores the channel number to be converted.
RF	5	rh	Refill Selects the handling of handled requests. 0 _B The request is discarded after the conversion start. 1 _B The request is automatically refilled into the queue after the conversion start.
ENSI	6	rh	Enable Source Interrupt 0 _B No request source interrupt 1 _B A request source event interrupt is generated upon a request source event (related conversion is finished)
EXTR	7	rh	External Trigger Enables external trigger events. 0 _B A valid queue entry immediately leads to a conversion request 1 _B The request handler waits for a trigger event
V	8	rh	Request Channel Number Valid Indicates a valid queue entry in queue register 0. 0 _B No valid queue entry 1 _B The queue entry is valid and leads to a conversion request

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
0	[31:9]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The Queue Backup Registers monitor the status of an aborted queued request.

GxQBUR0 (x = 0 - 1)

Queue 0 Backup Register, Group x

$$(x * 0400_H + 0510_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	V	EXTR	ENSI	RF	REQCHNR				
r	r	r	r	r	r	r	rh	rh	rh	rh	rh				

Field	Bits	Type	Description
REQCHNR	[4:0]	rh	Request Channel Number The channel number of the aborted conversion that has been requested by this request source
RF	5	rh	Refill The refill control bit of the aborted conversion
ENSI	6	rh	Enable Source Interrupt The enable source interrupt control bit of the aborted conversion
EXTR	7	rh	External Trigger The external trigger control bit of the aborted conversion
V	8	rh	Request Channel Number Valid Indicates if the entry (REQCHNR, RF, TR, ENSI) in the queue backup register is valid. Bit V is set when a running conversion (that has been requested by this request source) is aborted, it is cleared when the aborted conversion is restarted. 0 _B Backup register not valid 1 _B Backup register contains a valid entry. This will be requested before a valid entry in queue register 0 (stage 0) will be requested.
0	[31:9]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

*Note: Registers QBURx share addresses with registers QINRx.
Read operations return the status bits from register QBURx. Write operations target the control bits in register QINRx.*

Versatile Analog-to-Digital Converter (VADC)

Registers of Group Scan Source

There is a separate register set for each group scan source. These sources can be operated independently.

The control register of the autoscan source selects the external gate and/or trigger signals.

Write control bits allow separate control of each function with a simple write access.

GxASCTRL (x = 0 - 1)

Autoscan Source Control Register, Group x
(x * 0400_H + 0520_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TM WC	0	0	TM EN	0	0	0	0	GT WC	0	0	GT LVL			GT SEL	
w	r	r	rw	r	r	r	r	w	r	r	rh			rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XT WC	XT MODE	XT LVL			XT SEL			0	0	0	0	SRCRESREG			
w	rw	rh			rw			r	r	r	r				rw

Field	Bits	Type	Description
SRCRESREG	[3:0]	rw	Source-specific Result Register 0000 _B Use GxCHCTry.RESREG to select a group result register 0001 _B Store result in group result register GxRES1 ... 1111 _B Store result in group result register GxRES15
0	[7:4]	r	Reserved, write 0, read as 0
XTSEL	[11:8]	rw	External Trigger Input Selection The connected trigger input signals are listed in Table 19-13 “Digital Connections in the MMC4300” on Page 19-135 <i>Note: XTSEL = 1111_B uses the selected gate input as trigger source (ENGT must be 0X_B).</i>
XTLVL	12	rh	External Trigger Level Current level of the selected trigger input

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
XTMODE	[14:13]	rw	Trigger Operating Mode 00 _B No external trigger 01 _B Trigger event upon a falling edge 10 _B Trigger event upon a rising edge 11 _B Trigger event upon any edge
XTWC	15	w	Write Control for Trigger Configuration 0 _B No write access to trigger configuration 1 _B Bitfields XTMODE and XTSEL can be written
GTSEL	[19:16]	rw	Gate Input Selection The connected gate input signals are listed in Table 19-13 “Digital Connections in the XMC4300” on Page 19-135
GTLVL	20	rh	Gate Input Level Current level of the selected gate input
0	[22:21]	r	Reserved, write 0, read as 0
GTWC	23	w	Write Control for Gate Configuration 0 _B No write access to gate configuration 1 _B Bitfield GTSEL can be written
0	[27:24]	r	Reserved, write 0, read as 0
TMEN	28	rw	Timer Mode Enable 0 _B No timer mode: standard gating mechanism can be used 1 _B Timer mode for equidistant sampling enabled: standard gating mechanism must be disabled
0	[30:29]	r	Reserved, write 0, read as 0
TMWC	31	w	Write Control for Timer Mode 0 _B No write access to timer mode 1 _B Bitfield TMEN can be written

Versatile Analog-to-Digital Converter (VADC)

The Conversion Request Mode Register configures the operating mode of the channel scan request source.

GxASMR (x = 0 - 1)

Autoscan Source Mode Register, Group x

(x * 0400_H + 0524_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RPT DIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	LD EV	CLR PND	REQ GT	0	LDM	SCAN	EN SI	EN TR	ENGT	
r	r	r	r	r	r	w	w	rh	r	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
ENGT	[1:0]	rw	<p>Enable Gate</p> <p>Selects the gating functionality for source 1.</p> <p>00_B No conversion requests are issued</p> <p>01_B Conversion requests are issued if at least one pending bit is set</p> <p>10_B Conversion requests are issued if at least one pending bit is set and REQGTx = 1.</p> <p>11_B Conversion requests are issued if at least one pending bit is set and REQGTx = 0.</p> <p><i>Note: REQGTx is the selected gating signal.</i></p>
ENTR	2	rw	<p>Enable External Trigger</p> <p>0_B External trigger disabled</p> <p>1_B The selected edge at the selected trigger input signal REQTR generates the load event</p>
ENSI	3	rw	<p>Enable Source Interrupt</p> <p>0_B No request source interrupt</p> <p>1_B A request source interrupt is generated upon a request source event (last pending conversion is finished)</p>

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
SCAN	4	rw	Autoscan Enable 0 _B No autoscan 1 _B Autoscan functionality enabled: a request source event automatically generates a load event
LDM	5	rw	Autoscan Source Load Event Mode 0 _B Overwrite mode: Copy all bits from the select registers to the pending registers upon a load event 1 _B Combine mode: Set all pending bits that are set in the select registers upon a load event (logic OR)
0	6	r	Reserved, write 0, read as 0
REQGT	7	rh	Request Gate Level Monitors the level at the selected REQGT input. 0 _B The gate input is low 1 _B The gate input is high
CLRPND	8	w	Clear Pending Bits 0 _B No action 1 _B The bits in register GxASPNDx are cleared
LDEV	9	w	Generate Load Event 0 _B No action 1 _B A load event is generated
0	[15:10]	r	Reserved, write 0, read as 0
RPTDIS	16	rw	Repeat Disable 0 _B A cancelled conversion is repeated 1 _B A cancelled conversion is discarded
0	[31:17]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The Channel Select Register selects the channels to be converted by the group scan request source. Its bits are used to update the pending register, when a load event occurs.

The number of valid channel bits depends on the channels available in the respective product type (please refer to “**Product-Specific Configuration**” on Page 19-132).

GxASSEL (x = 0 - 1)

Autoscan Source Channel Select Register, Group x

$$(x * 0400_H + 0528_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	CH SEL	CH SEL	CH SEL	CH SEL	CH SEL	CH SEL	CH SEL	CH SEL
r	r	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
								7	6	5	4	3	2	1	0

Field	Bits	Type	Description
CHSELY (y = 0 - 7)	y	rwh	Channel Selection Each bit (when set) enables the corresponding input channel of the respective group to take part in the scan sequence. 0 _B Ignore this channel 1 _B This channel is part of the scan sequence
0	[31:8]	r	Reserved, write 0, read as 0

Note: Register GxASSEL is also updated when writing a pattern to register GxASPND.

The Channel Pending Register indicates the channels to be converted in the current conversion sequence. They are updated from the select register, when a load event occurs.

Versatile Analog-to-Digital Converter (VADC)

GxASPND (x = 0 - 1)

Autoscan Source Pending Register, Group x

(x * 0400_H + 052C_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	CH PND 7	CH PND 6	CH PND 5	CH PND 4	CH PND 3	CH PND 2	CH PND 1	CH PND 0
r	r	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CHPNDy (y = 0 - 7)	y	rwh	Channels Pending Each bit (when set) request the conversion of the corresponding input channel of the respective group. 0 _B Ignore this channel 1 _B Request conversion of this channel
0	[31:8]	r	Reserved, write 0, read as 0

Note: Writing to register GxASPND automatically updates register GxASSEL.

Versatile Analog-to-Digital Converter (VADC)

Registers of Background Scan Source

There is a single register set for the background scan source. This source is common for the complete VADC.

The control register of the background request source selects the external gate and/or trigger signals.

Write control bits allow separate control of each function with a simple write access.

BRCTRL

Background Request Source Control Register

(0200_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	GT WC	0	0	GT LVL			GT SEL	
r	r	r	r	r	r	r	r	w	r	r	rh			rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XT WC	XT MODE	XT LVL			XT SEL			0	0	0	0				SRCRESREG
w	rw	rh			rw			r	r	r	r				rw

Field	Bits	Type	Description
SRCRESREG	[3:0]	rw	Source-specific Result Register 0000 _B Use GxCHCTry.RESREG to select a group result register 0001 _B Store result in group result register GxRES1 ... 1111 _B Store result in group result register GxRES15
0	[7:4]	r	Reserved, write 0, read as 0
XTSEL	[11:8]	rw	External Trigger Input Selection The connected trigger input signals are listed in Table 19-13 “Digital Connections in the XMC4300” on Page 19-135 <i>Note: XTSEL = 1111_B uses the selected gate input as trigger source (ENGT must be 0X_B).</i>
XTLVL	12	rh	External Trigger Level Current level of the selected trigger input

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
XTMODE	[14:13]	rw	Trigger Operating Mode 00 _B No external trigger 01 _B Trigger event upon a falling edge 10 _B Trigger event upon a rising edge 11 _B Trigger event upon any edge
XTWC	15	w	Write Control for Trigger Configuration 0 _B No write access to trigger configuration 1 _B Bitfields XTMODE and XTSEL can be written
GTSEL	[19:16]	rw	Gate Input Selection The connected gate input signals are listed in Table 19-13 “Digital Connections in the XMC4300” on Page 19-135
GTLVL	20	rh	Gate Input Level Current level of the selected gate input
0	[22:21]	r	Reserved, write 0, read as 0
GTWC	23	w	Write Control for Gate Configuration 0 _B No write access to gate configuration 1 _B Bitfield GTSEL can be written
0	[31:24]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The Conversion Request Mode Register configures the operating mode of the background request source.

BRSMR

Background Request Source Mode Register
(0204_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RPT DIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	LD EV	CLR PND	REQ GT	0	LDM	SCAN	EN SI	EN TR	ENGT	
r	r	r	r	r	r	w	w	rh	r	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
ENGT	[1:0]	rw	<p>Enable Gate</p> <p>Selects the gating functionality for source 1.</p> <p>00_B No conversion requests are issued</p> <p>01_B Conversion requests are issued if at least one pending bit is set</p> <p>10_B Conversion requests are issued if at least one pending bit is set and REQGTx = 1.</p> <p>11_B Conversion requests are issued if at least one pending bit is set and REQGTx = 0.</p> <p><i>Note: REQGTx is the selected gating signal.</i></p>
ENTR	2	rw	<p>Enable External Trigger</p> <p>0_B External trigger disabled</p> <p>1_B The selected edge at the selected trigger input signal REQTR generates the load event</p>
ENSI	3	rw	<p>Enable Source Interrupt</p> <p>0_B No request source interrupt</p> <p>1_B A request source interrupt is generated upon a request source event (last pending conversion is finished)</p>

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
SCAN	4	rw	Autoscan Enable 0 _B No autoscan 1 _B Autoscan functionality enabled: a request source event automatically generates a load event
LDM	5	rw	Autoscan Source Load Event Mode 0 _B Overwrite mode: Copy all bits from the select registers to the pending registers upon a load event 1 _B Combine mode: Set all pending bits that are set in the select registers upon a load event (logic OR)
0	6	r	Reserved, write 0, read as 0
REQGT	7	rh	Request Gate Level Monitors the level at the selected REQGT input. 0 _B The gate input is low 1 _B The gate input is high
CLRPND	8	w	Clear Pending Bits 0 _B No action 1 _B The bits in registers BRSPNDx are cleared
LDEV	9	w	Generate Load Event 0 _B No action 1 _B A load event is generated
0	[15:10]	r	Reserved, write 0, read as 0
RPTDIS	16	rw	Repeat Disable 0 _B A cancelled conversion is repeated 1 _B A cancelled conversion is discarded
0	[31:17]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The Channel Select Registers select the channels to be converted by the background request source (channel scan source). Its bits are used to update the pending registers, when a load event occurs.

The number of valid channel bits depends on the channels available in the respective product type (please refer to **“Product-Specific Configuration” on Page 19-132**).

*Note: Priority channels selected in registers **GxCHASS (x = 0 - 1)** will not be converted.*

BRSELx (x = 0 - 1)

Background Request Source Channel Select Register, Group x
(0180_H + x * 0004_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	CH SEL G7	CH SEL G6	CH SEL G5	CH SEL G4	CH SEL G3	CH SEL G2	CH SEL G1	CH SEL G0
r	r	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CHSELGy (y = 0 - 7)	y	rwh	Channel Selection Group x Each bit (when set) enables the corresponding input channel of the respective group to take part in the background scan sequence. 0 _B Ignore this channel 1 _B This channel is part of the scan sequence
0	[31:8]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The Channel Pending Registers indicate the channels to be converted in the current conversion sequence. They are updated from the select registers, when a load event occurs.

BRSPNDx (x = 0 - 1)

Background Request Source Pending Register, Group x

(01C0_H + x * 0004_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	CH PND G7	CH PND G6	CH PND G5	CH PND G4	CH PND G3	CH PND G2	CH PND G1	CH PND G0
r	r	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CHPNDGy (y = 0 - 7)	y	rwh	Channels Pending Group x Each bit (when set) request the conversion of the corresponding input channel of the respective group. 0 _B Ignore this channel 1 _B Request conversion of this channel
0	[31:8]	r	Reserved, write 0, read as 0

Note: Writing to any of registers BRSPNDx automatically updates the corresponding register BRSELx and generates a load event that copies all bits from all registers BRSELx to BRSPNDx.

Use this shortcut only when writing the last word of the request pattern.

Versatile Analog-to-Digital Converter (VADC)

19.13.5 Channel Control Registers

G0CHCTry (y = 0 - 7)

Group 0, Channel y Ctrl. Reg. (0600_H + y * 0004_H) **Reset Value: 0000 0000_H**

G1CHCTry (y = 0 - 7)

Group 1, Channel y Ctrl. Reg. (0A00_H + y * 0004_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	BWD EN	BWD CH	0	0	0	0	0	0	0	RES POS	RES TBS	RESREG			
r	rw	rw	r	r	r	r	r	r	r	rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	REF SEL	SY NC	CHEV MODE	BNDSELU		BNDSELL		0	0	ICLSEL		
r	r	r	r	rw	rw	rw	rw		rw		r	r	rw		

Field	Bits	Type	Description
ICLSEL	[1:0]	rw	Input Class Select 00 _B Use group-specific class 0 01 _B Use group-specific class 1 10 _B Use global class 0 11 _B Use global class 1
0	[3:2]	r	Reserved, write 0, read as 0
BNDSELL	[5:4]	rw	Lower Boundary Select 00 _B Use group-specific boundary 0 01 _B Use group-specific boundary 1 10 _B Use global boundary 0 11 _B Use global boundary 1
BNDSELU	[7:6]	rw	Upper Boundary Select 00 _B Use group-specific boundary 0 01 _B Use group-specific boundary 1 10 _B Use global boundary 0 11 _B Use global boundary 1

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
CHEVMODE	[9:8]	rw	Channel Event Mode Generate a channel event either in normal compare mode (NCM) with limit checking ¹⁾ or in Fast Compare Mode (FCM) ²⁾ 00 _B Never 01 _B NCM: If result is inside the boundary band FCM: If result becomes high (above cmp. val.) 10 _B NCM: If result is outside the boundary band FCM: If result becomes low (below cmp. val.) 11 _B NCM: Always (ignore band) FCM: If result switches to either level
SYNC	10	rw	Synchronization Request 0 _B No synchroniz. request, standalone operation 1 _B Request a synchronized conversion of this channel (only taken into account for a master)
REFSEL	11	rw	Reference Input Selection Defines the reference voltage input to be used for conversions on this channel. 0 _B Standard reference input V_{AREF} 1 _B Alternate reference input from CH0 ³⁾
0	[15:12]	rw	Reserved, write 0, read as 0
RESREG	[19:16]	rw	Result Register 0000 _B Store result in group result register GxRES0 ... 1111 _B Store result in group result register GxRES15
RESTBS	20	rw	Result Target for Background Source 0 _B Store results in the selected group result register 1 _B Store results in the global result register
RESPOS	21	rw	Result Position 0 _B Store results left-aligned 1 _B Store results right-aligned
0	[27:22]	r	Reserved, write 0, read as 0
BWDCH	[29:28]	rw	Broken Wire Detection Channel 00 _B Select V_{AGND} 01 _B Select V_{AREF} 10 _B Reserved 11 _B Reserved

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
BWDEN	30	rw	Broken Wire Detection Enable 0 _B Normal operation 1 _B Additional preparation phase is enabled
0	31	r	Reserved, write 0, read as 0

- 1) The boundary band is defined as the area where the result is less than or equal to the selected upper boundary and greater than or equal to the selected lower boundary, see [Section 19.7.5](#).
- 2) The result is bit FCR in the selected result register.
- 3) Some channels cannot select an alternate reference.

GxICLASS0 (x = 0 - 1)

Input Class Register 0, Group x

(x * 0400_H + 04A0_H) **Reset Value: 0000 0000_H**

GxICLASS1 (x = 0 - 1)

Input Class Register 1, Group x

(x * 0400_H + 04A4_H) **Reset Value: 0000 0000_H**

GLOBICLASSy (y = 0 - 1)

Input Class Register y, Global

(00A0_H + y * 0004_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	CME		0	0	0	STCE					
r	r	r	r	r	rw		r	r	r	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	CMS		0	0	0	STCS					
r	r	r	r	r	rw		r	r	r	rw					

Field	Bits	Type	Description
STCS	[4:0]	rw	Sample Time Control for Standard Conversions Number of additional clock cycles to be added to the minimum sample phase of 2 analog clock cycles: Coding and resulting sample time see Table 19-9 . For conversions of external channels, the value from bitfield STCE can be used.
0	[7:5]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
CMS	[10:8]	rw	Conversion Mode for Standard Conversions 000 _B 12-bit conversion 001 _B 10-bit conversion 010 _B 8-bit conversion 011 _B Reserved 100 _B Reserved 101 _B 10-bit fast compare mode 110 _B Reserved 111 _B Reserved
0	[15:11]	r	Reserved, write 0, read as 0
STCE	[20:16]	rw	Sample Time Control for EMUX Conversions Number of additional clock cycles to be added to the minimum sample phase of 2 analog clock cycles: Coding and resulting sample time see Table 19-9 . For conversions of standard channels, the value from bitfield STCS is used.
0	[23:21]	r	Reserved, write 0, read as 0
CME	[26:24]	rw	Conversion Mode for EMUX Conversions 000 _B 12-bit conversion 001 _B 10-bit conversion 010 _B 8-bit conversion 011 _B Reserved 100 _B Reserved 101 _B 10-bit fast compare mode 110 _B Reserved 111 _B Reserved
0	[31:27]	r	Reserved, write 0, read as 0

Table 19-9 Sample Time Coding

STCS / STCE	Additional Clock Cycles	Sample Time
0 0000 _B	0	$2 / f_{ADCI}$
0 0001 _B	1	$3 / f_{ADCI}$
...
0 1111 _B	15	$17 / f_{ADCI}$
1 0000 _B	16	$18 / f_{ADCI}$
1 0001 _B	32	$34 / f_{ADCI}$

Versatile Analog-to-Digital Converter (VADC)

Table 19-9 Sample Time Coding (cont'd)

STCS / STCE	Additional Clock Cycles	Sample Time
...
1 1110 _B	240	242 / f_{ADCI}
1 1111 _B	256	258 / f_{ADCI}

Versatile Analog-to-Digital Converter (VADC)

19.13.6 Result Registers

The group result control registers select the behavior of the result registers of a given group.

G0RCR_y (y = 0 - 15)

Group 0 Result Control Reg. y (0680_H + y * 0004_H) **Reset Value: 0000 0000_H**

G1RCR_y (y = 0 - 15)

Group 1 Result Control Reg. y (0A80_H + y * 0004_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRG EN	0	0	0	0	FEN	WFR	0	0	DMM	DRCTR					
rw	r	r	r	r	rw	rw	r	r	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
0	[15:0]	r	Reserved, write 0, read as 0
DRCTR	[19:16]	rw	Data Reduction Control Defines how result values are stored/accumulated in this register for the final result. The data reduction counter DRC can be loaded from this bitfield. The function of bitfield DRCTR is determined by bitfield DMM.
DMM	[21:20]	rw	Data Modification Mode 00 _B Standard data reduction (accumulation) 01 _B Result filtering mode ¹⁾ 10 _B Difference mode 11 _B Reserved See “Data Modification” on Page 19-40
0	[23:22]	r	Reserved, write 0, read as 0
WFR	24	rw	Wait-for-Read Mode Enable 0 _B Overwrite mode 1 _B Wait-for-read mode enabled for this register

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
FEN	[26:25]	rw	FIFO Mode Enable 00 _B Separate result register 01 _B Part of a FIFO structure: copy each new valid result 10 _B Maximum mode: copy new result if bigger 11 _B Minimum mode: copy new result if smaller
0	[30:27]	r	Reserved, write 0, read as 0
SRGEN	31	rw	Service Request Generation Enable 0 _B No service request 1 _B Service request after a result event

1) The filter registers are cleared while bitfield DMM ≠ 01_B.

Versatile Analog-to-Digital Converter (VADC)

The group result registers provide a selectable storage location for all channels of a given group.

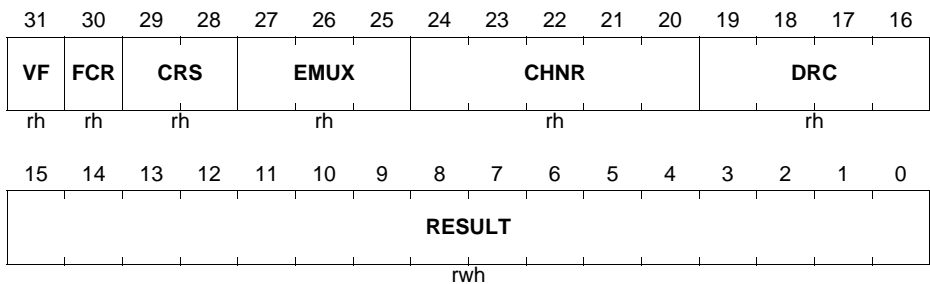
Note: The preset value used in fast compare mode is written to the respective result register. The debug result registers are not writable.

G0RESy (y = 0 - 15)

Group 0 Result Register y **(0700_H + y * 0004_H)** **Reset Value: 0000 0000_H**

G1RESy (y = 0 - 15)

Group 1 Result Register y **(0B00_H + y * 0004_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RESULT	[15:0]	rwh	Result of Most Recent Conversion The position of the result bits within this bitfield depends on the configured operating mode. Please, refer to Section 19.8.2 .
DRC	[19:16]	rh	Data Reduction Counter Indicates the number of values still to be accumulated for the final result. The final result is available and valid flag VF is set when bitfield DRC becomes zero (by decrementing or by reload). See “Data Modification” on Page 19-40
CHNR	[24:20]	rh	Channel Number Indicates the channel number corresponding to the value in bitfield RESULT.
EMUX	[27:25]	rh	External Multiplexer Setting Indicates the setting of the external multiplexer, corresponding to the value in bitfield RESULT. <i>Note: Available in GxRES0 only. Use GxRES0 if EMUX information is required.</i>

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
CRS	[29:28]	rh	Converted Request Source Indicates the request source that as requested the conversion to which the result value in bitfield RESULT belongs. 00 _B Request source 0 01 _B Request source 1 10 _B Request source 2 11 _B Reserved
FCR	30	rh	Fast Compare Result Indicates the result of an operation in Fast Compare Mode. 0 _B Signal level was below compare value 1 _B Signal level was above compare value
VF	31	rh	Valid Flag Indicates a new result in bitfield RESULT or bit FCR. 0 _B No new result available 1 _B Bitfield RESULT has been updated with new result value and has not yet been read, or bit FCR has been updated

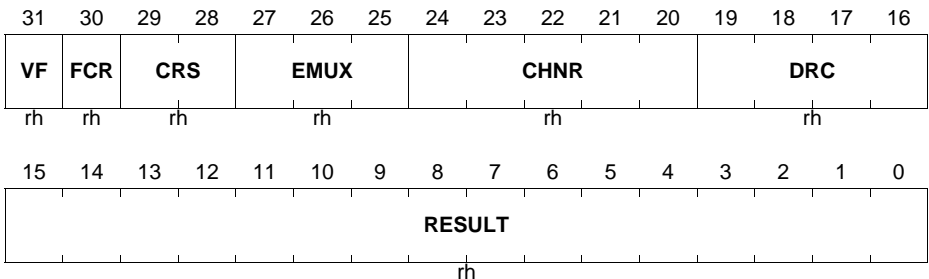
The debug view of the group result registers provides access to all result registers of a given group, however, without clearing the valid flag.

G0RESDy (y = 0 - 15)

Group 0 Result Reg. y, Debug (0780_H + y * 0004_H) **Reset Value: 0000 0000_H**

G1RESDy (y = 0 - 15)

Group 1 Result Reg. y, Debug (0B80_H + y * 0004_H) **Reset Value: 0000 0000_H**



Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
RESULT	[15:0]	rh	Result of Most Recent Conversion The position of the result bits within this bitfield depends on the configured operating mode. Please, refer to Section 19.8.2 .
DRC	[19:16]	rh	Data Reduction Counter Indicates the number of values still to be accumulated for the final result. The final result is available and valid flag VF is set when bitfield DRC becomes zero (by decrementing or by reload). See “Data Modification” on Page 19-40
CHNR	[24:20]	rh	Channel Number Indicates the channel number corresponding to the value in bitfield RESULT.
EMUX	[27:25]	rh	External Multiplexer Setting Indicates the setting of the external multiplexer, corresponding to the value in bitfield RESULT. <i>Note: Available in GxRES0 only. Use GxRES0 if EMUX information is required.</i>
CRS	[29:28]	rh	Converted Request Source Indicates the request source that as requested the conversion to which the result value in bitfield RESULT belongs. 00 _B Request source 0 01 _B Request source 1 10 _B Request source 2 11 _B Reserved
FCR	30	rh	Fast Compare Result Indicates the result of an operation in Fast Compare Mode. 0 _B Signal level was below compare value 1 _B Signal level was above compare value

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
VF	31	rh	Valid Flag Indicates a new result in bitfield RESULT or bit FCR. 0_B No new result available 1_B Bitfield RESULT has been updated with new result value and has not yet been read, or bit FCR has been updated

The global result control register selects the behavior of the global result register.

GLOBRCR

Global Result Control Register (0280_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRG EN	0	0	0	0	0	0	WFR	0	0	0	0	DRCTR			
rw	r	r	r	r	r	r	rw	r	r	r	r	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
0	[15:0]	r	Reserved, write 0, read as 0
DRCTR	[19:16]	rw	Data Reduction Control Defines how result values are stored/accumulated in this register for the final result. The data reduction counter DRC can be loaded from this bitfield. 0000_B Data reduction disabled others: see “Function of Bitfield DRCTR” on Page 19-40¹⁾
0	[23:20]	r	Reserved, write 0, read as 0
WFR	24	rw	Wait-for-Read Mode Enable 0_B Overwrite mode 1_B Wait-for-read mode enabled for this register
0	[30:25]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
SRGEN	31	rw	Service Request Generation Enable 0 _B No service request 1 _B Service request after a result event

1) Only standard data reduction is available for the global result register, i.e. DMM is assumed as 00_B.

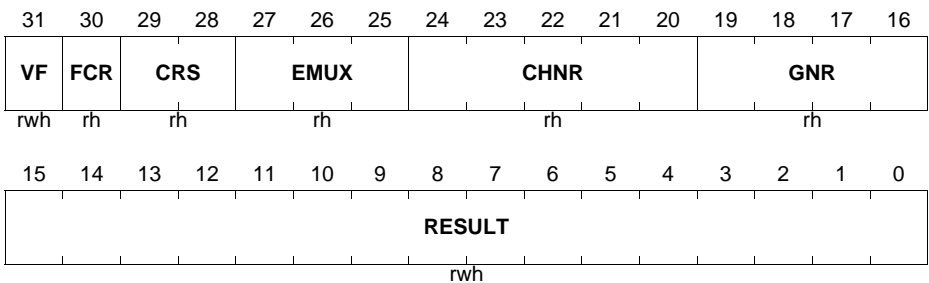
The global result register provides a common storage location for all channels of all groups.

GLOBRES

Global Result Register (0300_H) **Reset Value: 0000 0000_H**

GLOBRESD

Global Result Register, Debug (0380_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RESULT	[15:0]	rwh	Result of most recent conversion The position of the result bits within this bitfield depends on the configured operating mode. ¹⁾ Please, refer to Section 19.8.2 .
GNR	[19:16]	rh	Group Number Indicates the group to which the channel number in bitfield CHNR refers.
CHNR	[24:20]	rh	Channel Number Indicates the channel number corresponding to the value in bitfield RESULT.
EMUX	[27:25]	rh	External Multiplexer Setting Indicates the setting of the external multiplexer, corresponding to the value in bitfield RESULT.

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
CRS	[29:28]	rh	Converted Request Source Indicates the request source that as requested the conversion to which the result value in bitfield RESULT belongs.
FCR	30	rh	Fast Compare Result Indicates the result of an operation in Fast Compare Mode. 0 _B Signal level was below compare value 1 _B Signal level was above compare value
VF	31	rwh	Valid Flag Indicates a new result in bitfield RESULT or bit FCR. 0 _B Read access: No new valid data available Write access: No effect 1 _B Read access: Bitfield RESULT contains valid data and has not yet been read, or bit FCR has been updated Write access: Clear this valid flag and the data reduction counter (overrides a hardware set action) ¹⁾

1) Only writable in register GLOBRES, not in register GLOBRESD.

The valid flag register summarizes the valid flags of all result registers.

GxVFR (x = 0 - 1)

Valid Flag Register, Group x (x * 0400_H + 05F8_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VF15	VF14	VF13	VF12	VF11	VF10	VF9	VF8	VF7	VF6	VF5	VF4	VF3	VF2	VF1	VF0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Versatile Analog-to-Digital Converter (VADC)

The local boundary register GxBOUND defines group-specific boundary values or delta limits for Fast Compare Mode.

The global boundary register GLOBBOUND defines general compare values for all channels.

Depending on the conversion width, the respective left 12/10/8 bits of a bitfield are used. For 10/8-bit results, the lower 2/4 bits must be zero!

GxBOUND (x = 0 - 1)

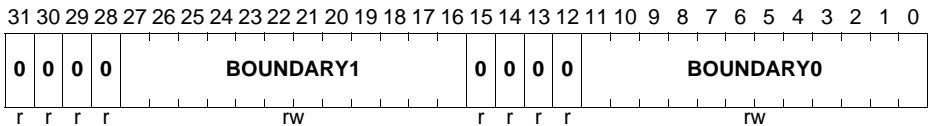
Boundary Select Register, Group x

(x * 0400_H + 04B8_H) **Reset Value: 0000 0000_H**

GLOBBOUND

Global Boundary Select Register (00B8_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
BOUNDARY0	[11:0]	rw	Boundary Value 0 for Limit Checking Standard Mode: This value is compared against the left-aligned conversion result. Fast Compare Mode: This value is added to the reference value (upper delta).
0	[15:12]	r	Reserved, write 0, read as 0
BOUNDARY1	[27:16]	rw	Boundary Value 1 for Limit Checking Standard Mode: This value is compared against the left-aligned conversion result. Fast Compare Mode: This value is subtracted from the reference value (lower delta).
0	[31:28]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The Boundary Flag Register holds the boundary flags themselves together with bits to select the activation condition and the output signal polarity for each flag.

GxBFL (x = 0 - 1)

Boundary Flag Register, Group x

(x * 0400_H + 04C8_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	BFI 3	BFI 2	BFI 1	BFI 0
r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	BFA 3	BFA 2	BFA 1	BFA 0	0	0	0	0	BFL 3	BFL 2	BFL 1	BFL 0
r	r	r	r	rw	rw	rw	rw	r	r	r	r	rh	rh	rh	rh

Field	Bits	Type	Description
BFLz (z = 0 - 3)	z	rh	Boundary Flag z 0 _B Passive state: result has not yet crossed the activation boundary (see bitfield BFAz), or selected gate signal is inactive, or this boundary flag is disabled 1 _B Active state: result has crossed the activation boundary
0	[7:4]	r	Reserved, write 0, read as 0
BFAz (z = 0 - 3)	8 + z	rw	Boundary Flag z Activation Select 0 _B Set boundary flag BFLz if result is above the defined band or compare value, clear if below 1 _B Set boundary flag BFLz if result is below the defined band or compare value, clear if above
0	[15:12]	r	Reserved, write 0, read as 0
BFlz (z = 0 - 3)	16 + z	rw	Boundary Flag z Inversion Control 0 _B Use BFLz directly 1 _B Invert value and use BFLz
0	[31:20]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The Boundary Flag Software Register provides means to set or clear each flag by software.

GxBFLS (x = 0 - 1)

Boundary Flag Software Register, Group x

(x * 0400_H + 04CC_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	BFS 3	BFS 2	BFS 1	BFS 0
r	r	r	r	r	r	r	r	r	r	r	r	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	BFC 3	BFC 2	BFC 1	BFC 0
r	r	r	r	r	r	r	r	r	r	r	r	w	w	w	w

Field	Bits	Type	Description
BFCz (z = 0 - 3)	z	w	Boundary Flag z Clear 0 _B No action 1 _B Clear bit BFLz
0	[15:4]	r	Reserved, write 0, read as 0
BFSz (z = 0 - 3)	16 + z	w	Boundary Flag z Set 0 _B No action 1 _B Set bit BFLz
0	[31:20]	r	Reserved, write 0, read as 0

Note: If a boundary flag is used together with Fast Compare Mode, it is recommended not to direct results from other channels to the corresponding result register.

Versatile Analog-to-Digital Converter (VADC)

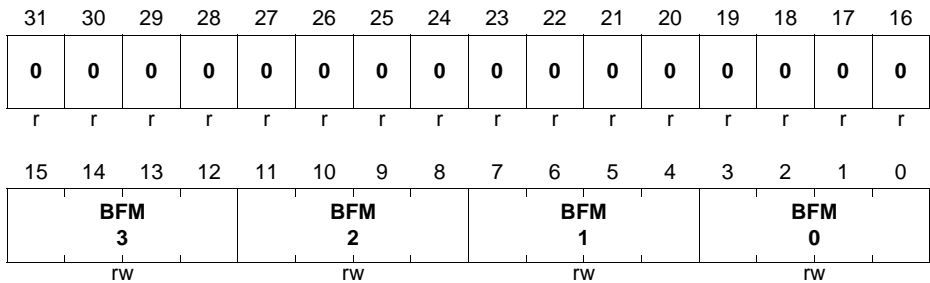
The Boundary Flag Control Register selects the basic operation of the boundary flags.

GxBFLC (x = 0 - 1)

Boundary Flag Control Register, Group x

$$(x * 0400_H + 04D0_H)$$

Reset Value: 0000 0000_H



Field	Bits	Type	Description
BFM0, BFM1, BFM2, BFM3	[3:0], [7:4], [11:8], [15:12]	rw	Boundary Flag y Mode Control 0000 _B Disable boundary flag, BFLy is not changed 0001 _B Always enable boundary flag (follow compare results) 0010 _B Enable boundary flag while gate of source 0 is active, clear BFLy while gate is inactive 0011 _B Enable boundary flag while gate of source 1 is active, clear BFLy while gate is inactive others: Reserved
0	[31:16]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The Boundary Flag Node Pointer Register directs signal VADCGxBFLy to alternate on-chip connections with other modules (in addition to the group-specific outputs). Possible targets are the corresponding common service request lines or the common boundary flag outputs (CBFLOUT0 ... CBFLOUT3).

GxBFLNP (x = 0 - 1)

Boundary Flag Node Pointer Register, Group x

$$(x * 0400_H + 04D4_H)$$

Reset Value: 0000 FFFF_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BFL3NP				BFL2NP				BFL1NP				BFL0NP			
rw				rw				rw				rw			

Field	Bits	Type	Description
BFL0NP, BFL1NP, BFL2NP, BFL3NP	[3:0], [7:4], [11:8], [15:12]	rw	Boundary Flag y Node Pointer 0000 _B Select common boundary flag output 0 ... 0011 _B Select common boundary flag output 3 0100 _B Select shared service request line 0 ... 0111 _B Select shared service request line 3 1111 _B Disabled, no common output signal others: Reserved <i>Note: For shared service request lines see common groups in Table 19-10.</i>
0	[31:16]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

The Synchronization Control Register controls the synchronization of kernels for parallel conversions.

Note: Program register GxSYNCTR only while bitfield GxARBCFG.ANONS = 00_B in all ADC kernels of the conversion group. Set the master's bitfield ANONC to 11_B afterwards.

GxSYNCTR (x = 0 - 1)

Synchronization Control Register, Group x

(x * 0400_H + 04C0_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	EVA LR3	EVA LR2	EVA LR1	0	0	STSEL	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
STSEL	[1:0]	rw	<p>Start Selection</p> <p>Controls the synchronization mechanism of the ADC kernel.</p> <p>00_B Kernel is synchronization master: Use own bitfield GxARBCFG.ANONC</p> <p>01_B Kernel is synchronization slave: Control information from input CI1</p> <p>10_B Kernel is synchronization slave: Control information from input CI2</p> <p>11_B Kernel is synchronization slave: Control information from input CI3</p> <p><i>Note: Control inputs CIx see Figure 19-24, connected kernels see Table 19-11.</i></p>
0	[3:2]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
EVALR1, EVALR2, EVALR3	4, 5, 6	rw	Evaluate Ready Input Rx Enables the ready input signal for a kernel of a conversion group. 0 _B No ready input control 1 _B Ready input Rx is considered for the start of a parallel conversion of this conversion group
0	[31:7]	r	Reserved, write 0, read as 0

GLOBTF

Global Test Functions Register (0160_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	MD WC	0	0	0	0	0	0	PDD
r	r	r	r	r	r	r	r	w	r	r	r	r	r	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CD WC	0	0	0	0	CD SEL	CD EN	CDGR			0	0	0	0		
w	r	r	r	r	rw	rw	rw			r	r	r	r		

Field	Bits	Type	Description
0	[3:0]	r	Reserved, write 0, read as 0
CDGR	[7:4]	rw	Converter Diagnostics Group Defines the group number to be used for converter diagnostics conversions.
CDEN	8	rw	Converter Diagnostics Enable 0 _B All diagnostic pull devices are disconnected 1 _B Diagnostic pull devices connected as selected by bitfield CDSEL
CDSEL	[10:9]	rw	Converter Diagnostics Pull-Devices Select 00 _B Connected to VAREF 01 _B Connected to VAGND 10 _B Connected to 1/3rd VAREF 11 _B Connected to 2/3rd VAREF
0	[14:11]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
CDWC	15	w	Write Control for Conversion Diagnostics 0 _B No write access to parameters 1 _B Bitfields CDSEL, CDEN, CDGR can be written
PDD	16	rw	Pull-Down Diagnostics Enable 0 _B Disconnected 1 _B The pull-down diagnostics device is active <i>Note: Channels with pull-down diagnostics device are marked in Table 19-12.</i>
0	[22:17]	r	Reserved, write 0, read as 0
MDWC	23	w	Write Control for Multiplexer Diagnostics 0 _B No write access to parameters 1 _B Bitfield PDD can be written
0	[31:24]	r	Reserved, write 0, read as 0

GxEMUXCTR (x = 0 - 1)

External Multiplexer Control Register, Group x

$$(x * 0400_H + 05F0_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EMX WC	EMX CSS	EMX ST	EMX COD	EMUX MODE						EMUX CH					
w	rw	rw	rw	rw						rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0		EMUX ACT		0	0	0	0	0		EMUX SET	
r	r	r	r	r		rh		r	r	r	r	r		rw	

Field	Bits	Type	Description
EMUXSET	[2:0]	rw	External Multiplexer Start Selection¹⁾ Defines the initial setting for the external multiplexer.
0	[7:3]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
EMUXACT	[10:8]	rh	External Multiplexer Actual Selection Defines the current value for the external multiplexer selection. This bitfield is loaded from bitfield EMUXSET and modified according to the operating mode selected by bitfield EMUXMODE.
0	[15:11]	r	Reserved, write 0, read as 0
EMUXCH	[25:16]	rw	External Multiplexer Channel Select Defines the channel(s) to which the external multiplexer control is applied. EMXCSS = 0: Channel number , the lower 5 bits select an arbitrary channel (valid numbers are limited by the number of available channels, unused bits shall be 0) EMXCSS = 1: Channel enable , each bit enables the associated channel (multiple channels can be selected/enabled)
EMUXMODE	[27:26]	rw	External Multiplexer Mode 00 _B Software control (no hardware action) 01 _B Steady mode (use EMUXSET value) 10 _B Single-step mode ¹⁾²⁾ 11 _B Sequence mode ¹⁾
EMXCOD	28	rw	External Multiplexer Coding Scheme 0 _B Output the channel number in binary code 1 _B Output the channel number in Gray code
EMXST	29	rw	External Multiplexer Sample Time Control 0 _B Use STCE whenever the setting changes 1 _B Use STCE for each conversion of an external channel
EMXCSS	30	r	External Multiplexer Channel Selection Style 0 _B Channel number: Bitfield EMUXCH selects an arbitrary channel 1 _B Channel enable: Each bit of bitfield EMUXCH selects the associated channel for EMUX control
EMXWC	31	w	Write Control for EMUX Configuration 0 _B No write access to EMUX cfg. 1 _B Bitfields EMUXMODE, EMXCOD, EMXST, EMXCSS can be written

Versatile Analog-to-Digital Converter (VADC)

- 1) For single-step mode and sequence mode: Select the start value before selecting the respective mode.
- 2) Single-step mode modifies the EMUX channel number each time an EMUX-enabled channel is converted. Therefore, single-step mode works best with a single channel, because otherwise some external channels may be skipped.

Versatile Analog-to-Digital Converter (VADC)

Register EMUXSEL is a global register which assigns an arbitrary group to each of the EMUX interfaces.

EMUXSEL

External Multiplexer Select Register

(03F0_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	EMUX GRP1				EMUX GRP0			
r	r	r	r	r	r	r	r	rw				rw			

Field	Bits	Type	Description
EMUXGRP0, EMUXGRP1	[3:0], [7:4]	rw	External Multiplexer Group for Interface x Defines the group whose external multiplexer control signals are routed to EMUX interface x. ¹⁾
0	[31:8]	r	Reserved, write 0, read as 0

1) The pins that are associated with each EMUX interface are listed in [Table 19-13 "Digital Connections in the XMC4300"](#) on [Page 19-135](#).

19.13.8 Service Request Registers

GxSEFLAG (x = 0 - 1)

Source Event Flag Register, Group x

(x * 0400_H + 0588_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	SEV 1	SEV 0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	rwh	rwh

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
SEV0, SEV1	0, 1	rwh	Source Event 0/1 0 _B No source event 1 _B A source event has occurred
0	[31:2]	r	Reserved, write 0, read as 0

*Note: Software can set all flags in register GxSEFLAG and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear all flags in register GxSEFLAG by writing 1 to the respective bit in register GxSEFCLR.*

GxCEFLAG (x = 0 - 1)

Channel Event Flag Register, Group x

$$(x * 0400_H + 0580_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	CEV 7	CEV 6	CEV 5	CEV 4	CEV 3	CEV 2	CEV 1	CEV 0
r	r	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CEVy (y = 0 - 7)	y	rwh	Channel Event for Channel y 0 _B No channel event 1 _B A channel event has occurred
0	[31:8]	r	Reserved, write 0, read as 0

*Note: Software can set all flags in register GxCEFLAG and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear all flags in register GxCEFLAG by writing 1 to the respective bit in register GxCEFCLR.*

Versatile Analog-to-Digital Converter (VADC)

GxREFLAG (x = 0 - 1)

Result Event Flag Register, Group x

$$(x * 0400_H + 0584_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV 15	REV 14	REV 13	REV 12	REV 11	REV 10	REV 9	REV 8	REV 7	REV 6	REV 5	REV 4	REV 3	REV 2	REV 1	REV 0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
REVy (y = 0 - 15)	y	rwh	Result Event for Result Register y 0 _B No result event 1 _B New result was stored in register GxRESy
0	[31:16]	r	Reserved, write 0, read as 0

*Note: Software can set all flags in register GxREFLAG and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear all flags in register GxREFLAG by writing 1 to the respective bit in register GxREFCLR.*

GxSEFCLR (x = 0 - 1)

Source Event Flag Clear Register, Group x

$$(x * 0400_H + 0598_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	SEV 1	SEV 0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	w	w

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
SEV0, SEV1	0, 1	w	Clear Source Event 0/1 0 _B No action 1 _B Clear the source event flag in GxSEFLAG
0	[31:2]	r	Reserved, write 0, read as 0

GxCEFCLR (x = 0 - 1)

Channel Event Flag Clear Register, Group x

$$(x * 0400_H + 0590_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	CEV 7	CEV 6	CEV 5	CEV 4	CEV 3	CEV 2	CEV 1	CEV 0
r	r	r	r	r	r	r	r	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
CEVy (y = 0 - 7)	y	w	Clear Channel Event for Channel y 0 _B No action 1 _B Clear the channel event flag in GxCEFLAG
0	[31:8]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

GxREFCLR (x = 0 - 1)

Result Event Flag Clear Register, Group x

(x * 0400_H + 0594_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV 15	REV 14	REV 13	REV 12	REV 11	REV 10	REV 9	REV 8	REV 7	REV 6	REV 5	REV 4	REV 3	REV 2	REV 1	REV 0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
REVy (y = 0 - 15)	y	w	Clear Result Event for Result Register y 0 _B No action 1 _B Clear the result event flag in GxREFLAG
0	[31:16]	r	Reserved, write 0, read as 0

GLOBEFLAG

Global Event Flag Register

(00E0_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	REV GLB CLR	0	0	0	0	0	0	0	SEV GLB CLR
r	r	r	r	r	r	r	w	r	r	r	r	r	r	r	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	REV GLB	0	0	0	0	0	0	0	SEV GLB
r	r	r	r	r	r	r	rwh	r	r	r	r	r	r	r	rwh

Field	Bits	Type	Description
SEVGLB	0	rwh	Source Event (Background) 0 _B No source event 1 _B A source event has occurred

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
0	[7:1]	r	Reserved, write 0, read as 0
REVGLB	8	rwh	Global Result Event 0 _B No result event 1 _B New result was stored in register GLOBRES
0	[15:9]	r	Reserved, write 0, read as 0
SEVGLBCLR	16	w	Clear Source Event (Background) 0 _B No action 1 _B Clear the source event flag SEVGLB
0	[23:17]	r	Reserved, write 0, read as 0
REVGLBCLR	24	w	Clear Global Result Event 0 _B No action 1 _B Clear the result event flag REVGLB
0	[31:25]	r	Reserved, write 0, read as 0

*Note: Software can set flags REVGLB and SEVGLB and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear these flags by writing 1 to bit REVGLBCLR and SECGLBCLR, respectively.*

GxSEVNP (x = 0 - 1)

Source Event Node Pointer Register, Group x

$$(x * 0400_H + 05C0_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	SEV1NP			SEV0NP				
r	r	r	r	r	r	r	r	rw			rw				

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
SEV0NP, SEV1NP	[3:0], [7:4]	rw	Service Request Node Pointer Source Event ¹⁾ Routes the corresponding event trigger to one of the service request lines (nodes). 0000 _B Select service request line 0 of group x ... 0011 _B Select service request line 3 of group x 0100 _B Select shared service request line 0 ... 0111 _B Select shared service request line 3 1xxx _B Reserved <i>Note: For shared service request lines see common groups in Table 19-10.</i>
0	[31:8]	r	Reserved, write 0, read as 0

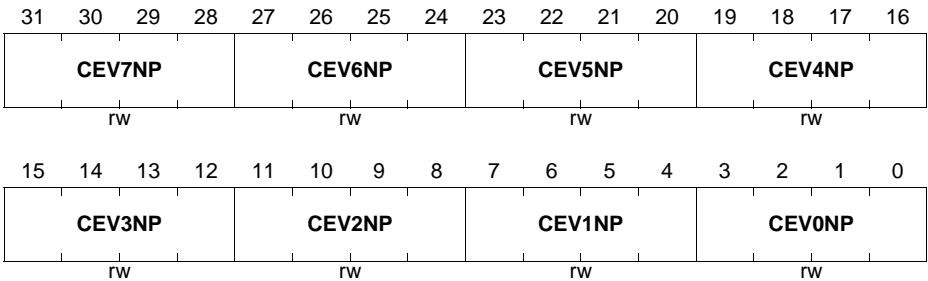
1) Source 0 is an 8-stage queued source, source 1 is a channel scan source.

GxCEVNP0 (x = 0 - 1)

Channel Event Node Pointer Register 0, Group x

$$(x * 0400_H + 05A0_H)$$

Reset Value: 0000 0000_H



Versatile Analog-to-Digital Converter (VADC)

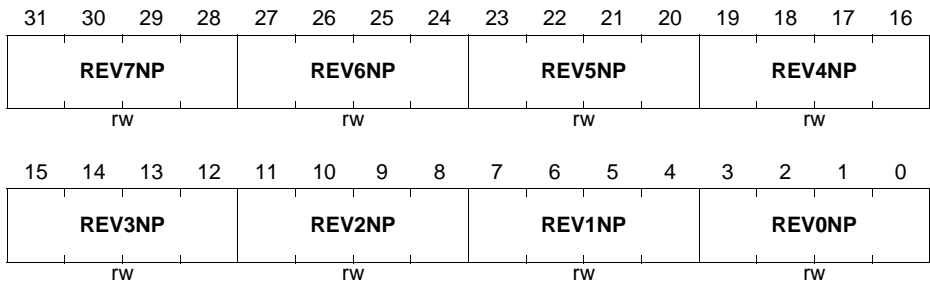
Field	Bits	Type	Description
CEV0NP, CEV1NP, CEV2NP, CEV3NP, CEV4NP, CEV5NP, CEV6NP, CEV7NP	[3:0], [7:4], [11:8], [15:12], [19:16], [23:20], [27:24], [31:28]	rw	Service Request Node Pointer Channel Event i Routes the corresponding event trigger to one of the service request lines (nodes). 0000 _B Select service request line 0 of group x ... 0011 _B Select service request line 3 of group x 0100 _B Select shared service request line 0 ... 0111 _B Select shared service request line 3 1xxx _B Reserved <i>Note: For shared service request lines see common groups in Table 19-10.</i>

GxREVNP0 (x = 0 - 1)

Result Event Node Pointer Register 0, Group x

$$(x * 0400_H + 05B0_H)$$

Reset Value: 0000 0000_H



Versatile Analog-to-Digital Converter (VADC)

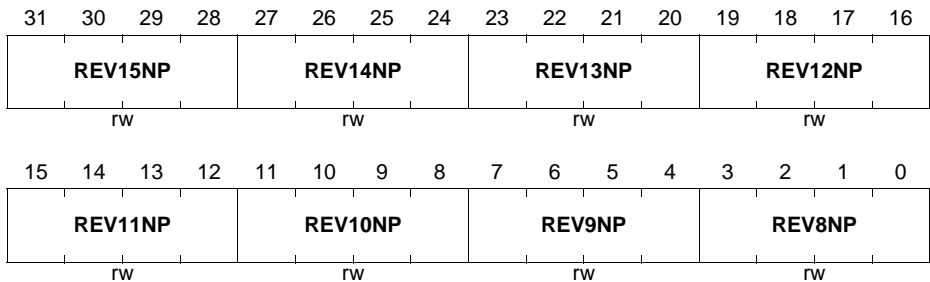
Field	Bits	Type	Description
REV0NP, REV1NP, REV2NP, REV3NP, REV4NP, REV5NP, REV6NP, REV7NP	[3:0], [7:4], [11:8], [15:12], [19:16], [23:20], [27:24], [31:28]	rw	<p>Service Request Node Pointer Result Event i</p> <p>Routes the corresponding event trigger to one of the service request lines (nodes).</p> <p>0000_BSelect service request line 0 of group x ... 0011_BSelect service request line 3 of group x 0100_BSelect shared service request line 0 ... 0111_BSelect shared service request line 3 1xxx_B Reserved</p> <p><i>Note: For shared service request lines see common groups in Table 19-10.</i></p>

GxREVNP1 (x = 0 - 1)

Result Event Node Pointer Register 1, Group x

$$(x * 0400_H + 05B4_H)$$

Reset Value: 0000 0000_H



Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
REV8NP, REV9NP, REV10NP, REV11NP, REV12NP, REV13NP, REV14NP, REV15NP	[3:0], [7:4], [11:8], [15:12], [19:16], [23:20], [27:24], [31:28]	rw	<p>Service Request Node Pointer Result Event i</p> <p>Routes the corresponding event trigger to one of the service request lines (nodes).</p> <p>0000_BSelect service request line 0 of group x ... 0011_BSelect service request line 3 of group x 0100_BSelect shared service request line 0 ... 0111_BSelect shared service request line 3 1xxx_B Reserved</p> <p><i>Note: For shared service request lines see common groups in Table 19-10.</i></p>

GLOBEVNP

Global Event Node Pointer Register

(0140_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	REV0NP			
r	r	r	r	r	r	r	r	r	r	r	r	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	SEV0NP			
r	r	r	r	r	r	r	r	r	r	r	r	rw			

Versatile Analog-to-Digital Converter (VADC)

Field	Bits	Type	Description
SEV0NP	[3:0]	rw	<p>Service Request Node Pointer Backgr. Source Routes the corresponding event trigger to one of the service request lines (nodes).</p> <p>0000_BSelect shared service request line 0 of common service request group 0</p> <p>...</p> <p>0011_BSelect shared service request line 3 of common service request group 0</p> <p>0100_BSelect shared service request line 0 of common service request group 1</p> <p>...</p> <p>0111_BSelect shared service request line 3 of common service request group 1</p> <p>1xxx_B Reserved</p> <p><i>Note: For shared service request lines see common groups in Table 19-10.</i></p>
0	[15:4]	r	Reserved, write 0, read as 0
REV0NP	[19:16]	rw	<p>Service Request Node Pointer Backgr. Result Routes the corresponding event trigger to one of the service request lines (nodes).</p> <p>0000_BSelect shared service request line 0 of common service request group 0</p> <p>...</p> <p>0011_BSelect shared service request line 3 of common service request group 0</p> <p>0100_BSelect shared service request line 0 of common service request group 1</p> <p>...</p> <p>0111_BSelect shared service request line 3 of common service request group 1</p> <p>1xxx_B Reserved</p> <p><i>Note: For shared service request lines see common groups in Table 19-10.</i></p>
0	[31:20]	r	Reserved, write 0, read as 0

Versatile Analog-to-Digital Converter (VADC)

GxSRACT (x = 0 - 1)

Service Request Software Activation Trigger, Group x

(x * 0400_H + 05C8_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	AS SR3	AS SR2	AS SR1	AS SR0	0	0	0	0	AG SR3	AG SR2	AG SR1	AG SR0
r	r	r	r	w	w	w	w	r	r	r	r	w	w	w	w

Field	Bits	Type	Description
AGSRy (y = 0 - 3)	y	w	Activate Group Service Request Node y 0 _B No action 1 _B Activate the associated service request line
0	[7:4]	r	Reserved, write 0, read as 0
ASSRy (y = 0 - 3)	8 + y	w	Activate Shared Service Request Node y 0 _B No action 1 _B Activate the associated service request line
0	[31:12]	r	Reserved, write 0, read as 0

19.14 Interconnects

This section describes the actual implementation of the ADC module into the XMC4300, i.e. the incorporation into the microcontroller system.

19.14.1 Product-Specific Configuration

The functional description describes the features and operating modes of the A/D Converters in a general way. This section summarizes the configuration that is available in this product (XMC4300).

Each converter group is equipped with a separate analog converter module and a dedicated analog input multiplexer.

Table 19-10 General Converter Configuration in the XMC4300

Converter Group	Input Channels	Channels with Alternate Reference	12-bit Performance	Common Service Request Group
G0	0 ... 7	7	Calibrated	C0
G1	0 ... 7	7	Calibrated	C0

Synchronization Groups in the XMC4300

The converter kernels in the XMC4300 can be connected to synchronization groups to achieve parallel conversion of several input channels.

Table 19-11 summarizes which kernels can be synchronized for parallel conversions.

Table 19-11 Synchronization Groups in the XMC4300

ADC Kernel	Synchr. Group	Master selected by control input Cix ¹⁾			
		CI0 ²⁾	CI1	CI2	CI3
ADC00	A	ADC00	ADC01	-	-
ADC01	A	ADC01	ADC00	-	-

1) The control input is selected by bitfield STSEL in register **GxSYNCTR (x = 0 - 1)**.

Select the corresponding ready inputs accordingly by bits EVALRx.

2) Control input CI0 always selects the own control signals of the corresponding ADC kernel. This selection is meant for the synchronization master or for stand-alone operation.

Versatile Analog-to-Digital Converter (VADC)

19.14.2 Analog Module Connections in the XMC4300

The VADC module accepts a number of analog input signals. The analog input multiplexers select the input channels to be converted from the signals available in this product.

The exact number of analog input channels and the available connection to port pins depend on the employed product type (see also [Table 19-10](#)). A summary of channels enclosing all versions of the XMC4300 can be found in [Table 19-12](#).

Input channels marked “PDD” provide a pull-down device for pull-down diagnostics.

Input channels marked “noAltref” cannot select the alternate reference voltage from channel 0 of the corresponding converter.

Table 19-12 Analog Connections in the XMC4300

Signal	Dir.	Source/Destin.	Description
V_{AREF}	I	VDDA	positive analog reference
V_{AGND}	I	VSSA	negative analog reference
G0CH0	I	P14.0	analog input channel 0 of group 0
G0CH1	I	P14.1	analog input channel 1 of group 0
G0CH2	I	P14.2	analog input channel 2 of group 0
G0CH3	I	P14.3	analog input channel 3 of group 0
G0CH4	I	P14.4	analog input channel 4 of group 0
G0CH5	I	P14.5	analog input channel 5 of group 0
G0CH6	I	P14.6	analog input channel 6 of group 0
G0CH7 (PDD)	I	P14.7	analog input channel 7 of group 0
G1CH0	I	P14.8	analog input channel 0 of group 1
G1CH1	I	P14.9	analog input channel 1 of group 1
G1CH2	I	P14.2	analog input channel 2 of group 1
G1CH3	I	P14.3	analog input channel 3 of group 1
G1CH4	I	P14.12	analog input channel 4 of group 1
G1CH5	I	P14.13	analog input channel 5 of group 1
G1CH6	I	P14.14	analog input channel 6 of group 1
G1CH7 (PDD)	I	P14.15	analog input channel 7 of group 1

Versatile Analog-to-Digital Converter (VADC)
19.14.3 Digital Module Connections in the XMC4300

The VADC module accepts a number of digital input signals and generates a number of output signals. This section summarizes the connection of these signals to other on-chip modules or to external resources via port pins.

Table 19-13 Digital Connections in the XMC4300

Signal	Dir.	Source/Destin.	Description
Gate Inputs for Each Group			
VADC.GxREQGTA	I	CCU40.ST3	Gating input A
VADC.GxREQGTB	I	CCU41.ST3	Gating input B
VADC.GxREQGTC	I	CCU40.SR0	Gating input C
VADC.GxREQGTD	I	CCU41.SR1	Gating input D
VADC.GxREQGTE	I	CCU80.ST3A	Gating input E
VADC.GxREQGTF	I	CCU80.ST3B	Gating input F
VADC.GxREQGTG	I	-	Gating input G
VADC.GxREQGTH	I	-	Gating input H
VADC.G0REQGTI	I (s)	DAC0.SGN	Gating input I
VADC.G1REQGTI	I (s)	DAC1.SGN	Gating input I
VADC.GxREQGTJ	I (s)	LEDTS.FN	Gating input J
VADC.G0REQGTK	I (s)	VADC.G1BFLOUT0	Gating input K
VADC.G1REQGTK	I (s)	VADC.G0BFLOUT0	Gating input K
VADC.G0REQGTL	I (s)	-	Gating input L
VADC.G1REQGTL	I (s)	VADC.G0SAMPLE ¹⁾	Gating input L
VADC.GxREQGTM	I	CCU80.SR0	Gating input M
VADC.GxREQGTN	I	CCU80.SR1	Gating input N
VADC.GxREQGTO	I	ERU1.PDOUT0	Gating input O
VADC.GxREQGTP	I	ERU1.PDOUT1	Gating input P
VADC.GxREQGTySEL	O	VADC.GxREQTRYP ²⁾	Selected gating signal of the respective source
Gate Inputs for Global Background Source			
VADC.BGREQGTA	I	CCU40.ST3	Gating input A, background source
VADC.BGREQGTB	I	CCU41.ST3	Gating input B, background source
VADC.BGREQGTC	I	CCU40.SR0	Gating input C, background source

Versatile Analog-to-Digital Converter (VADC)

Table 19-13 Digital Connections in the XMC4300 (cont'd)

Signal	Dir.	Source/Destin.	Description
VADC.BGREQGTD	I	CCU41.SR1	Gating input D, background source
VADC.BGREQGTE	I	CCU80.ST3A	Gating input E, background source
VADC.BGREQGTG	I	CCU80.ST3B	Gating input F, background source
VADC.BGREQGTG	I	-	Gating input G, background source
VADC.BGREQGTH	I	-	Gating input H, background source
VADC.BGREQGTI	I (s)	DAC0.SGN	Gating input I, background source
VADC.BGREQGTJ	I (s)	LEDTS.FN	Gating input J, background source
VADC.BGREQGTK	I (s)	VADC.G1BFLOUT0	Gating input K, background source
VADC.BGREQGTL	I (s)	-	Gating input L, background source
VADC.BGREQGTM	I	CCU80.SR0	Gating input M, background source
VADC.BGREQGTN	I	CCU80.SR1	Gating input N, background source
VADC.BGREQGTO	I	ERU1.PDOUT0	Gating input O, background source
VADC.BGREQGTP	I	ERU1.PDOUT1	Gating input P, background source
VADC.BGREQGTSE L	O	VADC.BGREQTRP ²)	Selected gating signal

Trigger Inputs for Each Group

VADC.GxREQTRA	I	CCU40.SR2	Trigger input A
VADC.GxREQTRB	I	CCU40.SR3	Trigger input B
VADC.GxREQTRC	I	CCU41.SR2	Trigger input C
VADC.GxREQTRD	I	CCU41.SR3	Trigger input D
VADC.GxREQTRE	I	-	Trigger input E
VADC.GxREQTRF	I	-	Trigger input F
VADC.GxREQTRG	I	ECAT.SYNC0	Trigger input G
VADC.GxREQTRH	I	ECAT.SYNC1	Trigger input H
VADC.GxREQTRI	I (s)	CCU80.SR2	Trigger input I
VADC.GxREQTRJ	I (s)	CCU80.SR3	Trigger input J
VADC.GxREQTRK	I (s)	-	Trigger input K
VADC.GxREQTRL	I (s)	-	Trigger input L
VADC.GxREQTRM	I	ERU1.IOUT0	Trigger input M
VADC.G0REQTRN	I	ERU1.IOUT1	Trigger input N
VADC.G1REQTRN	I	ERU1.IOUT1	Trigger input N

Versatile Analog-to-Digital Converter (VADC)
Table 19-13 Digital Connections in the XMC4300 (cont'd)

Signal	Dir.	Source/Destin.	Description
VADC.G0REQTRO	I	-	Trigger input O
VADC.G1REQTRO	I	-	Trigger input O
VADC.GxREQTRyP	I	VADC.GxREQGTyS EL ²⁾	Extend triggers to selected gating input of the respective source
VADC.GxREQTRyS EL	O	-	Selected trigger signal of the respective source

Trigger Inputs for Global Background Source

VADC.BGREQTRA	I	CCU40.SR2	Trigger input A, background source
VADC.BGREQTRB	I	CCU40.SR3	Trigger input B, background source
VADC.BGREQTRC	I	CCU41.SR2	Trigger input C, background source
VADC.BGREQTRD	I	CCU41.SR3	Trigger input D, background source
VADC.BGREQTRE	I	-	Trigger input E, background source
VADC.BGREQTRF	I	-	Trigger input F, background source
VADC.BGREQTRG	I	ECAT.SYNC0	Trigger input G, background source
VADC.BGREQTRH	I	ECAT.SYNC1	Trigger input H, background source
VADC.BGREQTRI	I (s)	CCU80.SR2	Trigger input I, background source
VADC.BGREQTRJ	I (s)	CCU80.SR3	Trigger input J, background source
VADC.BGREQTRK	I (s)	-	Trigger input K, background source
VADC.BGREQTRL	I (s)	-	Trigger input L, background source
VADC.BGREQTRM	I	ERU1.IOUT0	Trigger input M, background source
VADC.BGREQTRN	I	ERU1.IOUT1	Trigger input N, background source
VADC.BGREQTRO	I	-	Trigger input O, background source
VADC.BGREQTRP	I	VADC.BGREQGTS EL ²⁾	Extend triggers to selected gating input of the background source
VADC.BGREQTRSE L	O	-	Selected trigger signal of the background source

System-Internal Connections

VADC.G0SAMPLE ¹⁾	O	VADC.G1REQGTL	Indicates the input signal sample phase
VADC.G1SAMPLE	O	-	Indicates the input signal sample phase

Versatile Analog-to-Digital Converter (VADC)

Table 19-13 Digital Connections in the XMC4300 (cont'd)

Signal	Dir.	Source/Destin.	Description
VADC.G0ARBCNT	O	CCU40.IN3G	Outputs a (count) pulse for each arbiter round
VADC.G1ARBCNT	O	CCU41.IN3G	Outputs a (count) pulse for each arbiter round
VADC.GxSR0	O	NVIC, GPDMA	Service request 0 of group x
VADC.GxSR1	O	NVIC, GPDMA	Service request 1 of group x
VADC.GxSR2	O	NVIC, GPDMA	Service request 2 of group x
VADC.G0SR3	O	NVIC, GPDMA CCU80.IN0F	Service request 3 of group 0
VADC.G1SR3	O	NVIC, GPDMA	Service request 3 of group 1
VADC.C0SR0	O	NVIC, GPDMA ERU1.OGU01	Service request 0 of common block 0
VADC.C0SR1	O	NVIC, GPDMA ERU1.OGU11	Service request 1 of common block 0
VADC.C0SR2	O	NVIC, GPDMA ERU1.OGU21	Service request 2 of common block 0
VADC.C0SR3	O	NVIC, GPDMA ERU1.OGU31	Service request 3 of common block 0
VADC.EMUX00	O	GPIO	Control of external analog multiplexer interface 0
VADC.EMUX01	O	GPIO	
VADC.EMUX02	O	GPIO	
VADC.EMUX10	O	GPIO	Control of external analog multiplexer interface 1
VADC.EMUX11	O	GPIO	
VADC.EMUX12	O	GPIO	
CBFLOUT0	O	-	Common boundary flag output 0
CBFLOUT1	O	-	Common boundary flag output 1
CBFLOUT2	O	-	Common boundary flag output 2
CBFLOUT3	O	-	Common boundary flag output 3
VADC.G0BFLOUT0	O	VADC.G1REQGTK VADC.BGREQGTK CCU41.IN0L CCU80.IN0I	Boundary flag 0 output of group 0
VADC.G1BFLOUT0	O	VADC.G0REQGTK	Boundary flag 0 output of group 1

Versatile Analog-to-Digital Converter (VADC)

Table 19-13 Digital Connections in the XMC4300 (cont'd)

Signal	Dir.	Source/Destin.	Description
VADC.GxBFL0	O	-	Boundary flag 0 level of group x
VADC.GxBFSEL0	I	0	Boundary flag 0 (group x) source select
VADC.GxBFDAT0	I	0	Boundary flag 0 (group x) alternate data
VADC.G0BFLOUT1	O	VADC.G0REQGTK CCU41.IN2L CCU80.IN1I	Boundary flag 1 output of group 0
VADC.G1BFLOUT1	O	-	Boundary flag 1 output of group 1
VADC.GxBFL1	O	-	Boundary flag 1 level of group x
VADC.GxBFSEL1	I	0	Boundary flag 1 (group x) source select
VADC.GxBFDAT1	I	0	Boundary flag 1 (group x) alternate data
VADC.G0BFLOUT2	O	CCU41.IN3L CCU80.IN2I	Boundary flag 2 output of group 0
VADC.G1BFLOUT2	O	-	Boundary flag 2 output of group 1
VADC.GxBFL2	O	-	Boundary flag 2 level of group x
VADC.GxBFSEL2	I	0	Boundary flag 2 (group x) source select
VADC.GxBFDAT2	I	0	Boundary flag 2 (group x) alternate data
VADC.G0BFLOUT3	O	CCU80.IN3I ERU1.0B2 ERU1.2B2	Boundary flag 3 output of group 0
VADC.G1BFLOUT3	O	ERU1.1B2 ERU1.3B2	Boundary flag 3 output of group 1
VADC.GxBFL3	O	-	Boundary flag 3 level of group x
VADC.GxBFSEL3	I	0	Boundary flag 3 (group x) source select
VADC.GxBFDAT3	I	0	Boundary flag 3 (group x) alternate data

1) To use the SAMPLE output signals, enable them via register **OCS**.

2) Internal signal connection.



20 Digital to Analog Converter (DAC)

This chapter describes the two Digital to Analog Converter (DAC) channels available in the module.

20.1 Overview

The module consists of two separate 12-bit digital to analog converters (DACs). It converts two digital input signals into two analog voltage signal outputs at a maximum conversion rate of 5 MHz. The available design structure is based on a current steering architecture with internal reference generation and provides buffered voltage outputs. In order to reduce power consumption during inactive periods, a power down mode is available.

A built-in wave generator mode allows the CPU free generation of a selectable choice of wave forms. Alternatively values can be feed via CPU or DMA directly to one or both DAC channels. Additionally an offset can be added and the amplitude can be scaled. Several time trigger sources are possible.

20.1.1 Features

Analog features

- DAC resolution 12 bit;
- Conversion rate up to 5 MHz with reduced accuracy;
- Conversion rate up to 2 MHz at full accuracy;
- Maximum settling time of 2 μ s for a full scale 12-bit input code transition;
- Buffered voltage output;
- Direct drive of 5 kOhm / 50 pF terminated load;
- Segmented current steering architecture;
- Low glitch energy;
- Power down mode;
- DAC output and ADC input share the same analog input pin. ADC measurement is possible in parallel to DAC usage.
- V_{DDA} analog supply;

Digital features

- One Advanced Microcontroller Bus Architecture (AMBA) 32-bit AHB-Lite bus interface for data transfer and control of both DACs;
- Self triggered Direct Memory Access (DMA) handling capability with independent or simultaneous data handling for the two DAC channels (see [Section 20.2.4](#));
- First In First Out (FIFO) data buffers to allow a longer service request latency and to guarantee a continuous data transfer to the DACs (see [Section 20.2.4](#));

Digital to Analog Converter (DAC)

- Pattern generators available with freely programmable waveforms for both DACs (see [Section 20.2.5](#));
- Independent noise generators available for both DACs (see [Section 20.2.6](#));
- Data scaling by shift operation (multiplication and division by 2, 4, 8,..., 128) of the DACs' input data;
- Data offset value addition to the DACs input data;
- 8 selectable external trigger inputs;
- Internal integer clock divider for DAC trigger generation;
- Software trigger option;
- V_{DCC} digital supply;

20.1.2 Block Diagram

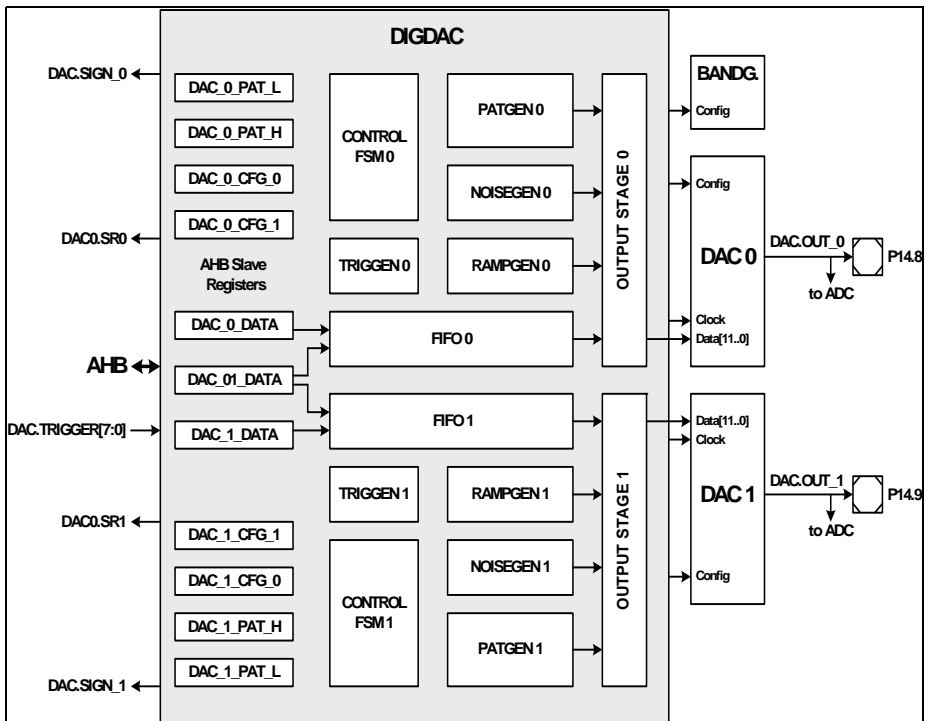


Figure 20-1 Block Diagram of DAC Module including Digdac Submodule

20.2 Operating Modes

The following chapters describe all the DAC's functional operating modes and how to use them. All used configuration parameters in this chapter are part of the registers described in [Section 20.6](#).

20.2.1 Hardware features

To facilitate the understanding of the different operating modes, a brief description of the supporting hardware features is given here:

Control and Data Registers

All control and data registers shown on the left hand side of [Figure 20-1](#) are described in [Section 20.6](#) in detail. They are connected to the AHB-Lite bus via an AHB-Lite slave interface. The interface also handles the necessary error response generation without introducing any latency.

Control Logic - Finite State Machines (FSM)

The two control finite state machines control all data processing modes of the DAC (see [Figure 20-1](#)). This means that they are responsible for the start and stop operating sequence, the service request generation for DMA handling for the so called data-mode and the control of the data FIFOs, the pattern generators, the noise generators and the ramp generators. Both FSMs are equivalent in structure and both are able to operate fully independent for the two DAC channels. For the simultaneous data-mode both FSMs are active, but only the service request signal of channel 0 (DAC0.Service Request(SR)0) should be evaluated by the DMA controller. Of course in that simultaneous data-mode the trigger source has to be the same for both channels.

20.2.1.1 Trigger Generators (TG)

The block diagram in [Figure 20-2](#) shows one of the two DAC's trigger generators.

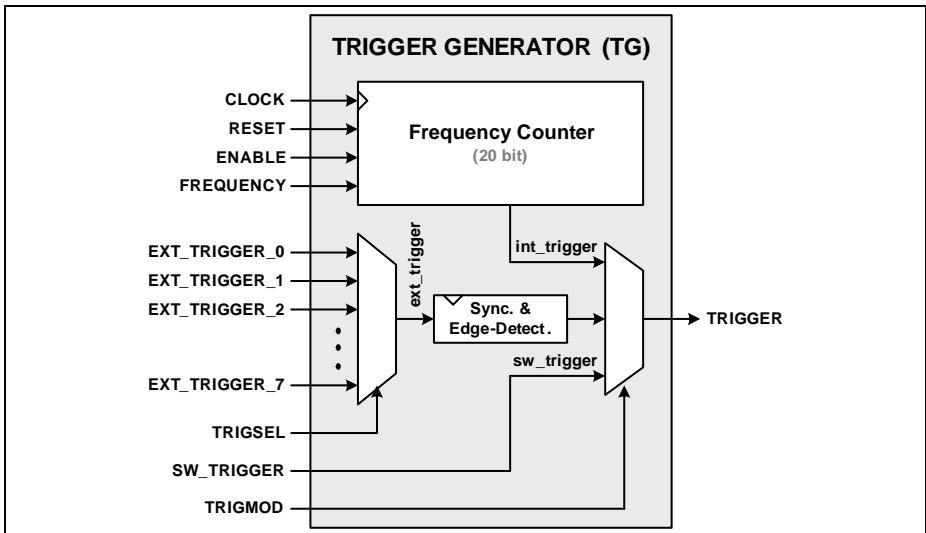


Figure 20-2 Trigger Generator Block Diagram

The TG consists of two multiplexers and one frequency counter. The first multiplexer selects between one of the eight external trigger sources whereas the second one enables switching between this selected external trigger source, an internally generated trigger and an additional software trigger input. The internal trigger is generated by the frequency counter which operates as a simple integer clock divider. So the internal trigger period can only be a multiple of the DACs system clock period. In order to guarantee correct operation of the analog part of the DAC the smallest frequency divider value is limited to 16 by hardware.

The external trigger inputs are synchronized to the system clock and only the rising edge is evaluated by the TG.

The software trigger input is connected to a register bit which is automatically cleared after it has been set to one. For this reason the output trigger of the TG is always a pulse with a pulse width of one system clock cycle for all three trigger possible modes.

20.2.1.2 Data FIFO buffer (FIFO)

The block diagram in [Figure 20-3](#) shows one of the two data FIFO buffers, one for each DAC.

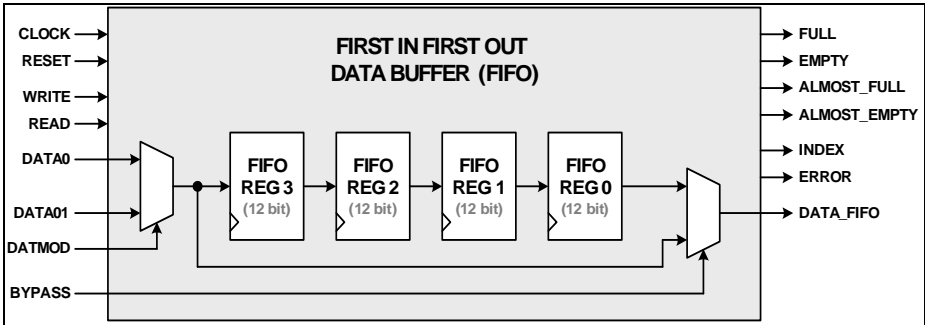


Figure 20-3 Data FIFO Block Diagram

This data FIFO buffer with four FIFO registers is introduced to allow a longer service request latency and to guarantee a continuous data processing for the DAC channels. It is used for DAC's so called data processing mode described in [Section 20.2.4](#). All FIFO's read and write operations are controlled by the corresponding Control FSM. A read operation is triggered by the chosen trigger and a write operation is initiated by an Advanced High-performance Bus (AHB) write operation to the currently used data register. The FIFO's status outputs named full, empty and index can be read by the software. A FIFO bypass used for all other DACs operating modes is also available.

20.2.1.3 Data output stage

The block diagram in [Figure 20-4](#) shows one of the two DACs' data output stages.

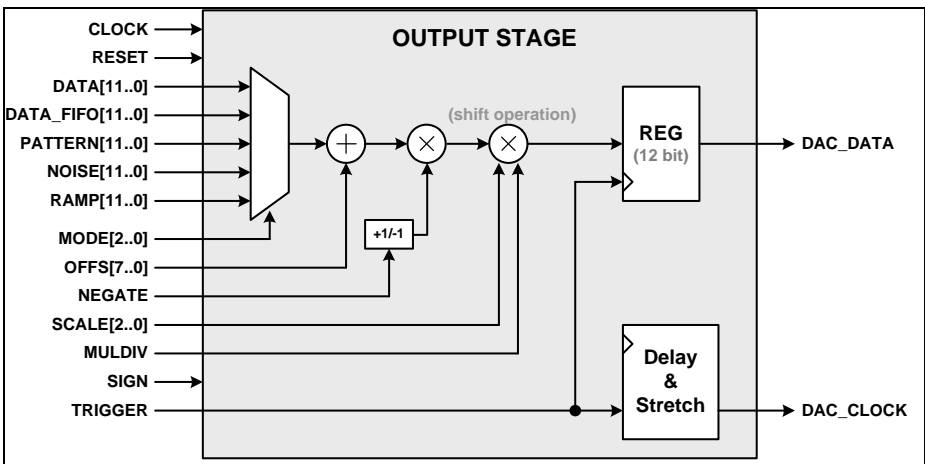


Figure 20-4 Data Output Stage Block Diagram

Digital to Analog Converter (DAC)

This output stage is the last element in the DAC's data path before the data is converted to the analog. It consists of a multiplexer, an adder, a multiplier, an output register and the generation of the DAC clock output.

The multiplexer selects between the five possible data sources and is programmed with the mode parameter. The adder stage gives the possibility to add an 8-bit offset value which is mainly needed for the PG mode in order to also process unsigned signal patterns to the DACs. In that case a certain offset value can be added to the signed output pattern values. The multiplier enables scaling by simple binary shifting of the data values. Therefore it allows multiplication and division by a programmed 2^n scale value. Additionally the output value can be negated, i.e. converted to its two's complement value. These operations are possible in all functional operating modes. The output register contains the final sample delivered together with the corresponding trigger to the analog converter.

The clock output for operating the analog part of the DAC is generated using the DAC's trigger generator (TG). For that purpose the TG's trigger output is delayed by four system clock periods and stretched to a high-length of eight system clock periods.

20.2.1.4 Pattern Generators (PG) - Waveform Generator

The block diagram in [Figure 20-5](#) shows one of the two DAC's pattern generators.

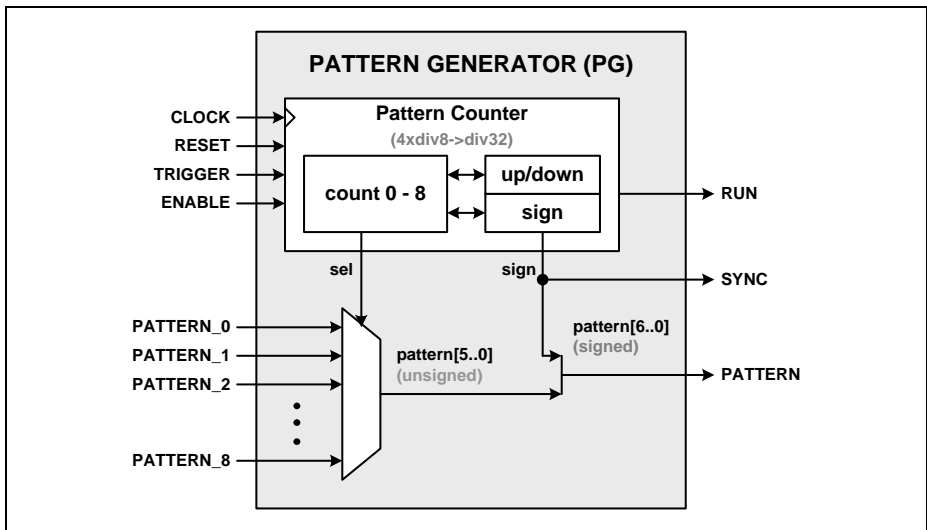


Figure 20-5 Pattern Generator Block Diagram

The nine pattern inputs on the left side of the PG block diagram are directly connected to the pattern registers described in [Section 20.6.3.4](#). The pattern registers contain only

Digital to Analog Converter (DAC)

one quarter of the actual programmed periodic pattern. The output of the pattern counter in the PG is used to select one of the nine input patterns. This pattern counter is an up-down counter with an additional sign output which is inverted every time the counter reaches zero. Since the sign information is concatenated with the currently selected pattern, it is possible to generate a complete pattern sequence for a full period of any $2^* \pi$ periodic waveform. For a detailed description how to operate the pattern generator please also refer to [Section 20.2.5](#).

20.2.1.5 Noise Generators (NG) - Pseudo Random Number Generator

The block diagram in [Figure 20-6](#) shows one of the two DAC's noise generators.

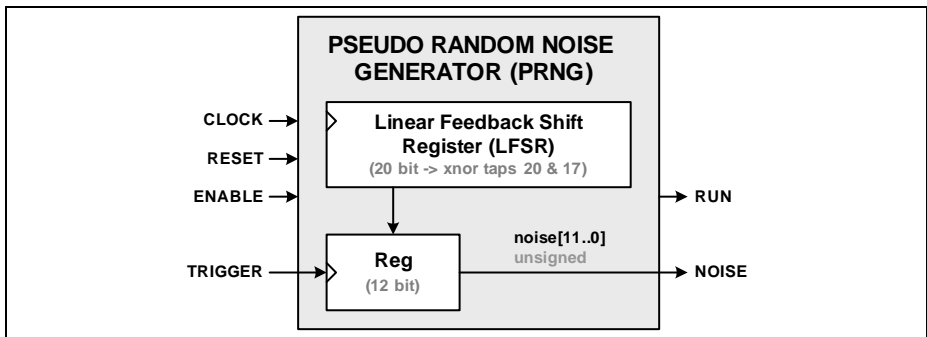


Figure 20-6 Noise Generator Block Diagram

The NG outputs a 12-bit pseudo random number. A 20-bit LFSR (linear feedback shift register) operating at the system clock frequency and a sample output register which is triggered by the NG's trigger inputs are used for this purpose. After enabling the NG, the LFSR is set back to its reset value and therefore it always starts outputting the same pseudo random number sequence.

20.2.1.6 Ramp Generators (RG)

The block diagram in [Figure 20-7](#) shows one of the two DAC's ramp generators.

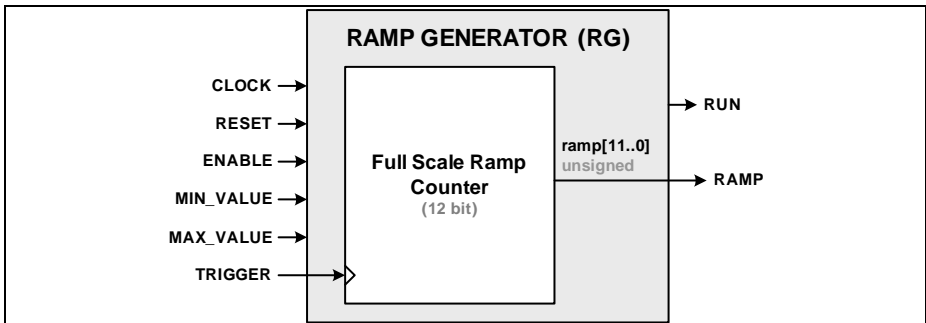


Figure 20-7 Ramp Generator Block Diagram

The ramp generator is basically an 12-bit up counter representing the full DAC range. If the RG is enabled it always starts at the programmed minimum value. The up-counting by ones is triggered by the selected trigger of the DAC channel. If the ramp counter reaches the programmed maximum value it restarts from the minimum value. This allows the generation of ramps within any desired value range and by variation of the trigger frequency also the ramp's slope can be modified.

20.2.2 Entering any Operating Mode

Before entering the desired operating mode with the **MODE** parameter, the corresponding analog DAC channel should be enabled with **ANAEN** and the startup time of the analog DAC channel should be considered.

Setting the **DATMOD** parameter to one enables simultaneous data processing. This means that both DAC channels use the same trigger source and both channels are always started and stopped synchronously. Hence the parameter setting of **MODE**, **TRIGMOD**, **TRIGSEL** and **FREQ** for DAC channel 0 is used for DAC channel 1 also.

20.2.3 Single Value Mode

By setting **MODE** to "single value mode", it is possible to convert only one single data value by the DACs. To start a conversion, a data value can be written to either **DATA0** or **DATA1** in **DAC0DATA** or **DAC1DATA** registers. This write operation itself then initiates a single trigger pulse and the value gets processed by DAC0 or DAC1. Only for this mode, no further external, internal or software trigger pulse is necessary. The DAC holds the processed value until a new value is written to **DATA0** or **DATA1**. This operating mode is intended for outputting static DAC output values. For processing sequential data streams in this "self triggered", single value mode, it is important not to exceed the maximum DAC data rate. If the **DATMOD** parameter is set to zero, data from the independent data registers **DAC0DATA** or **DAC1DATA** is processed. Whereas if

DATMOD is set to one, data from the simultaneous data register **DAC01DATA** is processed for both DACs.

20.2.4 Data Processing Mode

This operating mode is intended for continuous data processing from the system memory to DAC0 and/or DAC1. To enable it, the **MODE** parameter has to be set to data mode. Also, the desired trigger source has to be selected by setting **TRIGMOD**, **TRIGSEL** and **FREQ** in the configuration registers. The DAC can operate either with an internal generated trigger, one of the eight external trigger sources (see [Figure 20-2](#)) or the software trigger bit **SWTRIG**.

Simultaneous and independent Data Modes

With the **DATMOD** parameter, either the simultaneous or the independent “data mode” can be selected. In the simultaneous mode, both DACs receive their data from the same register **DAC01DATA**. In the independent “data mode” DAC0 gets its data from **DAC0DATA** and DAC1 from **DAC1DATA**. The two data paths are shown in [Figure 20-8](#) and also in [Figure 20-3](#). Both DAC channels can activate a service request output signal to trigger a DMA channel, if they are configured in this mode and if the corresponding **SREN** bit is set to one. The service requests are DAC0.SR0 for the DAC0 channel and DAC0.SR1 for the DAC1 channel. For the simultaneous mode either DAC0.SR0 or DAC0.SR1 can be used by the DMA controller.

Start-Stop Operation

Before the DAC is started in “data mode”, all configuration registers DACx_CFG_x should be set according to the desired processing mode (see [Section 20.6.3.2](#)). Once this has been done, the DAC can be started by setting the **MODE** parameter to “data mode”. The control FSM will then start its operation until it reaches the run status indicated by the read parameter **RUN**. The run state can be left either by an operation error or by setting the **MODE** parameter to “disable DAC” again. An operation error can be a FIFO overflow or a FIFO underflow (see [Figure 20-3](#) and [Figure 20-8](#)).

20.2.4.1 FIFO Data Handling

[Figure 20-8](#) shows the data handling for the FIFO of the DAC0 channel. Certainly the same structure also exists for the DAC1 channel (see [Figure 20-1](#) and [Figure 20-3](#) also). The data word **DATA0** from either data register **DAC0DATA** or **DAC01DATA** on the left hand side in [Figure 20-8](#) is loaded into one of the FIFO buffers registers. The load position in the FIFO depends on its actual filling level represented by the read parameters **FIFOIND**, **FIFOEMP** and **FIFOFUL**. If the FIFO is empty, **FIFOIND** = 0. If it is full, **FIFOIND** = 3. If a trigger occurs and the FIFO is not empty, the data is shifted to the next register (from left to right). At the same time, a service request (DAC0.SR0 or DAC0.SR1) is initiated in order to fill up the FIFO again. The service request should end

Digital to Analog Converter (DAC)

with a write operation to **DAC0DATA** or **DAC01DATA**. This write operation itself then triggers a write from the data registers to the FIFO buffer registers. If the FIFO stores only one last element (**FIFOIND** = 0 and **FIFOEMP** = 0) and a trigger has occurred, a service request is initiated and additionally **FIFOEMP** is set to 1. On the other hand, if there is only one last free register in the FIFO (**FIFOIND** = 2) and a write operation has been initiated, the **FIFOFUL** bit is set to 1. All the control signals for the FIFO handling are generated by the DAC's control FSM. This includes filling up the FIFO when "data mode" is entered and emptying the FIFO when leaving "data mode".

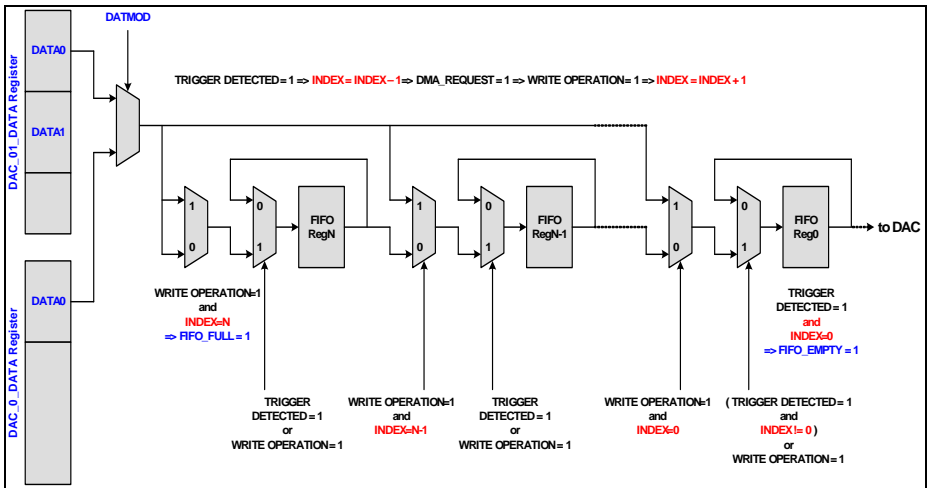


Figure 20-8 Data Handling for the FIFO Buffer

20.2.5 Pattern Generation Mode

This chapter describes the operation of the pattern generator. This mode is used to output a pattern or waveform to the DACs and it is activated by setting the **MODE** configuration parameter to "patgen mode".

Like in the "data mode" (see [Section 20.2.4](#)), the DAC in "patgen mode" can operate either with an internally generated trigger, one of the eight external trigger sources or a software trigger bit.

The desired pattern or waveform is freely programmable with the 5-bit parameters **PAT0** to **PAT8**. The pattern registers contain only one quadrant of the full waveform. The other quadrants of the $2 \cdot \pi$ periodic odd function are generated out of the first one with the help of a counter. The counter generates also the sign bit of the output pattern, therefore the 6-bit output signed values are in the range of -31 to +31.

In order to use the full range of the DAC in signed mode, a scaling by 32 by setting **SCALE** to "101" in the output stage is necessary. If the DAC should output a full range

Digital to Analog Converter (DAC)

pattern in unsigned mode, it is also possible to add an offset value programmed with **OFFS** to the output stage before doing the scaling. All the control signals for the pattern generators are generated by the DAC's control FSM.

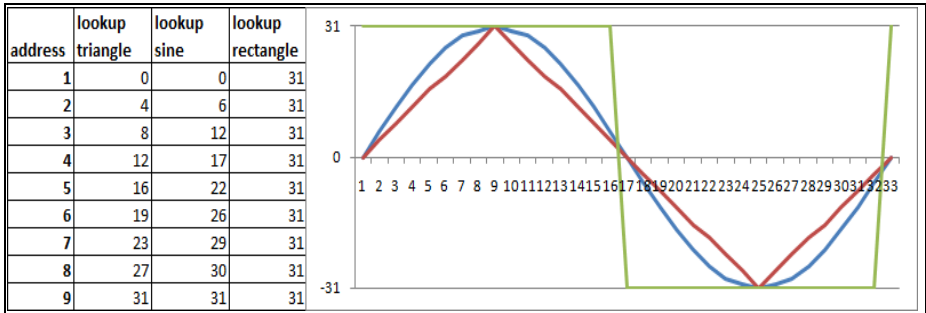


Figure 20-9 Example 5-bit Patterns and their corresponding Waveform Output

Figure 20-9 gives examples of lookup table entries for triangular, sine and rectangular pattern. These values can be programmed to **PAT0** to **PAT8** in order to get the corresponding waveforms at the DAC's output like shown in the chart on the right hand side. If the pattern generation is restarted / enabled again, it always starts with the first value of the first quarter of the actual programmed pattern (positive value and up-counting). The current sign information of the generated pattern is one of the DAC's system on chip outputs (see **Section 20.7.2.3**) and can be enabled using the parameter **SIGNEN**.

20.2.6 Noise Generation Mode

A 20-bit linear feedback shift register (LFSR) is used to produce a pseudo random number. In order to enable the noise generator, the **MODE** parameter has to be set to "noise mode". The LFSR itself runs with the system clock. The 12-bit random numbers is sampled with the preselected trigger source using **TRIGMOD**, **TRIGSEL** and optionally also **FREQ** or **SWTRIG** into an output register. The 12-bit values can be interpreted as signed or unsigned values. By setting the **MODE** parameter to "disable DAC" again, the noise generation stops and the DAC holds its last processed value. **Figure 20-10** shows an example pseudo random noise output. After restarting the noise generation mode, the LFSR is set back to its reset value and therefore it always starts outputting the same pseudo random number sequence.

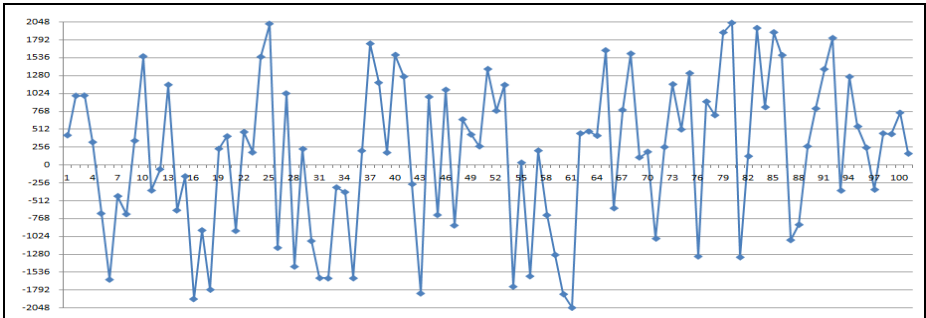


Figure 20-10 Signed 12-bit pseudo random Noise Example Output

20.2.7 Ramp Generation Mode

A 12-bit ramp counter is also part of the DAC. It is activated by setting the **MODE** parameter to “ramp mode”. The trigger source can be selected using **TRIGMOD**, **TRIGSEL** and optionally also **FREQ** or **SWTRIG**. The ramp counter starts at a programmed start value and each trigger pulse increments the counter by one. The start values are programmable via **DATA0** for DAC channel 0 and **DATA1** for DAC channel 1 of the independent data registers. The stop values are programmable via **DATA0** or **DATA1** of the simultaneous data register. If the ramp counter reaches its stop value, it restarts from the start value with the next trigger pulse. This allows the generation of ramps within any desired value range. The ramp’s slope can be modified by varying the trigger frequency. **Figure 20-11** shows two examples of ramp generation output waveforms.

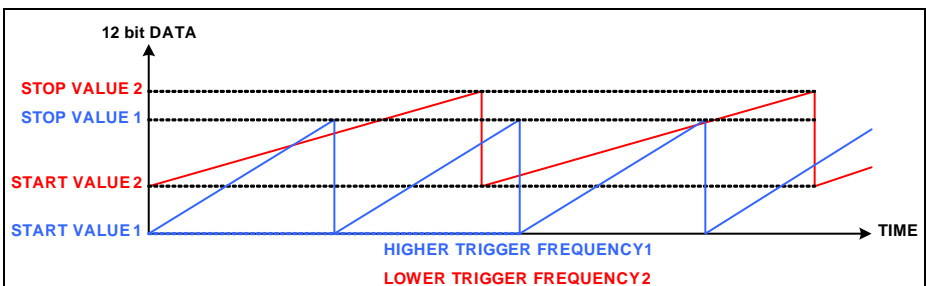


Figure 20-11 Unsigned 12-bit Ramp Generator Example Output

20.3 Service Request Generation

Service Requests are available in **Data Processing Mode** only.

20.4 Power, Reset and Clock

As long as **ANAEN** is set to default value “standby”, the corresponding DAC channel stays in power down mode, and its output is floating.

By setting **ANAEN** to “enable DAC”, the analog output is starting up; the max. possible startup time (t_{STARTUP}) is specified in the Data Sheet. Note, that in case the output is triggered before the startup time expired, the programmed value is not passed to the output driver. The DAC output will then continue to drive the default value, till the first trigger event after elapsing the startup time. ¹⁾

The DAC module reset is shared with the peripheral bus reset line, as well as the module clock is shared with the peripheral bus clock line.

With **FREQ** the clock divider ratio of the internal trigger generator is set.

20.5 Initialization

A feasible initialization sequence of the DAC reads as follows:

1st Step: De-assert the reset of DAC module by setting DACRS bit in PRCLR1 register and disable gating by setting DAC bit in CGATCLR1 register.

2nd Step: Write the DACxCFG0 register values. Here you select the operating mode of the corresponding DAC channel by writing the **MODE** field, e.g. Patgen mode. By setting or clearing the **SIGN** bit, the choice between signed and unsigned input data format is made.

In the same step service request generation can be enabled with the **SREN** bit, as well as sign output with **SIGNEN** bit. Also the frequency divider of the internal trigger generator can be set up by writing the **FREQ** field.

3rd Step: Write the DACxCFG1 register values. Here you select the trigger source by writing the **TRIGMOD** field, e.g. software trigger. The DAC channel output is enabled by setting the **ANAEN** bit. You also need to choose now your values for **SCALE**, **MULDIV**, **OFFS**, and **DATMOD** fields.

4th Step: Configure the chosen data source. E.g. in case you selected Patgen mode, the pattern must be defined by programming DACxPATL and DACxPATH registers.

5th Step: E.g. in case software trigger is selected, the runtime code is responsible for generating the trigger signals by setting **SWTRIG** bit inside DACxCFG1 register. In this case, please mind the startup time (t_{STARTUP}) after setting **ANAEN** bit (see also [Section 20.4](#)).

C code example

The C code of this example is given below:

1) Please note, that the RUN status bit is set by hardware, before the startup time has expired.

Digital to Analog Converter (DAC)

```

void DAC_init()
{
//RESET MODULE
    SCU_RESET->PRCLR1 |= 0x00000020; //De-asserts reset for
VADC module  SCU_CLK->CGATCLR1  |=  SCU_CLK_CGATCLR1_DAC_Msk;
//Disable Gating
//DAC0 CONFIGURATION AS PATTERN GENERATOR
    DAC->DAC0CFG0=0x20300FFF;           //Pattern gen enable,
Signal enabled and Frequency
    DAC->DAC0CFG1=0x010405FD;           //Enable AN, Software
trigger
                                                    // Offset and
Scale divider set up.

    DAC->DAC0PATL=0x3568B0C0;           //Sinus waveform
configuration
    DAC->DAC0PATH=0x00007FDD;           //Sinus waveform
configuration
//For triangle waveform use:
    //DAC->DAC0PATL=0x27062080;         //Triangle waveform
configuration
    //DAC->DAC0PATH=0x00007F77;         //Triangle waveform
configuration

//DAC1 CONFIGURATION AS RAMP GENERATOR
    DAC->DAC1CFG0=0x20500000;           //Ramp Mode, Signal enabled
and Frequency
    DAC->DAC1CFG1=0x01000000;           //Enable AN. No offset or
scaling. Internal Trigger

    DAC->DAC1DATA=0x00000003F;         //Start value
    DAC->DAC01DATA=0x0AFF0000;         //Stop value
}
-----
//For SW trigger in runtime code (after waiting startup time):

    DAC->DAC0CFG1|=0x00010000;         //Software Trigger of DAC0

```

20.6 Registers

20.6.1 Address Map

The DAC is available at the following base address:

Table 20-1 Registers Address Space

Module	Base Address	End Address	Note
DAC	4801 8000 _H	4801 BFFF _H	16 kB

20.6.2 Register Overview

Table 20-2 shows all registers required for the operation of the DAC module:

Table 20-2 Register Overview of DAC

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
ID	Module Identification Register	000 _H	U, PV, 32	U, PV, 32	Page 20-16
DAC0CFG0	DAC0 Configuration Register Number 0	004 _H	U, PV, 32	U, PV, 32	Page 20-17
DAC0CFG1	DAC0 Configuration Register Number 1	008 _H	U, PV, 32	U, PV, 32	Page 20-18
DAC1CFG0	DAC1 Configuration Register Number 0	00C _H	U, PV, 32	U, PV, 32	Page 20-20
DAC1CFG1	DAC1 Configuration Register Number 1	010 _H	U, PV, 32	U, PV, 32	Page 20-22
DAC0DATA	Data Register for DAC0 for Independent Data Mode	014 _H	U, PV, 32	U, PV, 32	Page 20-24
DAC1DATA	Data Register for DAC1 for Independent Data Mode	018 _H	U, PV, 32	U, PV, 32	Page 20-25
DAC01DATA	Data Register for DAC0 and DAC1 for Simultaneous Data Mode	01C _H	U, PV, 32	U, PV, 32	Page 20-25
DAC0PATL	Lower Samples of Pattern for DAC0 PATGEN	020 _H	U, PV, 32	U, PV, 32	Page 20-26

Digital to Analog Converter (DAC)

Table 20-2 Register Overview of DAC (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
DAC0PATH	Higher Samples of Pattern for DAC0 PATGEN	024 _H	U, PV, 32	U, PV, 32	Page 20-27
DAC1PATL	Lower Samples of Pattern for DAC1 PATGEN	028 _H	U, PV, 32	U, PV, 32	Page 20-27
DAC1PATH	Higher Samples of Pattern for DAC1 PATGEN	02C _H	U, PV, 32	U, PV, 32	Page 20-28

1) The absolute register address is calculated as follows:
Module Base Address + Offset Address (shown in this column)

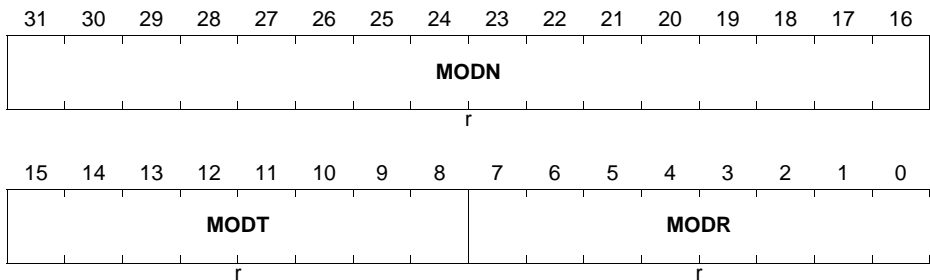
20.6.3 Register Description

20.6.3.1 DAC_ID Register

The DAC module identification register contains the XMC4000 ID code.

DAC_ID

Module Identification Register (000_H) **Reset Value: 00A5 C0XX_H**



Field	Bits	Type	Description
MODR	[7:0]	r	Module Revision MOD_REV defines the module revision number. The value of a module revision starts with 01 _H (first rev.).
MODT	[15:8]	r	Module Type This bit field is C0 _H . It defines the module as a 32-bit module.

Digital to Analog Converter (DAC)

Field	Bits	Type	Description
MODN	[31:16]	r	Module Number For the DAC this bit field is A5 _H

20.6.3.2 DAC Configuration Registers

The DAC configuration registers contain all the necessary bits to set DAC0 and DAC1 in the desired operating mode and to start and stop conversions.

DAC0CFG0

DAC0 Configuration Register 0 (004_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RUN	SRE N	SIGN EN	NEG ATE	FIFO FUL	FIFO EMP	FIFOIND	SIGN	MODE			FREQ				
rh	rw	rw	r	rh	rh	rh	rw	rw			rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FREQ															
rw															

Field	Bits	Type	Description
FREQ	[19:0]	rw	Integer Frequency Divider Value <ul style="list-style-type: none"> 0 to 16: divide by 16 16 to 2²⁰-1 divide by FREQ
MODE	[22:20]	rw	Enables and Sets the Mode for DAC0 000 _B disable/switch-off DAC 001 _B Single Value Mode 010 _B Data Mode 011 _B Patgen Mode 100 _B Noise Mode 101 _B Ramp Mode 110 _B na 111 _B na
SIGN	23	rw	Selects Between Signed and Unsigned DAC0 Mode 0 _B DAC expects unsigned input data 1 _B DAC expects signed input data
FIFOIND	[25:24]	rh	Current write position inside the data FIFO

Digital to Analog Converter (DAC)

Field	Bits	Type	Description
FIFOEMP	26	rh	Indicate if the FIFO is empty 0 _B FIFO not empty 1 _B FIFO empty
FIFOFUL	27	rh	Indicate if the FIFO is full 0 _B FIFO not full 1 _B FIFO full
NEGATE	28	rw	Negates the DAC0 output 0 _B DAC output not negated 1 _B DAC output negated Negation means the DAC value is converted to its two's complement value.
SIGNEN	29	rw	Enable Sign Output of DAC0 Pattern Generator 0 _B Disable 1 _B Enable
SREN	30	rw	Enable DAC0 service request interrupt generation 0 _B disable 1 _B enable
RUN	31	rh	RUN indicates the current DAC0 operation status 0 _B DAC0 channel disabled 1 _B DAC0 channel in operation RUN is set/cleared by hardware.

DAC0CFG1

DAC0 Configuration Register 1

(008_H)

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFCFG1			0			ANA EN	ANACFG				TRIGMOD		SWT RIG		
rw			r			rw	rw				rw		rw/h		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT MOD		TRIGSEL			OFFS						MUL DIV		SCALE		
rw		rw			rw						rw		rw		

Digital to Analog Converter (DAC)

Field	Bits	Type	Description
SCALE	[2:0]	rw	<p>Scale value for up- or downscale of the DAC0 input data in steps by the power of 2 (=shift operation)</p> <p>000_B no shift = multiplication/division by 1 001_B shift by 1 = multiplication/division by 2 010_B shift by 2 = multiplication/division by 4 011_B shift left by 3 = multiplication/division by 8 100_B shift left by 4 = multiplication/division by 16 101_B shift left by 5 = multiplication/division by 32 110_B shift left by 6 = multiplication/division by 64 111_B shift left by 7 = multiplication/division by 128</p>
MULDIV	3	rw	<p>Switch between up- and downscale of the DAC0 input data values</p> <p>0_B downscale = division (shift SCALE positions to the right) 1_B upscale = multiplication (shift SCALE positions to the left)</p>
OFFS	[11:4]	rw	<p>8-bit offset value addition</p> <p>e.g.: PATGEN output is a sine wave -31 to +31 and OFFS = 31 => the DAC0 input data will be a sine wave with an amplitude between 0 and 62. Depending on the SIGN bit this value is interpreted as signed or unsigned.</p>
TRIGSEL	[14:12]	rw	<p>Selects one of the eight external trigger sources for DAC0</p>
DATMOD	15	rw	<p>Switch between independent or simultaneous DAC mode and select the input data register for DAC0 and DAC1</p> <p>0_B independent data handling - process data from DATA0 register (bits 11:0) to DAC0 and data from DATA1 register (bits 11:0) to DAC1 1_B simultaneous data handling - process data from DAC01 register to both DACs (bits 11:0 to DAC0 and bits 23:12 to DAC1).</p> <p>Trigger setting and MODE parameter for DAC0 is used for DAC1 also if DATMOD is set to 1 = simultaneous data mode!</p>

Digital to Analog Converter (DAC)

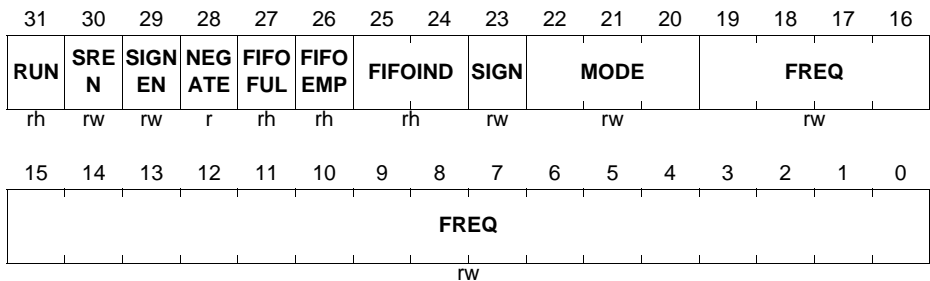
Field	Bits	Type	Description
SWTRIG	16	rwh	Software Trigger Triggers DAC channel 0 if TRIGMOD is set to 10. Setting the bit to 1 generates one trigger pulse. The bit is cleared (set to 0) automatically. If DATMOD is set to simultaneous data mode this bit is used for both DAC channels (see DATMOD parameter).
TRIGMOD	[18:17]	rw	Select the trigger source for channel 0 00 _B internal Trigger (integer divided clock - see FREQ parameter) 01 _B external Trigger (preselected trigger by TRIGSEL parameter) 10 _B software Trigger (see SWTRIG parameter) 11 _B reserved
ANACFG	[23:19]	rw	DAC0 analog configuration/calibration parameters reserved for future use
ANAEN	24	rw	Enable analog DAC for channel 0 0 _B DAC0 is set to standby (analog output only) 1 _B enable DAC0 (analog output only)
0	[27:25]	r	Reserved Read as 0; Should be written with 0.
REFCFG	[31:28]	rw	Lower 4 band-gap configuration/calibration parameters reserved for future use

DAC1CFG0

DAC1 Configuration Register 0

(00C_H)

Reset Value: 0000 0000_H



Digital to Analog Converter (DAC)

Field	Bits	Type	Description
FREQ	[19:0]	rw	Integer Frequency Divider Value <ul style="list-style-type: none"> 0 to 16: divide by 16 16 to 2²⁰-1 divide by FREQ FREQ for DAC1 is not applicable if DATMOD is set to 1 = simultaneous data mode.
MODE	[22:20]	rw	Enables and sets the Mode for DAC1 <p>000_B disable/switch-off DAC 001_B Single Value Mode 010_B Data Mode 011_B Patgen Mode 100_B Noise Mode 101_B Ramp Mode 110_B na 111_B na</p> MODE for DAC1 is not applicable if DATMOD is set to 1 = simultaneous data mode.
SIGN	23	rw	Selects between signed and unsigned DAC1 mode <p>0_B DAC expects unsigned input data 1_B DAC expects signed input data</p>
FIFOIND	[25:24]	rh	Current write position inside the data FIFO
FIFOEMP	26	rh	Indicate if the FIFO is empty <p>0_B FIFO not empty 1_B FIFO empty</p>
FIFOFUL	27	rh	Indicate if the FIFO is full <p>0_B FIFO not full 1_B FIFO full</p>
NEGATE	28	rw	Negates the DAC1 output <p>0_B DAC output not negated 1_B DAC output negated</p> Negation means the DAC value is converted to its two's complement value.
SIGNEN	29	rw	Enable sign output of DAC1 pattern generator <p>0_B disable 1_B enable</p>

Digital to Analog Converter (DAC)

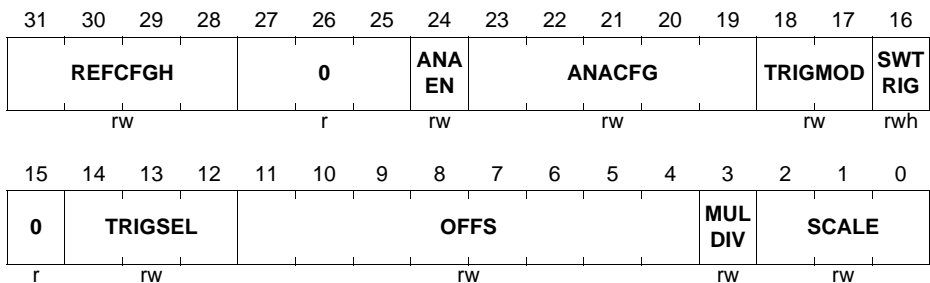
Field	Bits	Type	Description
SREN	30	rw	Enable DAC1 service request interrupt generation 0 _B disable 1 _B enable
RUN	31	rh	RUN indicates the current DAC1 operation status 0 _B DAC1 channel disabled 1 _B DAC1 channel in operation RUN is set/cleared by hardware.

DAC1CFG1

DAC1 Configuration Register 1

(010_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
SCALE	[2:0]	rw	Scale value for up- or downscale of the DAC1 input data in steps by the power of 2 (=shift operation) 000 _B no shift = multiplication/division by 1 001 _B shift by 1 = multiplication/division by 2 010 _B shift by 2 = multiplication/division by 4 011 _B shift left by 3 = multiplication/division by 8 100 _B shift left by 4 = multiplication/division by 16 101 _B shift left by 5 = multiplication/division by 32 110 _B shift left by 6 = multiplication/division by 64 111 _B shift left by 7 = multiplication/division by 128

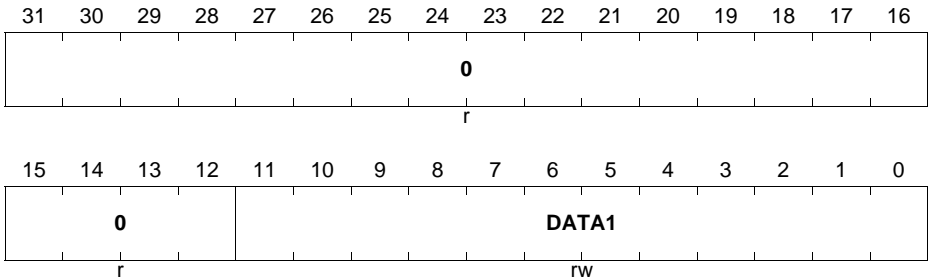
Digital to Analog Converter (DAC)

Field	Bits	Type	Description
MULDIV	3	rw	<p>Switch between up- and downscale of the DAC1 input data values</p> <p>0_B downscale = division (shift SCALE positions to the right)</p> <p>1_B upscale = multiplication (shift SCALE positions to the left)</p>
OFFS	[11:4]	rw	<p>8-bit offset value addition</p> <p>e.g.: PATGEN output is a sine wave -31 to +31 and OFFS = 31 => the DAC1 input data will be a sine wave with an amplitude between 0 and 62. Depending on the SIGN bit this value is interpreted as signed or unsigned.</p>
TRIGSEL	[14:12]	rw	<p>Selects one of the eight external trigger sources for DAC1</p> <p>TRIGSEL for DAC1 is not applicable if DATMOD is set to 1 = simultaneous data mode.</p>
0	15	r	<p>Reserved</p> <p>Read as 0; Should be written with 0.</p>
SWTRIG	16	rwh	<p>Software Trigger</p> <p>Triggers DAC channel 1 if TRIGMOD is set to 10. Setting the bit to 1 generates one trigger pulse. The bit is cleared (set to 0) automatically. If DATMOD is set to simultaneous data mode (see DATMOD parameter) this bit is not applicable and the SWTRIG bit from channel 0 is used for channel 1 also.</p>
TRIGMOD	[18:17]	rw	<p>Select the trigger source for channel 1</p> <p>00_B internal Trigger (integer divided clock - see FREQ parameter)</p> <p>01_B external Trigger (preselected trigger by TRIGSEL parameter)</p> <p>10_B software Trigger (see SWTRIG parameter)</p> <p>11_B reserved</p>
ANACFG	[23:19]	rw	<p>DAC1 analog configuration/calibration parameters</p> <p>reserved for future use</p>
ANAEN	24	rw	<p>Enable analog DAC for channel 1</p> <p>0_B DAC1 is set to standby (analog output only)</p> <p>1_B enable DAC1 (analog output only)</p>

Digital to Analog Converter (DAC)

DAC1DATA

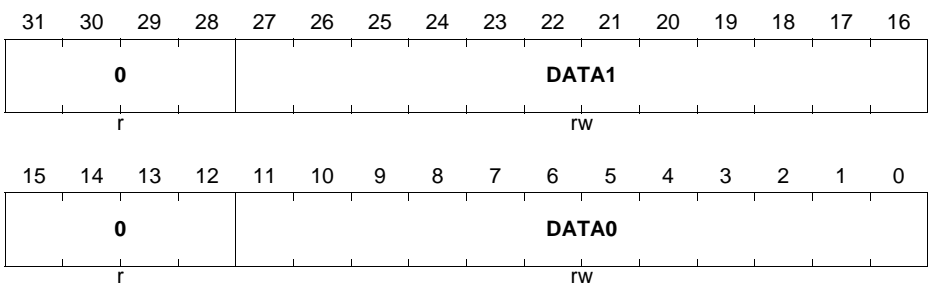
DAC1 Data Register (018_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DATA1	[11:0]	rw	DAC1 Data Bits Used as DAC1 data value and as counter start value in ramp generation mode
0	[31:12]	r	Reserved Read as 0; Should be written with 0.

DAC01DATA

DAC01 Data Register (01C_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DATA0	[11:0]	rw	DAC0 Data Bits Used as DAC0 data value and as counter stop value in ramp generation mode

Digital to Analog Converter (DAC)

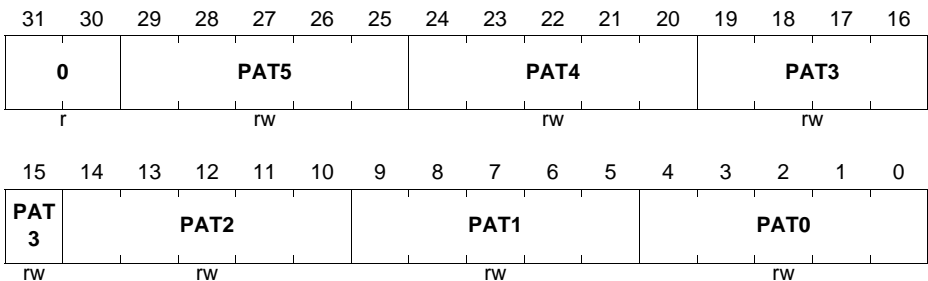
Field	Bits	Type	Description
0	[15:12]	r	Reserved Read as 0; Should be written with 0.
DATA1	[27:16]	rw	DAC1 Data Bits Used as DAC1 data value and as counter stop value in ramp generation mode
0	[31:28]	r	Reserved Read as 0; Should be written with 0.

20.6.3.4 DAC Pattern Registers

The DAC pattern registers contain the waveform patterns for the pattern generators of DAC0 and DAC1.

DAC0PATL

DAC0 Lower Pattern Register (020_H) Reset Value: 3568 B0C0_H



Field	Bits	Type	Description
PAT0	[4:0]	rw	Pattern Number 0 for PATGEN of DAC0
PAT1	[9:5]	rw	Pattern Number 1 for PATGEN of DAC0
PAT2	[14:10]	rw	Pattern Number 2 for PATGEN of DAC0
PAT3	[19:15]	rw	Pattern Number 3 for PATGEN of DAC0
PAT4	[24:20]	rw	Pattern Number 4 for PATGEN of DAC0
PAT5	[29:25]	rw	Pattern Number 5 for PATGEN of DAC0
0	[31:30]	r	Reserved Read as 0; Should be written with 0

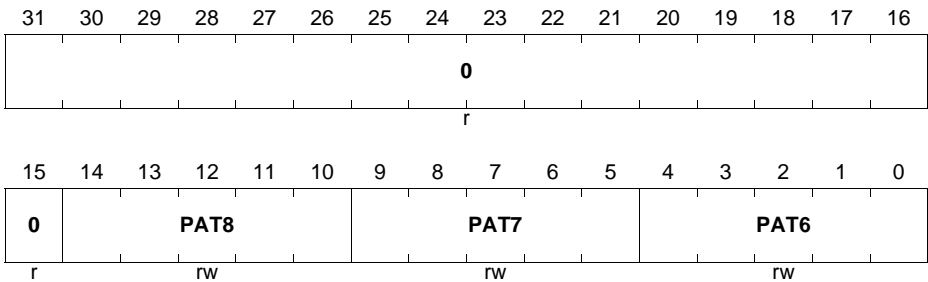
Digital to Analog Converter (DAC)

DAC0PATH

DAC0 Higher Pattern Register

(024_H)

Reset Value: 0000 7FDD_H



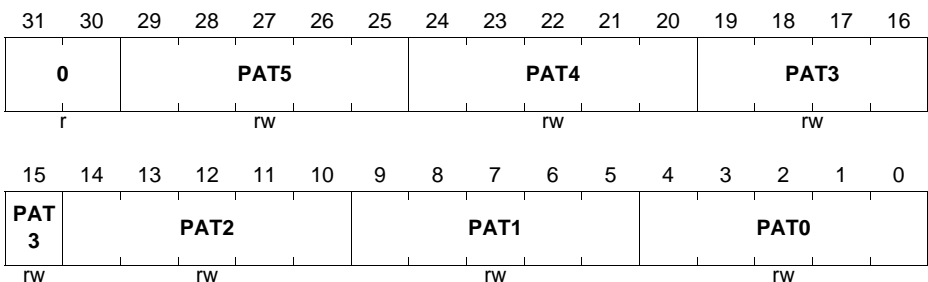
Field	Bits	Type	Description
PAT6	[4:0]	rw	Pattern Number 6 for PATGEN of DAC0
PAT7	[9:5]	rw	Pattern Number 7 for PATGEN of DAC0
PAT8	[14:10]	rw	Pattern Number 8 for PATGEN of DAC0
0	[31:15]	r	Reserved Read as 0; Should be written with 0.

DAC1PATL

DAC1 Lower Pattern Register

(028_H)

Reset Value: 3568 B0C0_H



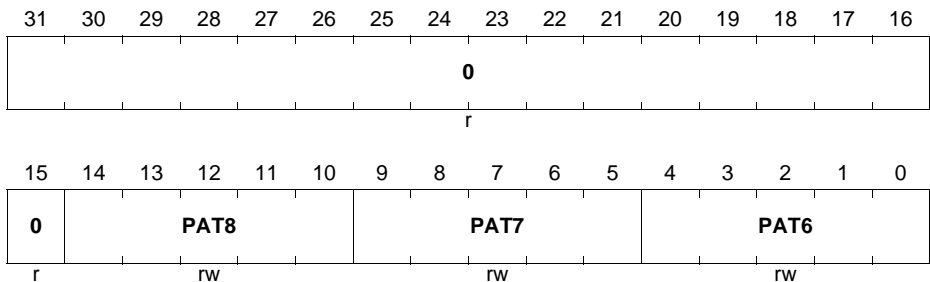
Field	Bits	Type	Description
PAT0	[4:0]	rw	Pattern Number 0 for PATGEN of DAC1
PAT1	[9:5]	rw	Pattern Number 1 for PATGEN of DAC1

Digital to Analog Converter (DAC)

Field	Bits	Type	Description
PAT2	[14:10]	rw	Pattern Number 2 for PATGEN of DAC1
PAT3	[19:15]	rw	Pattern Number 3 for PATGEN of DAC1
PAT4	[24:20]	rw	Pattern Number 4 for PATGEN of DAC1
PAT5	[29:25]	rw	Pattern Number 5 for PATGEN of DAC1
0	[31:30]	r	Reserved Read as 0; Should be written with 0.

DAC1PATH

DAC1 Higher Pattern Register (02C_H) **Reset Value: 0000 7FDD_H**



Field	Bits	Type	Description
PAT6	[4:0]	rw	Pattern Number 6 for PATGEN of DAC1
PAT7	[9:5]	rw	Pattern Number 7 for PATGEN of DAC1
PAT8	[14:10]	rw	Pattern Number 8 for PATGEN of DAC1
0	[31:15]	r	Reserved Read as 0; Should be written with 0.

20.7 Interconnects

20.7.1 Analog Connections

The analog interface lines of the DAC are listed below:

Table 20-3 Analog Connections

Input/Output	I/O	Connected To	Descriptions
DAC.OUT_0	O	P14.8	Analog output of channel 0
DAC.OUT_1	O	P14.9	Analog output of channel 1

20.7.2 Digital Connections

The DAC has the following system level connections to other modules:

20.7.2.1 Service Request Connections

Two service requests DAC.SR0 and DAC.SR1 are used for simultaneous and independent data mode. DAC.SR1 can be enabled on DMA channel 2 and DAC.SR0 can be enabled on DMA channel 3.

Table 20-4 Service Request Connections

Input/Output	I/O	Connected To	Descriptions
DAC.SR0	O	NVIC GPDMA	Service request
DAC.SR1	O	NVIC GPDMA	Service request

20.7.2.2 Trigger Connections

The eight trigger inputs are connected to the following sources:

Table 20-5 Trigger Connections

Input/Output	I/O	Connected To	Descriptions
DAC.TRIGGER[0]	I	CCU80.SR1	Trigger
DAC.TRIGGER[1]	I	reserved	Trigger
DAC.TRIGGER[2]	I	CCU40.SR1	Trigger
DAC.TRIGGER[3]	I	CCU41.SR1	Trigger
DAC.TRIGGER[4]	I	Port	Trigger
DAC.TRIGGER[5]	I	Port	Trigger
DAC.TRIGGER[6]	I	U0C0.DX1INS	Trigger
DAC.TRIGGER[7]	I	U1C0.DX1INS	Trigger

20.7.2.3 Synchronization Interface of the Pattern Generator

The interface consists of only two output signals called “DAC.SIGN_0” and “DAC.SIGN_1”. They are generated by the pattern generator of the two DAC channels and represent the actual sign information of the processed signal waveform converted by the DACs (see [Figure 20-5](#)).

Table 20-6 Pattern Generator Synchronization Connections

Input/Output	I/O	Connected To	Descriptions
DAC.SIGN_0	O	VADC.G0REQGTI VADC.BGREQGTI ERU1.0A3	
DAC.SIGN_1	O	VADC.G1REQGTI ERU1.2A3	

Industrial Control Peripherals

21 Capture/Compare Unit 4 (CCU4)

The CCU4 peripheral is a major component for systems that need general purpose timers for signal monitoring/conditioning and Pulse Width Modulation (PWM) signal generation. Power electronic control systems like switched mode power supplies or uninterruptible power supplies, can easily be implemented with the functions inside the CCU4 peripheral.

The internal modularity of CCU4, translates into a software friendly system for fast code development and portability between applications.

Table 21-1 Abbreviations table

PWM	Pulse Width Modulation
CCU4x	Capture/Compare Unit 4 module instance x
CC4y	Capture/Compare Unit 4 Timer Slice instance y
ADC	Analog to Digital Converter
POSIF	Position Interface peripheral
SCU	System Control Unit
f_{ccu4}	CCU4 module clock frequency
f_{tclk}	CC4y timer clock frequency

Note: A small “y” or “x” letter in a register indicates an index

21.1 Overview

Each CCU4 module is comprised of four identical 16 bit Capture/Compare Timer slices, CC4y. Each timer slice can work in compare mode or in capture mode. In compare mode one compare channel is available while in capture mode, up to four capture registers can be used in parallel.

Each CCU4 module has four service request lines and each timer slice contains a dedicated output signal, enabling the generation of up to four independent PWM signals. Straightforward timer slice concatenation is also possible, enabling up to 64 bit timing operations. This offers a flexible frequency measurement, frequency multiplication and pulse width modulation scheme.

A programmable function input selector for each timer slice, that offers up to nine functions, discards the need of complete resource mapping due to input ports availability.

A built-in link between the CCU4 and several other modules enable flexible digital motor control loops implementation, e.g. with Hall Sensor monitoring or direct coupling with Encoders.

21.1.1 Features

CCU4 module features

Each CCU4 represents a combination of four timer slices, that can work independently in compare or capture mode. Each timer slice has a dedicated output for PWM signal generation.

All four CCU4 timer slices, CC4y, are identical in terms of available functions and operating modes. Avoiding this way the need of implementing different software routines, depending on which resource of CCU4 is used.

A built-in link between the four timer slices is also available, enabling this way a simplified timer concatenation and sequential operations.

General Features

- 16 bit timer cells
- capture and compare mode for each timer slice
 - four capture registers in capture mode
 - one compare channel in compare mode
- programmable low pass filter for the inputs
- built-in timer concatenation
 - 32, 48 or 64 bit width
- shadow transfer for the period and compare values
- programmable clock prescaler
- normal timer mode
- gated timer mode
- three counting schemes
 - center aligned
 - edge aligned
 - single shot
- PWM generation
- TRAP function
- start/stop can be controlled by external events
- counting external events
- four dedicated service request lines per CCU4

Additional features

- external modulation function
- load controlled by external events
- dithering PWM
- floating point pre scaler
- output state override by an external event
- suitable and flexible connectivity to several modules:
 - motor and power conversion applications
 - high number of signal conditioning possibilities

CCU4 features vs. applications

On [Table 21-2](#) a summary of the major features of the CCU4 unit mapped with the most common applications.

Table 21-2 Applications summary

Feature	Applications
Four independent timer cells	Independent PWM generation: <ul style="list-style-type: none"> • Multiple buck/boost converter control (with independent frequencies) • Different modes of operation for each timer, increasing the resource optimization • Up to 2 Half-Bridges control • multiple Zero Voltage Switch (ZVS) converter control with easy link to the ADC channels.
Concatenated timer cells	Easy to configure timer extension up to 64 bit: <ul style="list-style-type: none"> • High dynamic trigger capturing • High dynamic signal measurement
Dithering PWM	Generating a fractional PWM frequency or duty cycle: <ul style="list-style-type: none"> • To avoid big steps on frequency or duty cycle adjustment in slow control loop applications • Increase the PWM signal resolution over time
Floating prescaler	Automated control signal measurement: <ul style="list-style-type: none"> • decrease SW activity for monitoring signals with high or unknown dynamics • emulating more than a 16 bit timer for system control
Up to 9 functions via external signals for each timer	Flexible resource optimization: <ul style="list-style-type: none"> • The complete set of external functions is always available • Several arrangements can be done inside a CCU4, e.g., one timer working in capture mode and one working in compare

Table 21-2 Applications summary (cont'd)

Feature	Applications
4 dedicated service request lines	Specially developed for: <ul style="list-style-type: none"> • generating interrupts for the microprocessor • flexible connectivity between peripherals, e.g. ADC triggering.
Linking with other modules	Flexible profiles for: <ul style="list-style-type: none"> • Hall Sensor feedback/monitoring • Motor Encoders feedback/monitoring • PWM parallel modulation • Flexible signal conditioning

21.1.2 Block Diagram

Each CCU4 timer slice can operate independently from the other slices for all the available modes. Each timer slice contains a dedicated input selector for functions linked with external events and has a dedicated compare output signal, for PWM signal generation.

The built-in timer concatenation is only possible with adjacent slices, e.g. CC40/CC41. Combinations for slice concatenations like, CC40/CC42 or CC40/CC43 are not possible.

The individual service requests for each timer slice (four per slice) are multiplexed into four module service requests lines, [Figure 21-1](#).

Capture/Compare Unit 4 (CCU4)

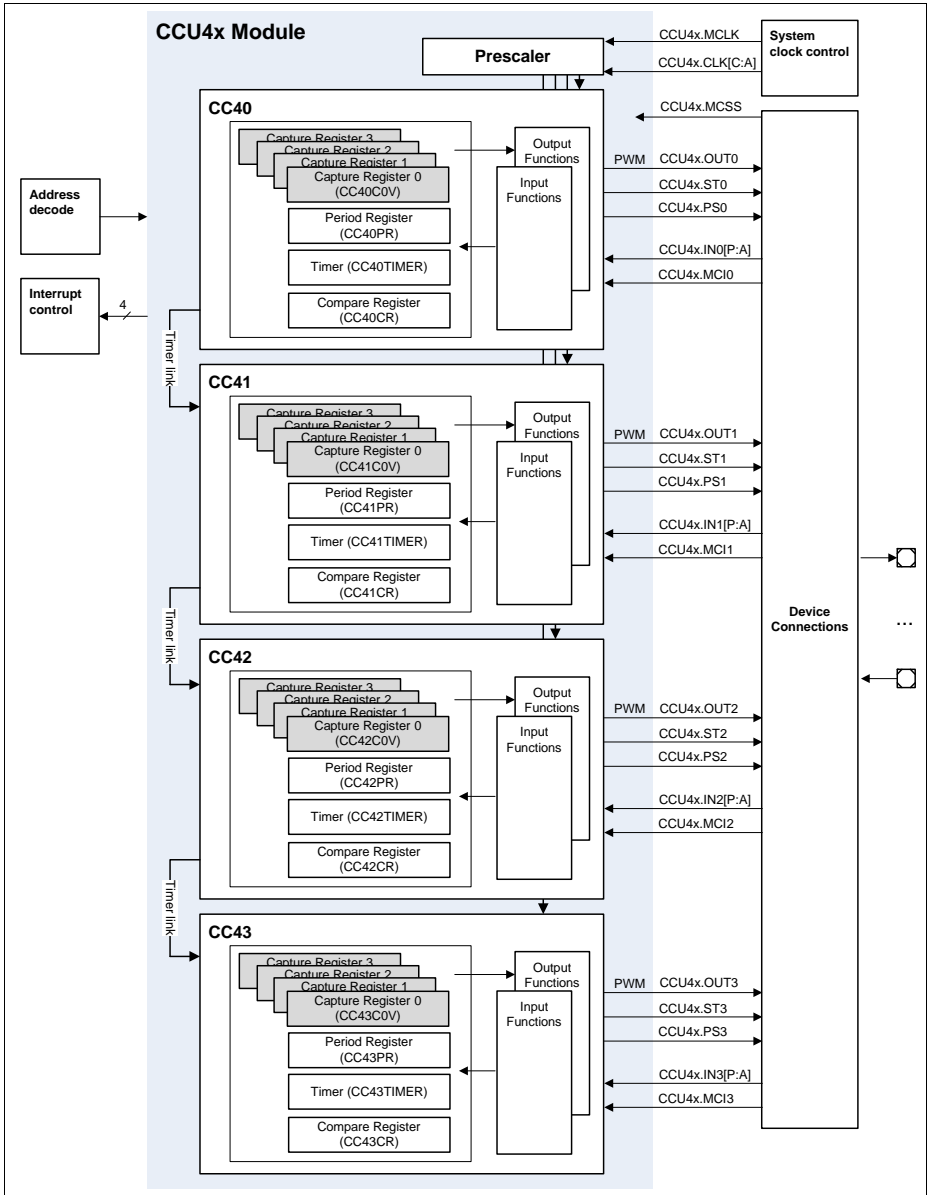


Figure 21-1 CCU4 block diagram

21.2 Functional Description

The following sections describe the complete set of functions and usability of the CCU4 peripheral.

In each figure several registers may be depicted to indicate controlability or configurability. These registers follow the description given in [Figure 21-2](#). One should also note that indexing in a register can be done via the non capital y, x or n.

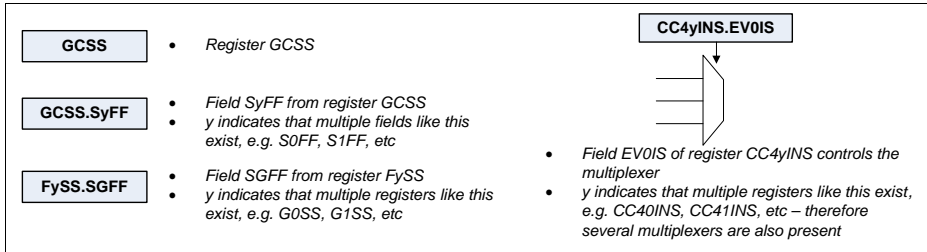


Figure 21-2 Register description in figures (example)

21.2.1 CC4y Overview

The input path of a CCU4 slice is comprised of a selector ([Section 21.2.2](#)) and a connection matrix unit ([Section 21.2.3](#)). The output path contains a service request control unit, a timer concatenation unit and two units that control directly the state of the output signal for each specific slice (for TRAP and modulation handling), see [Figure 21-3](#).

The timer core is built of a 16 bit counter one period and one compare register in compare mode, or up to four capture registers in capture mode.

In compare mode the period register sets the maximum counting value while the compare channel is controlling the ACTIVE/PASSIVE state of the dedicated comparison slice output.

Capture/Compare Unit 4 (CCU4)

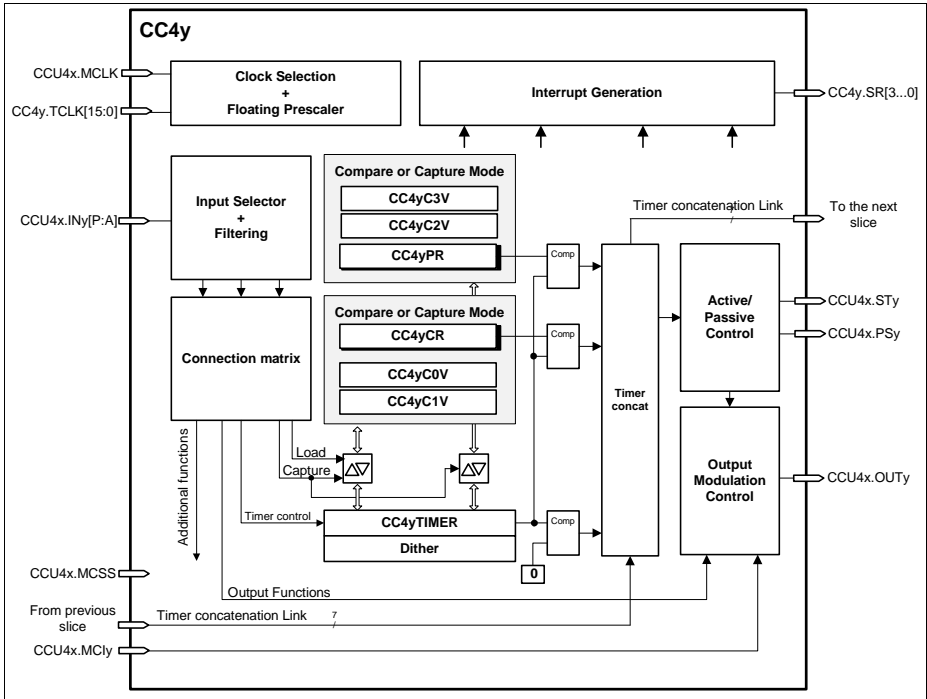


Figure 21-3 CCU4 slice block diagram

Each CCU4 slice, with the exception of the first, contains six dedicated inputs outputs that are used for the built-in timer concatenation functionality.

Inputs and outputs that are not seen at the CCU4 boundaries have a nomenclature of CC4y.<name>, whilst CCU4 module inputs and outputs are described as CCU4x.<signal_name>y (indicating the variable y the object slice).

Table 21-3 CCU4 slice pin description

Pin	I/O	Description
CCU4x.MCLK	I	Module clock
CC4y.TCLK[15:0]	I	Clocks from the pre scaler
CCU4x.INy[P:A]	I	Slice functional inputs (used to control the functionality throughout slice external events)
CCU4x.MCly	I	Multi Channel mode input
CCU4x.MCSS	I	Multi Channel shadow transfer trigger

Table 21-3 CCU4 slice pin description (cont'd)

Pin	I/O	Description
CC4y.SR[3...0]	O	Slice service request lines
CC4x.STy	O	Slice comparison status value
CCU4x.PSy	O	Multi channel pattern update trigger
CCU4x.OUTy	O	Slice dedicated output pin

Note:

3. The status bit outputs of the Kernel, CCU4x.STy, are extended for one more kernel clock cycle.
4. The Service Request signals at the output of the kernel are extended for one more kernel clock cycle.
5. The maximum output signal frequency of the CCU4x.STy outputs is module clock divided by 4.

The slice timer, can count up or down depending on the selected operating mode. A direction flag holds the actual counting direction.

The timer is connected to two stand alone comparators, one for the period match and one for a compare match. The registers used for period match and comparison match can be programmed to serve as capture registers enabling sequential capture capabilities on external events.

In normal edge aligned counting scheme, the counter is cleared to 0000_H each time that matches the period value defined in the period register. In center aligned mode, the counter direction changes from 'up counting' to 'down counting' after reaching the period value. Both period and compare registers have an aggregated shadow register, which enables the update of the PWM period and duty cycle on the fly.

A single shot mode is also available, where the counter stops after it reaches the value set in the period register.

The start and stop of the counter can be controlled via software access or by a programmable input pin.

Functions like, load, counting direction (up/down), TRAP and output modulation can also be controlled with external events, see [Section 21.2.3](#).

21.2.2 Input Selector

The first unit of the slice input path, is used to select which inputs are used to control the available external functions.

Inside this block the user also has the possibility to perform a low pass filtering of the signals and selecting the active edge(s) or level of the external event, see [Figure 21-4](#).

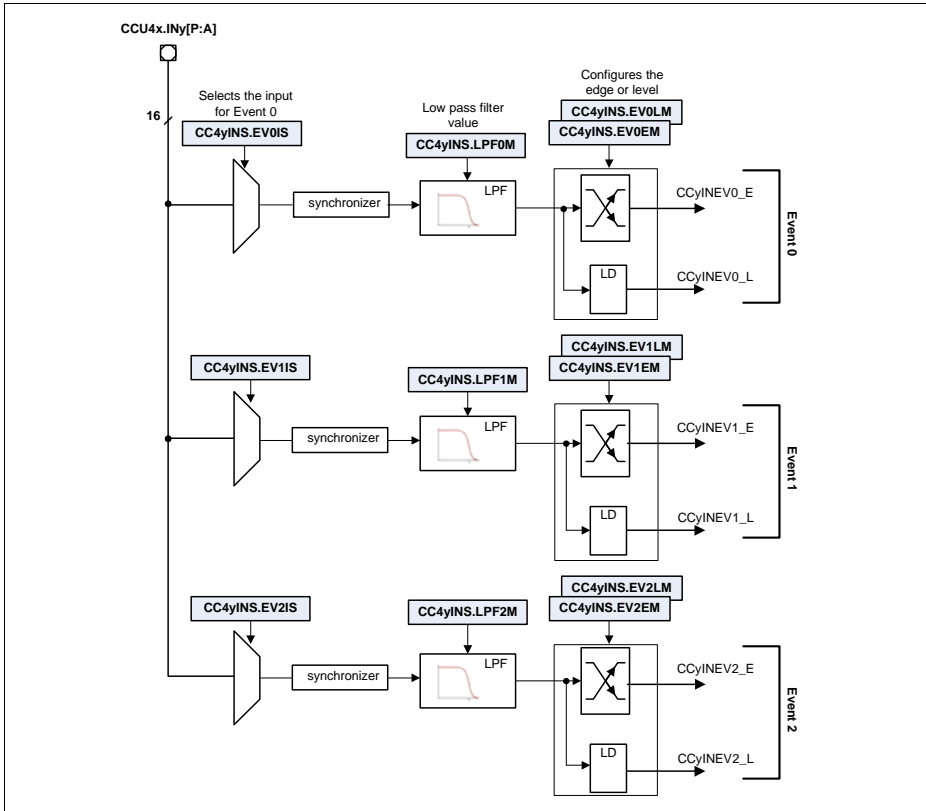


Figure 21-4 Slice input selector diagram

The user has the possibility of selecting any of the CCU4x.INy[P:A] inputs as the source of an event.

At the output of this unit we have a user selection of three events, that were configured to be active at rising, falling or both edges, or level active. These selected events can then be mapped to several functions.

Notice that each decoded event contains two outputs, one edge active and one level active, due to the fact that some functions like counting, capture or load are edge sensitive events while, timer gating or up down counting selection are level active.

21.2.3 Connection Matrix

The connection matrix maps the events coming from the input selector to several user configured functions, [Figure 21-5](#). The following functions can be enabled on the connection matrix:

Table 21-4 Connection matrix available functions

Function	Brief description	Map to figure Figure 21-5
Start	Edge signal to start the timer	CCystrt
Stop	Edge signal to stop the timer	CCystp
Count	Edge signal used for counting events	CCycnt
Up/down	Level signal used to select up or down counting direction	CCyupdn
Capture 0	Edge signal that triggers a capture into the capture registers 0 and 1	CCycapt0
Capture 1	Edge signal that triggers a capture into the capture register 2 and 3	CCycapt1
Gate	Level signal used to gate the timer clock	CCygate
Load	Edge signal that loads the timer with the value present at the compare register	CCyload
TRAP	Level signal used for fail-safe operation	CCytrap
Modulation	Level signal used to modulate/clear the output	CCymod
Status bit override	Status bit is going to be overridden with an input value	CCyoval for the value CCyoset for the trigger

Inside the connection matrix we also have a unit that performs the built-in timer concatenation. This concatenation enables a completely synchronized operation between the concatenated slices for timing operations and also for capture and load actions. The timer slice concatenation is done via the [CC4yCMC.TCE](#) bitfield. For a complete description of the concatenation function, please address [Section 21.2.9](#).

Capture/Compare Unit 4 (CCU4)

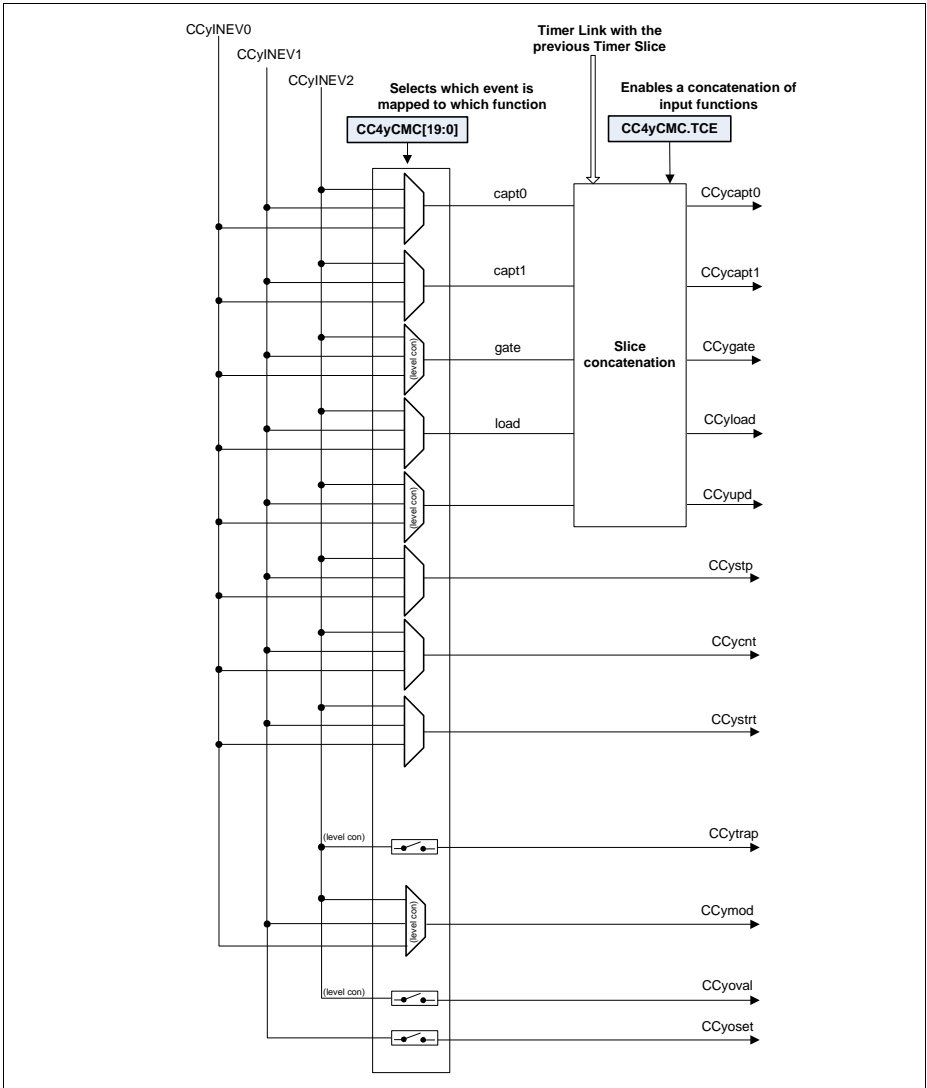


Figure 21-5 Slice connection matrix diagram

21.2.4 Starting/Stopping the Timer

Each timer slice contains a run bit register that indicates the actual status of the timer, **CC4yTCST.TRB**. The start and stop of the timer can be done via software access or can be controlled directly by external events, see **Figure 21-6**.

Selecting an external signal that acts as a start trigger does not force the user to use an external stop trigger and vice versa.

Selecting the single shot mode, imposes that after the counter reaches the period value the run bit, **CC4yTCST.TRB**, is going to be cleared and therefore the timer is stopped.

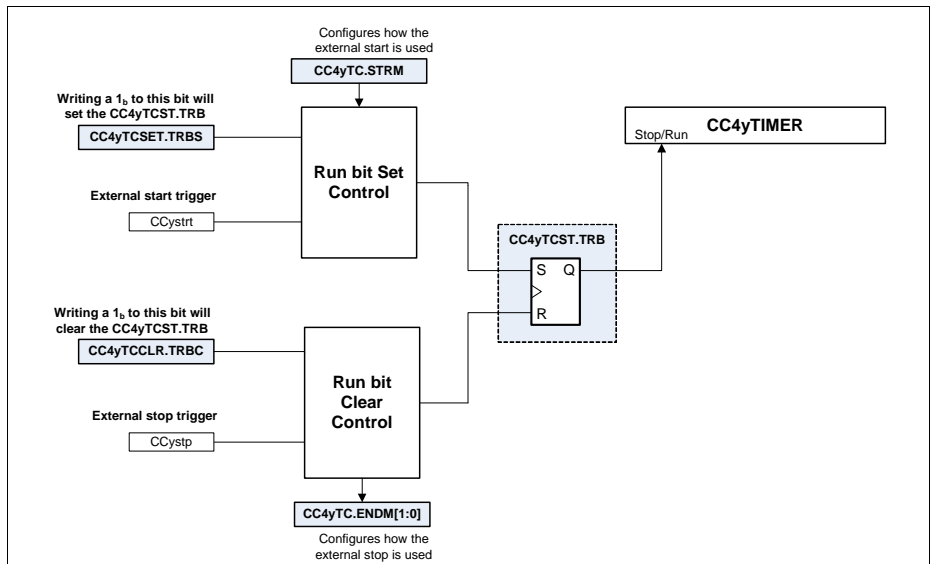


Figure 21-6 Timer start/stop control diagram

One can use the external stop signal to perform the following functions (configuration via **CC4yTC.ENDM**):

- Clear the run bit (stops the timer) - default
- Clear the timer (to 0000_H) but it does not clear the run bit (timer still running)
- Clear the timer and the run bit

One can use the external start to perform the following functions (configuration via **CC4yTC.STRM**):

- Start the timer (resume operation)
- Clear and start the timer

The set (start the timer) of the timer run bit, always has priority over a clear (stop the timer).

Capture/Compare Unit 4 (CCU4)

To start multiple CCU4 timers at the same time/synchronously one should use a dedicated input as external start (see [Section 21.2.7.1](#) for a description how to configure an input as start function). This input should be connected to all the Timers that need to be started synchronously (see [Section 21.8](#) for a complete list of module connections), [Figure 21-7](#).

For starting the timers synchronously via software there is a dedicated input signal, controlled by the SCU (System Control Unit), that is connected to all the CCU4 timers. This signal should then be configured as an external start signal (see [Section 21.2.7.1](#)) and then the software must write a $1_B/0_B$ (depending on the configuration of the external start signal) to the specific bitfield of the CCUCON register (this register is described on the SCU chapter).

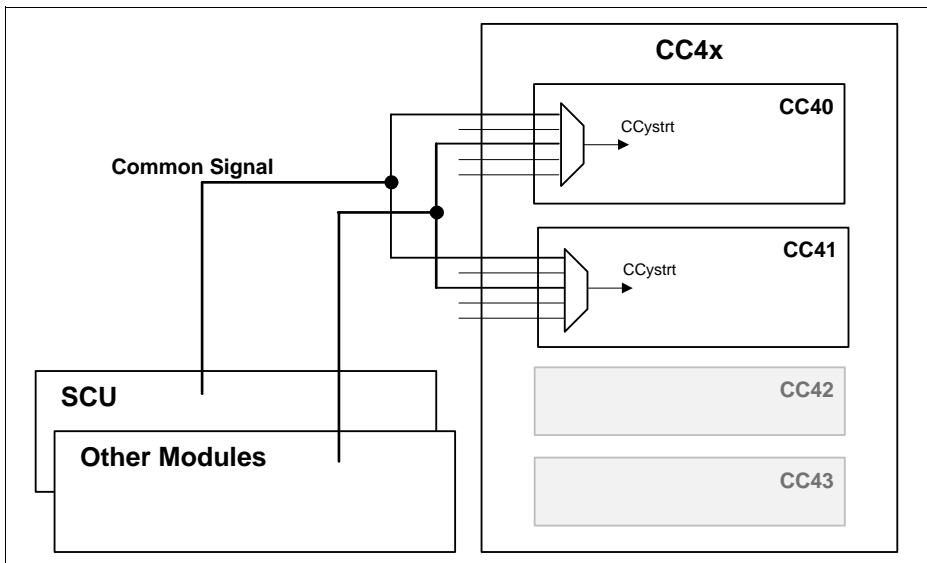


Figure 21-7 Starting multiple timers synchronously

21.2.5 Counting Modes

Each CC4y timer slice can be programmed into three different counting schemes:

- Edge aligned (default)
- Center aligned
- Single shot (can be edge or center aligned)

These three counting schemes can be used as stand alone without the need of selecting any inputs as external event sources. Nevertheless it is also possible to control the

Capture/Compare Unit 4 (CCU4)

counting operation via external events like, timer gating, counting trigger, external stop, external start, etc.

For all the counting modes, it is possible to update on the fly the values for the timer period and compare channel. This enables a cycle by cycle update of the PWM frequency and duty cycle.

The compare channel of each CC4y Timer Slice, has an associated Status Bit (**GCST.CC4yST**), that indicates the active or passive state of the channel, **Figure 21-8**. The set and clear of the status bit and the respective PWM signal generation is dictated by the timer period, compare value and the current counting mode. See the different counting mode descriptions, **Section 21.2.5.3** to **Section 21.2.5.5** to understand how this bit is set and cleared.

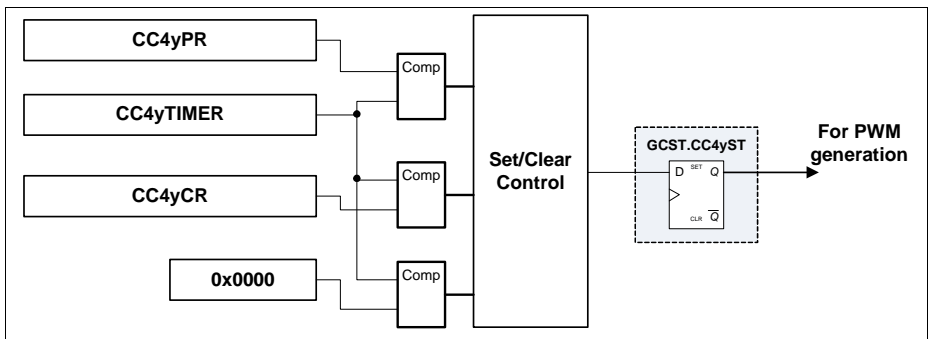


Figure 21-8 CC4y Status Bit

21.2.5.1 Calculating the PWM Period and Duty Cycle

The period of the timer is determined by the value in the period register, **CC4yPR** and by the timer mode.

The base for the PWM signal frequency and duty cycle, is always related to the clock frequency of the timer itself and not to the frequency of the module clock (due to the fact that the timer clock can be a scaled version of the module clock).

In Edge Aligned Mode, the timer period is:

$$T_{\text{per}} = \langle \text{Period-Value} \rangle + 1; \text{ in } f_{\text{tclk}} \quad (21.1)$$

In Center Aligned Mode, the timer period is:

$$T_{\text{per}} = (\langle \text{Period-Value} \rangle + 1) \times 2; \text{ in } f_{\text{tclk}} \quad (21.2)$$

For each of these counting schemes, the duty cycle of generated PWM signal is dictated by the value programmed into the **CC4yCR** register.

In Edge Aligned and Center Aligned Mode, the PWM duty cycle is:

$$DC = 1 - \frac{\text{Compare-Value}}{\text{Period-Value} + 1} \quad (21.3)$$

Both **CC4yPR** and **CC4yCR** can be updated on the fly via software, enabling a glitch free transition between different period and duty cycle values for the generated PWM signal, **Section 21.2.5.2**

21.2.5.2 Updating the Period and Duty Cycle

Each CCU4 timer slice provides an associated shadow register for the period and compare values. This facilitates a concurrent update by software for these two parameters, with the objective of modifying during run time the PWM signal period and duty cycle.

In addition to the shadow registers for the period and compare values, one also has available shadow registers for the floating prescaler and dither functions, **CC4yFPCS** and **CC4yDITS** respectively (please address **Section 21.2.11** and **Section 21.2.10** for a complete description of these functions).

The structure of the shadow registers can be seen in **Figure 21-9**.

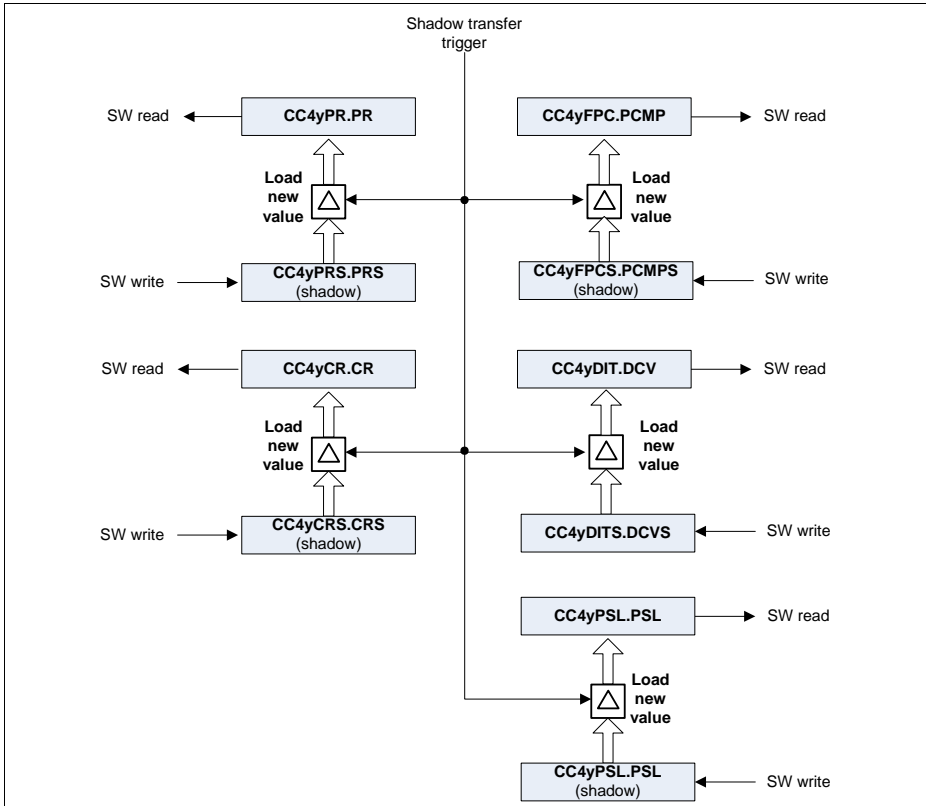


Figure 21-9 Shadow registers overview

The update of these registers can only be done by writing a new value into the associated shadow register and wait for a shadow transfer to occur.

Each group of shadow registers have an individual shadow transfer enable bit, [Figure 21-10](#). The software must set this enable bit to 1_B, whenever an update of the values is needed. These bits are automatically cleared by the hardware, whenever an update of the values is finished. Therefore every time that an update of the registers is needed the software must set again the specific bit(s).

Nevertheless it is also possible to clear the enable bit via software. This can be used in the case that an update of the values needs to be cancelled (after the enable bit has already been set).

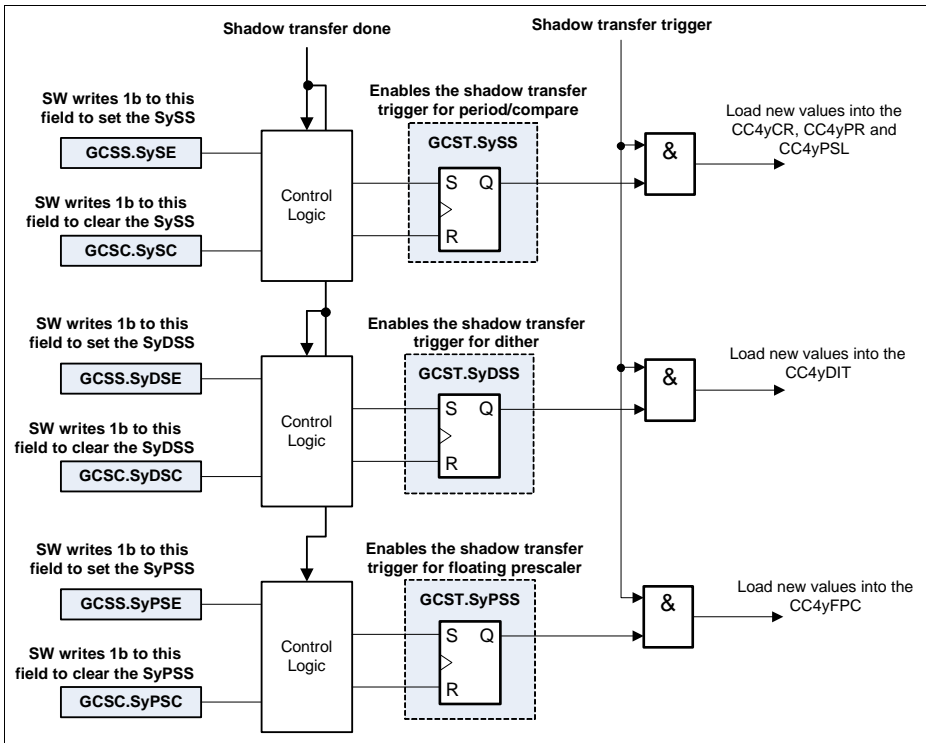


Figure 21-10 Shadow transfer enable logic

The shadow transfer operation is going to be done in the immediately next occurrence of a shadow transfer trigger, after the shadow transfer enable is set (**GCST.SySS**, **GCST.SyDSS**, **GCST.SyPSS** set to 1_B).

The occurrence of the shadow transfer trigger is imposed by the timer counting scheme (edge aligned or center aligned). Therefore the slots when the values are updated can be:

- in the next clock cycle after a Period Match while counting up
- in the next clock cycle after an One Match while counting down
- immediately, if the timer is stopped and the shadow transfer enable bit(s) is set

Figure 21-11 shows an example of the shadow transfer control when the timer slice has been configured into center aligned mode. For a complete description of all the timer slice counting modes, please address [Section 21.2.5.3](#), [Section 21.2.5.4](#) and [Section 21.2.5.5](#).

Capture/Compare Unit 4 (CCU4)

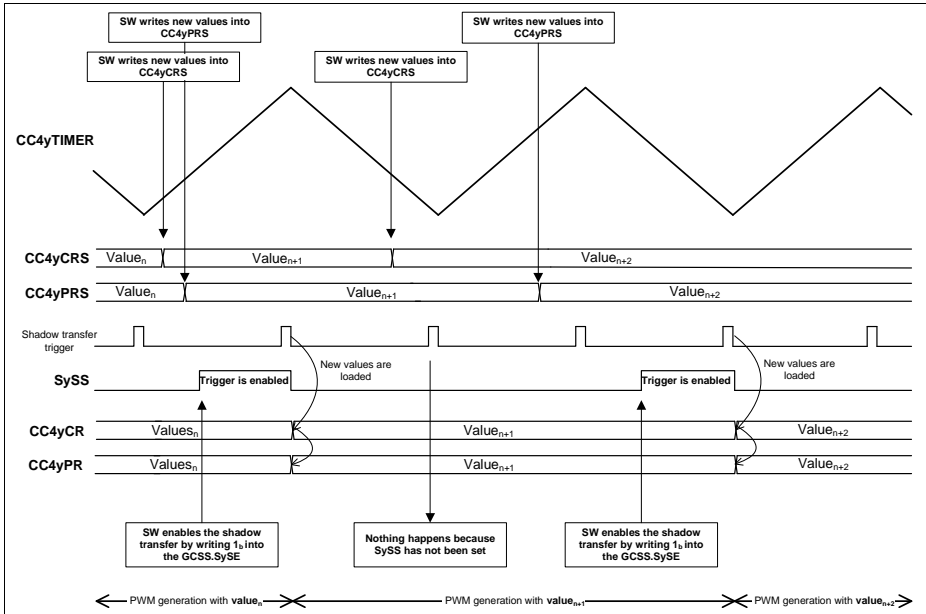


Figure 21-11 Shadow transfer timing example - center aligned mode

In some application cases it may be necessary to request shadow transfers not by software but by hardware. To perform this action each CCU4 contains a dedicated input that can be used to request a shadow transfer by hardware, the CCU4x.MCSS.

This input, when enabled, is used to set the shadow transfer enable bitfields (**GCST.SySS**, **GCST.SyDSS** and **GCST.SyPSS**) of the specific slice. It is possible to select which slice is using this input to perform the synchronization via the **GCTRL.MSEy** bit field. It is also possible to enable the usage of this signal for the three different shadow transfer signals: compare and period values, dither compare value and prescaler compare value. This can be configured on the **GCTRL.MSDE** field.

The structure for using the CCU4x.MCSS input signal can be seen in [Figure 21-10](#). The usage of this signal is just an add on to the shadow transfer control and therefore all the previous described functions are still available.

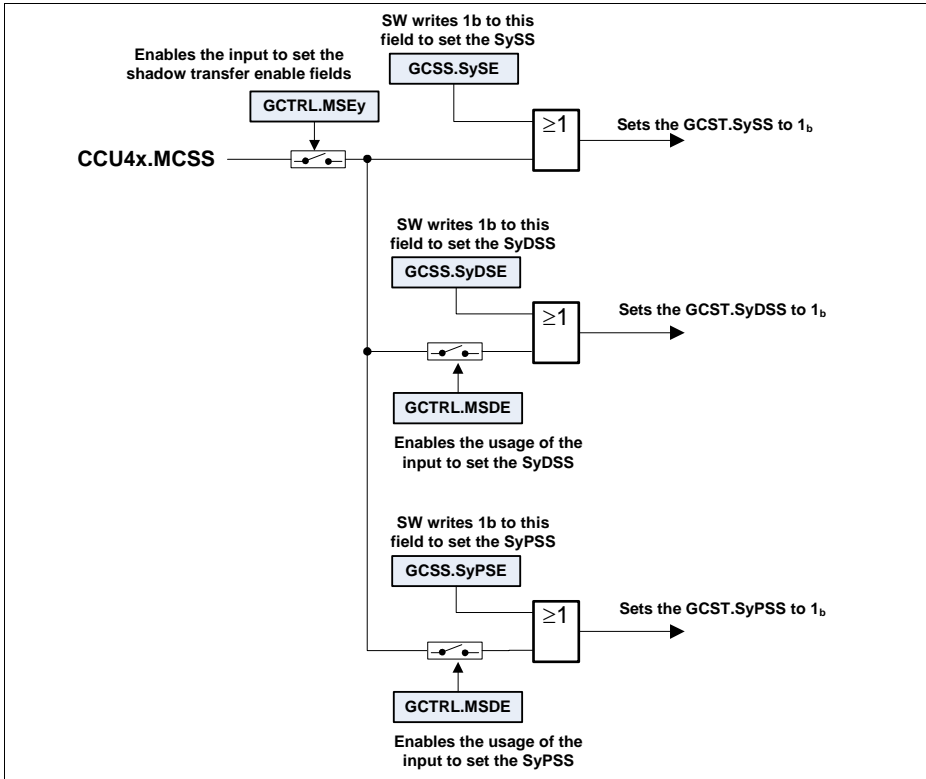


Figure 21-12 Usage of the CCU4x.MCSSS input

21.2.5.3 Edge Aligned Mode

Edge aligned mode is the default counting scheme. In this mode, the timer is incremented until it matches the value programmed in the period register, **CC4yPR**. When period match is detected the timer is cleared to 0000_H and continues to be incremented.

In this mode, the value of the period register and compare register are updated with the value written by software into the correspondent shadow register, every time that an overflow occurs (period match), see **Figure 21-13**.

In edge aligned mode, the status bit of the comparison (CC4yST) is set one clock cycle after the timer hits the value programmed into the compare register. The clear of the status bit is done one clock cycle after the timer reaches 0000_H.

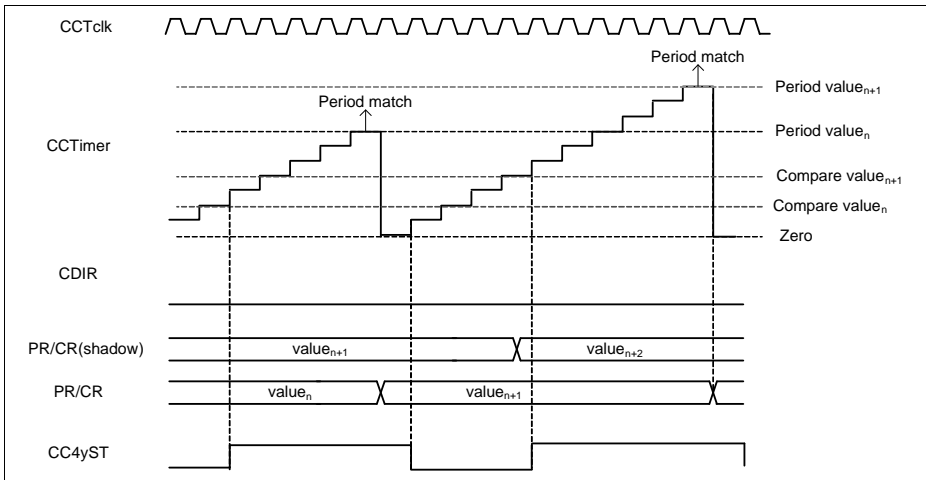


Figure 21-13 Edge aligned mode, $CC4yTCM = 0_B$

21.2.5.4 Center Aligned Mode

In center aligned mode, the timer is counting up or down with respect to the following rules:

- The counter counts up while $CC4yTCST.CDIR = 0_B$ and it counts down while $CC4yTCST.CDIR = 1_B$.
- Within the next clock cycle, the count direction is set to counting up ($CC4yTCST.CDIR = 0_B$) when the counter reaches 0001_H while counting down.
- Within the next clock cycle, the count direction is set to counting down ($CC4yTCST.CDIR = 1_B$), when the period match is detected while counting up.

The status bit ($CC4yST$) is always 1_B when the counter value is equal or greater than the compare value and 0_B otherwise.

While in edge aligned mode, the shadow transfer for compare and period registers is executed once per period. It is executed twice in center aligned mode as follows

- Within the next clock cycle after the counter reaches the period value, while counting up ($CC4yTCST.CDIR = 0_B$).
- Within the next clock cycle after the counter reaches 0001_H , while counting down ($CC4yTCST.CDIR = 1_B$).

Note: Bit $CC4yTCST.CDIR$ changes within the next timer clock after the one-match or the period-match, which means that the timer continues counting in the previous direction for one more cycle before changing the direction.

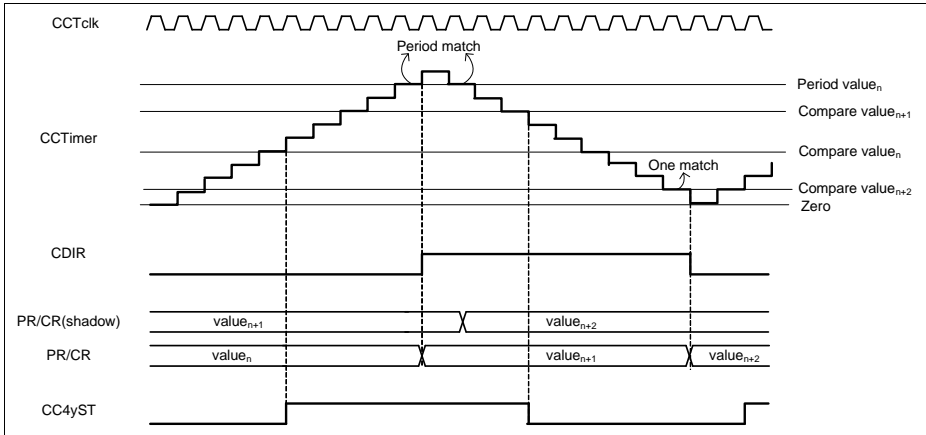


Figure 21-14 Center aligned mode, $CC4yTC.TCM = 1_B$

21.2.5.5 Single Shot Mode

In single shot mode, the timer is stopped after the current timer period is finished. This mode can be used with center or edge aligned scheme.

In edge aligned mode, **Figure 21-15**, the timer is stopped when it is cleared to 0000_H after having reached the period value. In center aligned mode, **Figure 21-16**, the period is finished when the timer has counted down to 0000_H.

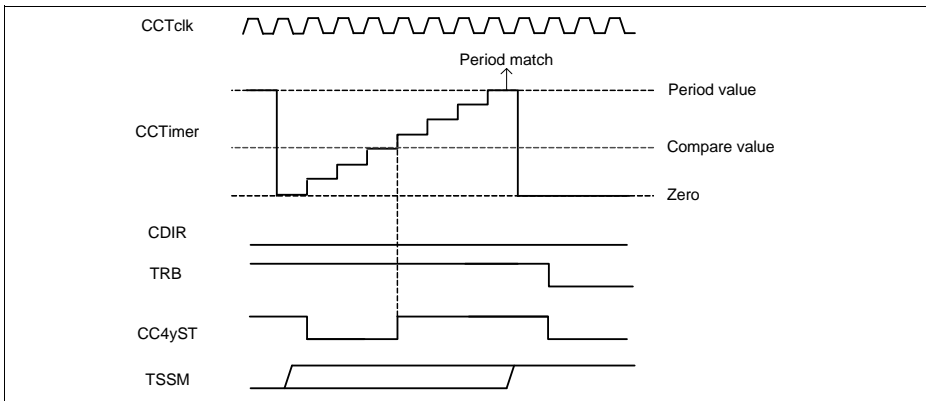


Figure 21-15 Single shot edge aligned - $CC4yTC.TSSM = 1_B$, $CC4yTC.TCM = 0_B$

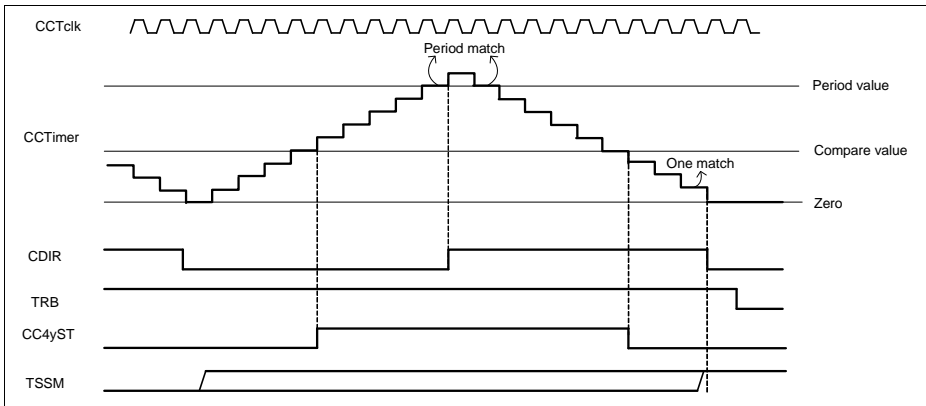


Figure 21-16 Single shot center aligned - $CC4yTC.TSSM = 1_B$, $CC4yTC.TCM = 1_B$

21.2.6 Active/Passive Rules

The general rules that set or clear the associated timer slice status bit (CC4yST), can be generalized independently of the timer counting mode.

The following events set the Status bit (CC4yST) to Active:

- in the next f_{tclk} cycle after a compare match while counting up
- in the next f_{tclk} cycle after a zero match while counting down

The following events set the Status bit (CC4yST) to Inactive:

- in the next f_{tclk} cycle after a zero match (and not compare match) while counting up
- in the next f_{tclk} cycle after a compare match while counting down

If external events are being used to control the timer operation, these rules are still applicable.

The status bit state can only be 'override' via software or by the external status bit override function, [Section 21.2.7.10](#).

The software can at any time write a 1_B into the [GCSS.SySTS](#) bitfield, which will set the status bit [GCST.CC4yST](#) of the specific timer slice. Writing a 1_B into the [GCSC.SySTC](#) bitfield will clear the specific status bit.

21.2.7 External Events Control

Each CCU4 timer slice has the possibility of using up to three different input events, see [Section 21.2.2](#). These three events can then be mapped to Timer Slice functions (the full set of available functions is described at [Section 21.2.3](#))

These events can be mapped to any of the CCU4x.INy[P...A] inputs and there isn't any imposition that an event cannot be used to perform several functions, or that an input

cannot be mapped to several events (e.g. input X triggers event 0 with rising edge and triggers event 1 with the falling edge).

21.2.7.1 External Start/Stop

To select an external start function, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC4yINS.EVxIS** field and indicating the active edge of the signal on the **CC4yINS.EVxEM** field.

This event should be then mapped to the start or stop functionality by setting the **CC4yCMC.STRTS** (for the start) or the **CC4yTC.ENDM** (for the stop) with the proper value.

Notice that both start and stop functions are edge and not level active and therefore the active/passive configuration is set only by the **CC4yINS.EVxEM**.

The external stop by default just clears the run bit (**CC4yTCST.TRB**), while the start functions does the opposite. Nevertheless one can select an extended subset of functions for the external start and stop. This subset is controlled by the registers **CC4yTC.ENDM** (for the stop) and **CC4yTC.STRM** (for the start).

For the start subset (**CC4yTC.STRM**):

- sets the run bit/starts the timer (resume operation)
- clears the timer, sets the run bit/starts the timer (flush and start)

For the stop subset (**CC4yTC.ENDM**):

- clears the run/stops the timer (stop)
- clears the timer (flush)
- clears the timer, clears the run bit/stops the timer (flush and stop)

If in conjunction with an external start/stop (configured also/only as flush) and external up/down signal is used, during the flush operation the timer is going to be set to 0000_H if the actual counting direction is up or set with the value of the period register if the counting direction is down.

Figure 21-17 to **Figure 21-20** shows the usage of two signals to perform the start/stop functions in all the previously mentioned subsets. External Signal(1) acts as an active HIGH start signal, while External Signal(2) is used as an active HIGH stop function.

Capture/Compare Unit 4 (CCU4)

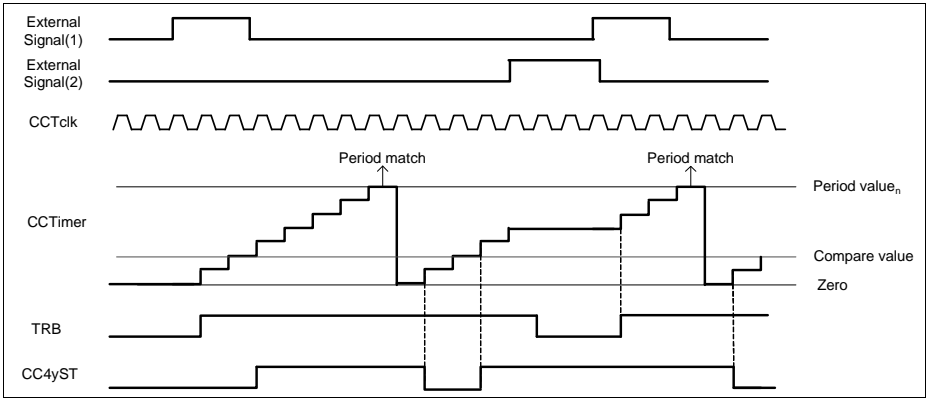


Figure 21-17 Start (as start)/ stop (as stop) - $CC4yTC.STRM = 0_B$, $CC4yTC.ENDM = 00_B$

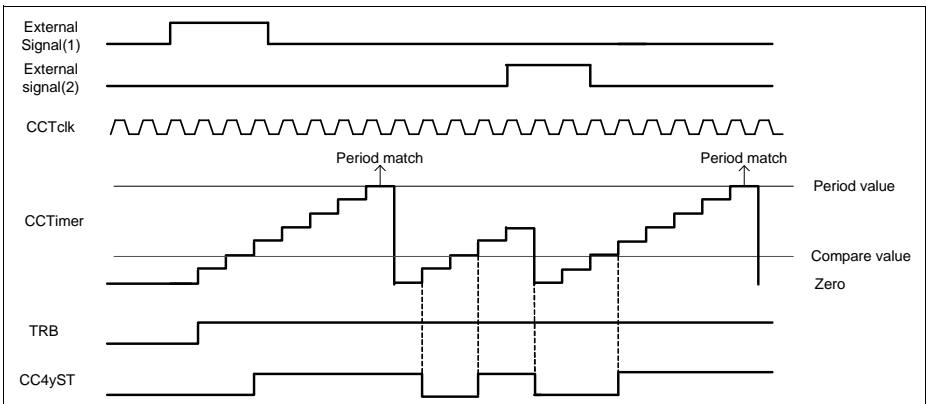


Figure 21-18 Start (as start)/ stop (as flush) - $CC4yTC.STRM = 0_B$, $CC4yTC.ENDM = 01_B$

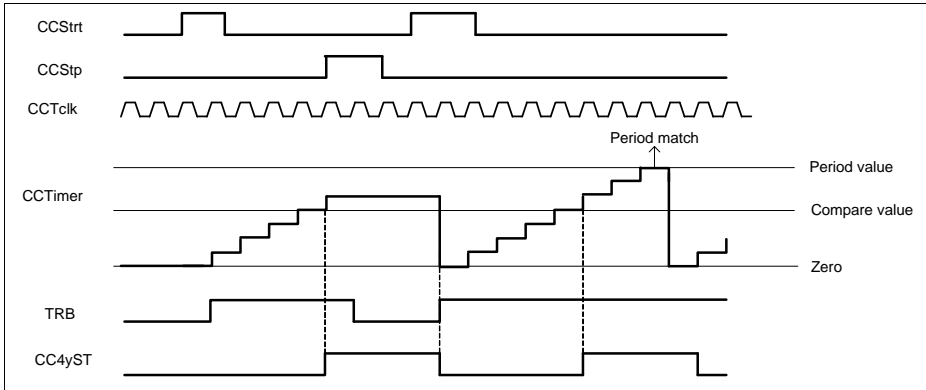


Figure 21-19 Start (as flush and start)/ stop (as stop) - $CC4yTC.STRM = 1_B$, $CC4yTC.ENDM = 00_B$

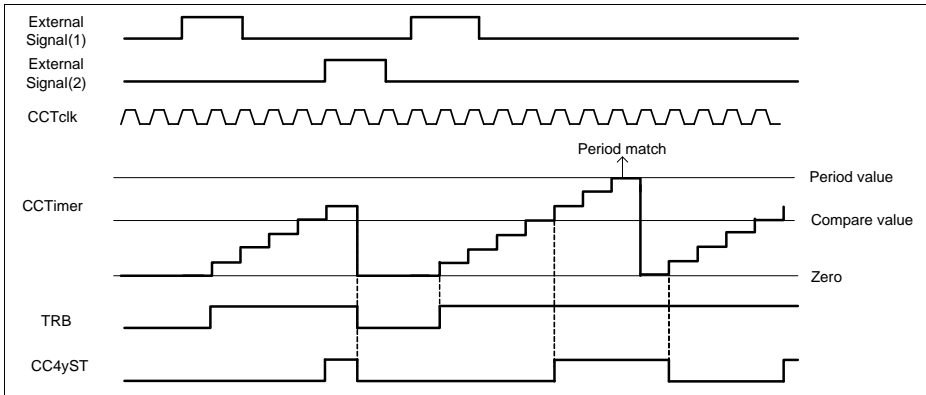


Figure 21-20 Start (as start)/ stop (as flush and stop) - $CC4yTC.STRM = 0_B$, $CC4yTC.ENDM = 10_B$

21.2.7.2 External Counting Direction

There is the possibility of selecting an input signal to act as increment/decrement control.

To select an external up/down control, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the $CC4yINS.EVxIS$ field and indicating the active level of the signal on the $CC4yINS.EVxLM$. This event should be then mapped to the up/down functionality by setting $CC4yCMC.UDS$ with the proper value.

Capture/Compare Unit 4 (CCU4)

Notice that the up/down function is level active and therefore the active/passive configuration is set only by the **CC4yINS.EVxLM**.

The status bit of the slice (**CC4yST**) is always set when the timer value is equal or greater than the value stored in the compare register, see **Section 21.2.6**.

The update of the period and compare register values is done when:

- with the next clock after a period match, while counting up (**CC4yTCST.CDIR** = 0_B)
- with the next clock after a one match, while counting down (**CC4yTCST.CDIR** = 1_B)

The value of the **CC4yTCST.CDIR** register is updated accordingly with the changes on the decoded event. The Up/Down direction is always understood as **CC4yTCST.CDIR** = 1_B when counting down and **CC4yTCST.CDIR** = 0_B when counting up. Using an external signal to perform the up/down counting function and configuring the event as active HIGH means that the timer is counting up when the signal is HIGH and counting down when LOW.

Figure 21-21 shows an external signal being used to control the counting direction of the time. This signal was selected as active HIGH, which means that the timer is counting down while the signal is HIGH and counting up when the signal is LOW.

*Note: For a signal that should impose an increment when LOW and a decrement when HIGH, the user needs to set the **CC4yINS.EVxLM** = 0_B. When the operation is switched, then the user should set **CC4yINS.EVxLM** = 1_B.*

Note: Using an external counting direction control, sets the slice in edge aligned mode.

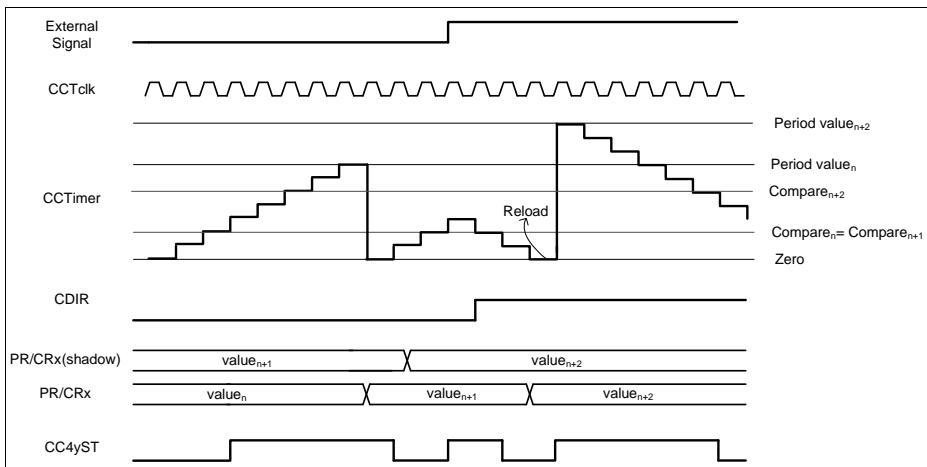


Figure 21-21 External counting direction

21.2.7.3 External Gating Signal

For pulse measurement, the user has the possibility of selecting an input signal that operates as counting gating.

To select an external gating control, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC4yINS.EVxIS** register and indicating the active level of the signal on the **CC4yINS.EVxLM** register. This event should be then mapped to the gating functionality by setting the **CC4yCMC.GATES** with the proper value.

Notice that the gating function is level active and therefore the active/passive configuration is set only by the **CC4yINS.EVxLM**.

The status bit during an external gating signal continues to be asserted when the compare value is reached and deasserted when the counter reaches 0000_H. One should note that the counter continues to use the period register to identify the wrap around condition. **Figure 21-22** shows the usage of an external signal for gating the slice counter. The signal was set as active LOW, which means the counter gating functionality is active when the external value is zero.

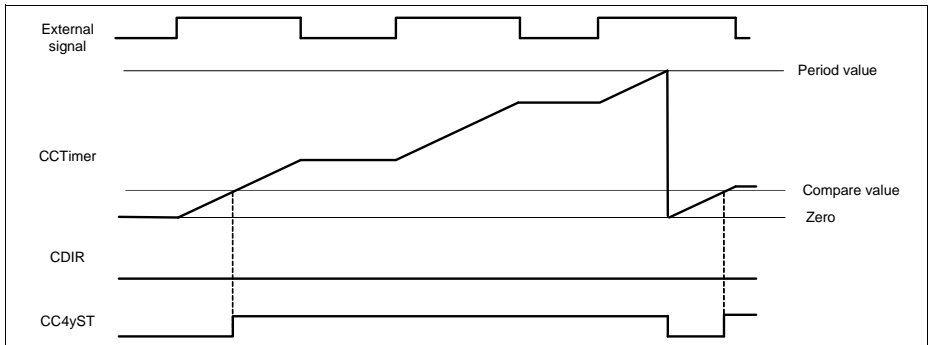


Figure 21-22 External gating

For any type of usage of the external gating function, the specific run bit of the Timer Slice, **CC4yTCST.TRB**, needs to be set. This can be done via an additional external signal or directly via software.

21.2.7.4 External Count Signal

There is also the possibility of selecting an external signal to act as the counting event.

To select an external counting, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC4yINS.EVxIS** register and indicating the active edge of the signal on the **CC4yINS.EVxEM** register.

Capture/Compare Unit 4 (CCU4)

This event should be then mapped to the counting functionality by setting the **CC4yCMC.CNTS** with the proper value.

Notice that the counting function is edge active and therefore the active/passive configuration is set only by the **CC4yINS.EVxEM**.

One can select just a the rising, falling or both edges to perform a count. On **Figure 21-23**, the external signal was selected as a counter event for both falling and rising edges. Wrap around condition is still applied with a comparison with the period register.

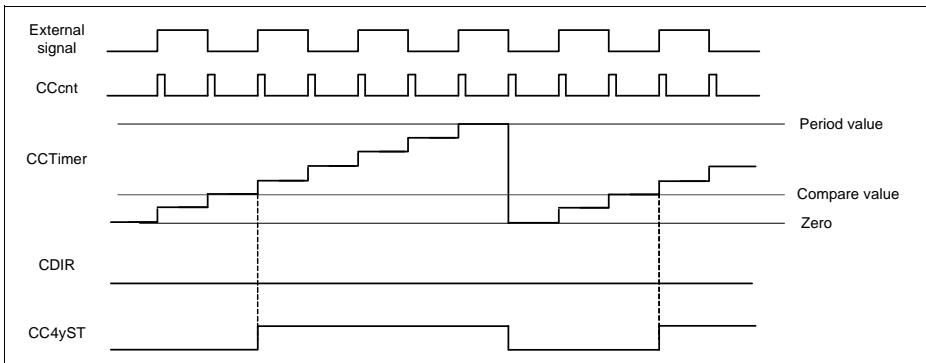


Figure 21-23 External count

For any type of usage of the external gating function, the specific run bit of the Timer Slice, **CC4yTCST.TRB**, needs to be set. This can be done via an additional external signal or directly via software.

21.2.7.5 External Load

Each slice of the CCU4 also has a functionality that enables the user to select an external signal as trigger for reloading the value of the timer with the current value of the compare register (if **CC4yTCST.CDIR = 0_B**) or with the value of the period register (if **CC4yTCST.CDIR = 1_B**).

To select an external load signal, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC4yINS.EVxIS** register and indicating the active edge of the signal on the **CC4yINS.EVxEM** register. This event should be then mapped to the load functionality by setting the **CC4yCMC.LDS** with the proper value.

Notice that the load function is edge active and therefore the active/passive configuration is set only by the **CC4yINS.EVxEM**.

On figure **Figure 21-24**, the external signal (1) was used to act as a load trigger, active on the rising edge. Every time that a rising edge on external signal (1) is detected, the

Capture/Compare Unit 4 (CCU4)

timer value is loaded with the value present on the compare register. If an external signal is being used to control the counting direction, up or down, the timer value can be loaded also with the value set in the period register. The External signal (2) represents the counting direction control (active HIGH). If at the moment that a load trigger is detected, the signal controlling the counting direction is imposing a decrement, then the value set in the timer is the period value.

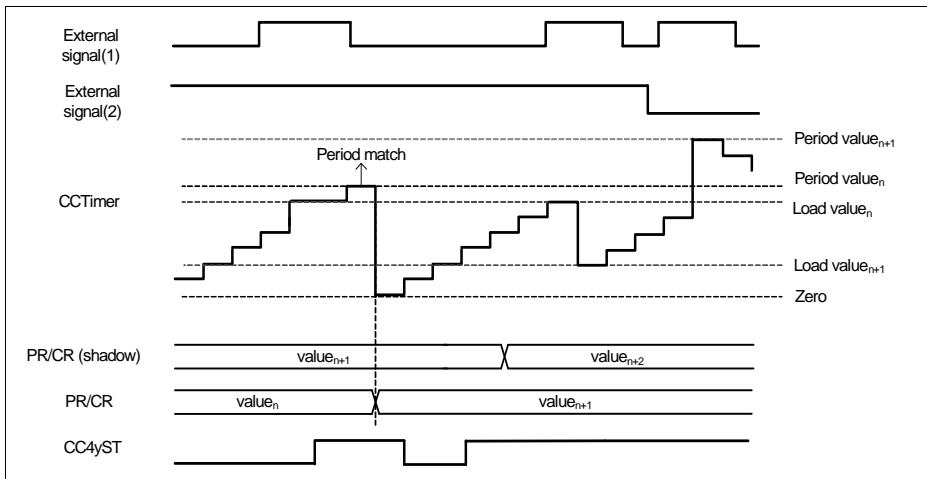


Figure 21-24 External load

21.2.7.6 External Capture

When selecting an external signal to be used as a capture trigger (if **CC4yCMC.CAP0S** or **CC4yCMC.CAP1S** are different from 0_H), the user is automatically setting the specific slice into capture mode.

In capture mode the user can have up to four capture registers, see **Figure 21-27**: capture register 0 (**CC4yC0V**), capture register 1 (**CC4yC1V**), capture register 2 (**CC4yC2V**) and capture register 3 (**CC4yC3V**).

These registers are shared between compare and capture modes which imposes:

- if **CC4yC0V** and **CC4yC1V** are used for capturing, the compare registers **CC4yCR** and **CC4yCRS** are not available (no compare channel)
- if **CC4yC2V** and **CC4yC3V** are used for capturing, the period registers **CC4yPR** and **CC4yPRS** are not available (no period control)

To select an external capture signal, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC4yINS.EVxIS** register and indicating the active edge of the signal on the

Capture/Compare Unit 4 (CCU4)

CC4yINS.EVxEM register. This event should be then mapped to the capture functionality by setting the **CC4yCMC.CAP0S/CC4yCMC.CAP1S** with the proper value. Notice that the capture function is edge active and therefore the active/passive configuration is set only by the **CC4yINS.EVxEM**.

The user has the possibility of selecting the following capture schemes:

- Different capture events for **CC4yC0V/CC4yC1V** and **CC4yC2V/CC4yC3V**
- The same capture event for **CC4yC0V/CC4yC1V** and **CC4yC2V/CC4yC3V** with the same capture edge. For this capture scheme, only the CCcapt1 functionality needs to be programmed. To enable this scheme, the field **CC4yTC.SCE** needs to be set to 1.

Different Capture Events (SCE = 0_B)

Every time that a capture trigger 1 occurs, CCcapt1, the actual value of the timer is captured into the capture register 3 and the previous value stored in this register is transferred into capture register 2.

Every time that a capture trigger 0 occurs, CCcapt0, the actual value of the timer is captured into the capture register 1 and the previous value stored in this register is transferred into capture register 0.

Every time that a capture procedure into one of the registers occurs, the respective full flag is set. This flag is cleared automatically by HW when the SW reads back the value of the capture register (by reading the specific capture register or by reading the extended capture read value, see [Section 21.2.7.7](#)).

The capture of a new value into a specific capture registers is dictated by the status of the full flag as follows:

$$CC4yCIV_{capt} = \text{NOT}(CC4yCIV_{full_flag} \text{ AND } CC4yC0V_{full_flag}) \quad (21.4)$$

$$CC4yC0V_{capt} = CC4yCIV_{full_flag} \text{ AND NOT}(CC4yC0V_{full_flag}) \quad (21.5)$$

It is also possible to disable the effect of the full flags reset by setting the **CC4yTC.CCS = 1_B**. This enables a continuous capturing independent if the values captured have been read or not.

*Note: When using the period registers for capturing, **CC4yCMC.CAP1S** different from 00_B, the counter always uses its full 16 bit width as period value.*

On [Figure 21-25](#), an external signal was selected as an event for capturing the timer value into the **CC4yC0V/CC4yC1V** registers. The status bit, CC4yST, during capture mode is asserted whenever a capture trigger is detected and deasserted when the counter matches 0000_H.

Capture/Compare Unit 4 (CCU4)

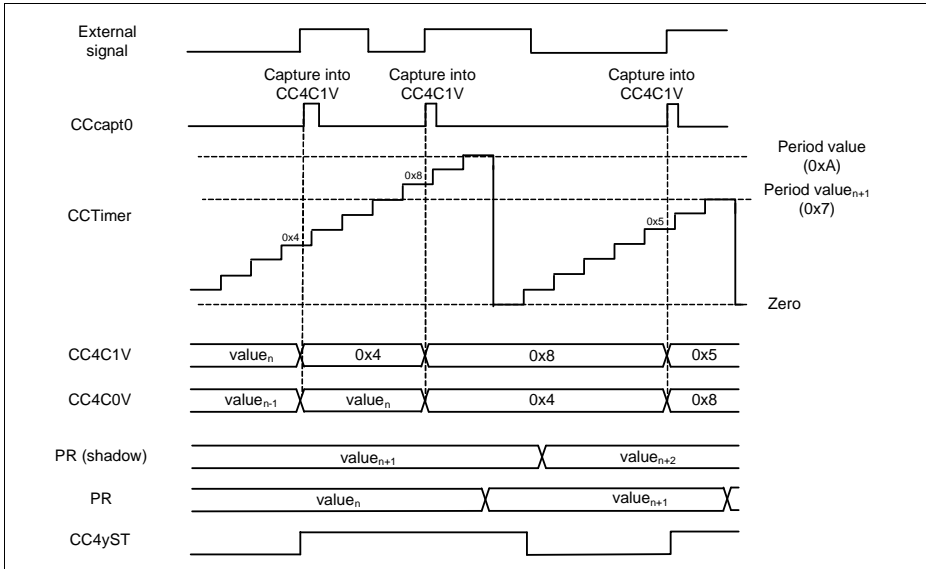


Figure 21-25 External capture - $CC4yCMC.CAP0S \neq 00_B$, $CC4yCMC.CAP1S = 00_B$

On [Figure 21-26](#), two different signals were used as source for capturing the timer value into the [CC4yC0V/CC4yC1V](#) and [CC4yC2V/CC4yC3V](#) registers.

External signal(1) was selected as rising edge active capture source for [CC4yC0V/CC4yC1V](#). External signal(2) was selected as the capture source for [CC4yC2V/CC4yC3V](#), but as opposite to the external signal(1), the active edge was selected as falling.

See [Section 21.2.12.4](#), for the complete capture mode usage description.

Capture/Compare Unit 4 (CCU4)

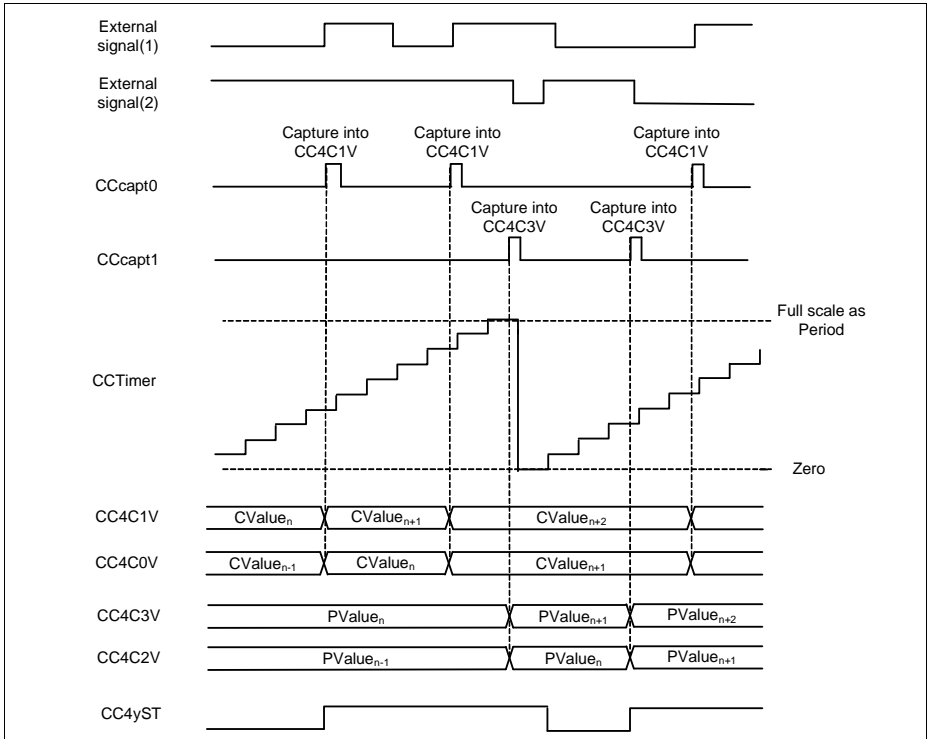


Figure 21-26 External capture - $CC4yCMC.CAP0S \neq 00_B$, $CC4yCMC.CAP1S \neq 00_B$

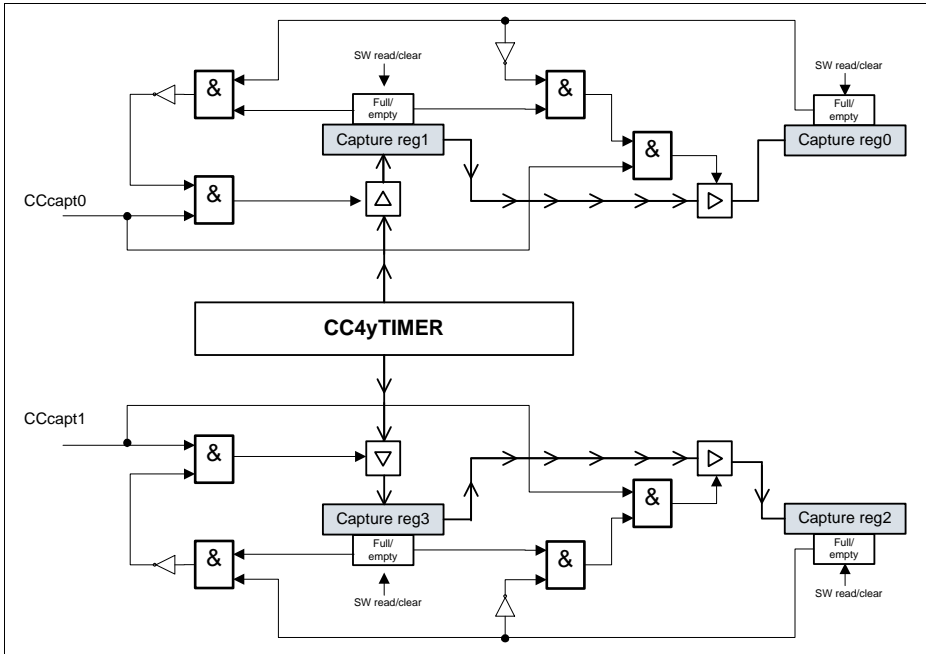


Figure 21-27 Slice capture logic

Same Capture Event (SCE = 1_B)

Setting the field **CC4yTC.SCE** = 1_B, enables the possibility of having 4 capture registers linked with the same capture event, **Figure 21-29**. The function that controls the capture is the CCcapt1.

The capture logic follows the same structure shown in **Figure 21-27** but extended to a four register chain, see **Figure 21-28**. The same full flag lock rules are applied to the four register chain (it also can be disabled by setting the **CC4yTC.CCS** = 1_B):

$$CC4yC3V_{capt} = \text{NOT}(CC4yC3V_{full_flag} \text{ AND } CC4yC2V_{full_flag} \text{ AND } CC4yC2V_{full_flag} \text{ AND } CC4yC1V_{full_flag}) \quad (21.6)$$

$$CC4yC2V_{capt} = CC4yC3V_{full_flag} \text{ AND NOT}(CC4yC2V_{full_flag} \text{ AND } CC4yC1V_{full_flag} \text{ AND } CC4yC0V_{full_flag}) \quad (21.7)$$

$$CC4yC1V_{capt} = CC4yC2V_{full_flag} \text{ AND NOT}(CC4yC1V_{full_flag} \text{ AND } CC4yC0V_{full_flag}) \quad (21.8)$$

$$CC4yC0V_{capt} = CC4yC1V_{full_flag} \text{ AND NOT}(CC4yC0V_{full_flag}) \quad (21.9)$$

Capture/Compare Unit 4 (CCU4)

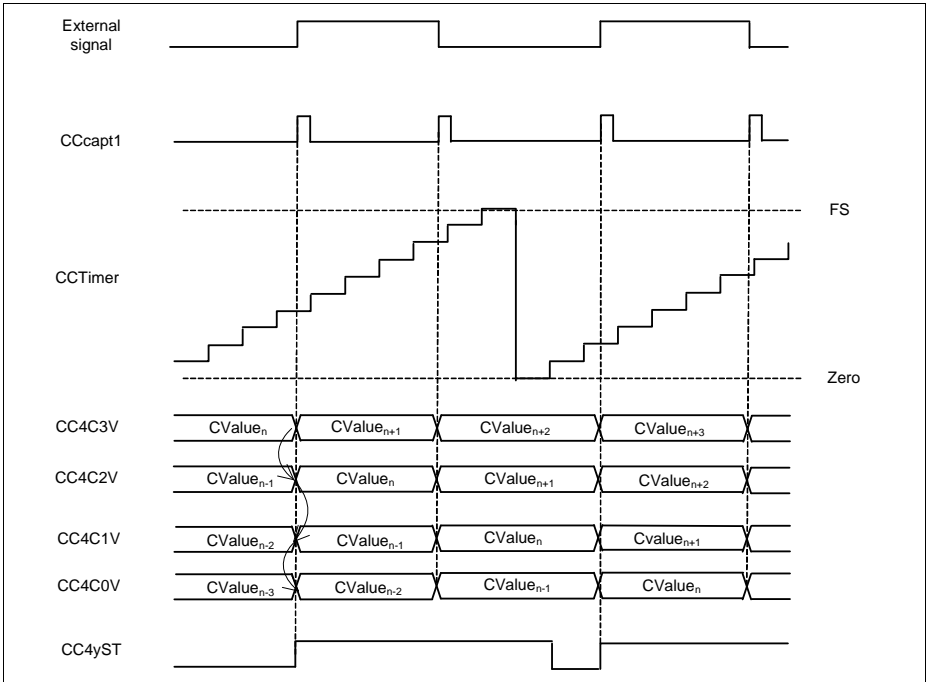


Figure 21-28 External Capture - CC4yTC.SCE = 1_B

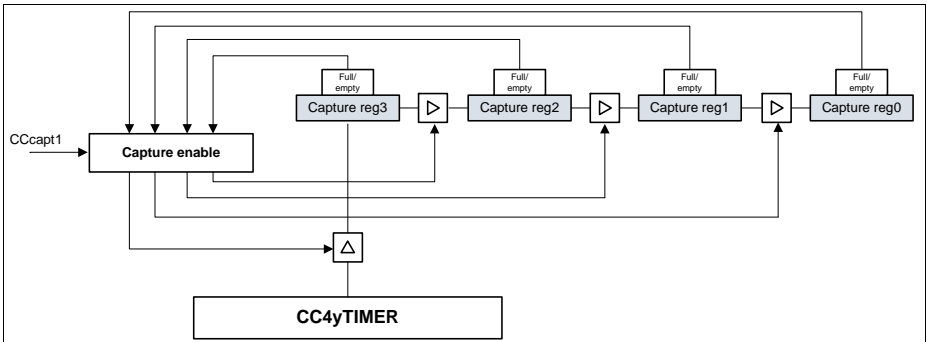


Figure 21-29 Slice Capture Logic - CC4yTC.SCE = 1_B

21.2.7.7 Capture Extended Read Back Mode

Each Timer Slice capture logic can operate in a FIFO read back mode. This mode can be enabled by setting the **CC4yTC.ECM** = 1_B. This Extended Read back mode allows the software to read back the capture data always from the same address (**CC4yECRD0** for the structure linked with the capture trigger 0 or **CC4yECRD1** for the one linked with capture trigger 1). This read back will always return the oldest captured value, enabling an easy software routine implementation for reconstructing the capture data.

This function allows the usage of a FIFO structure for each capturing trigger. This relaxes the software read back routine when multiple capture triggers are present, and the software is not fast enough to perform a read operation in each capture event.

This FIFO read back function is present for a depth-4 and depth-2 FIFO structure.

The read back data contains also a lost value bitfield, that indicates if a capture trigger was lost due to the fact that the FIFO structure was full. This bitfield is set whenever a capture event was sensed and the FIFO was full (regardless if the continuous capture mode was enabled or not). This bitfield is cleared automatically by HW whenever the next read of the **CC4yECRD0/CC4yECRD1** register occurs. This bitfield does not indicate how many capture events were lost, it just indicates that between two ECRD reads at least a capture event was lost (this can help the SW evaluate which part of the data read, can be used for calculation).

*Note: When the ECM bitfield is set, reading the individual capture registers is still possible. Nevertheless the full flags can only be cleared by the HW when a read back is done via the **CC4yECRD0/CC4yECRD1** address.*

Depth 4 Structure

The FIFO depth-4 structure is present in the hardware when the capture trigger 1 is enabled and the **CC4yTC.SCE** = 1_B (same capture event), **Figure 21-30**.

Capture/Compare Unit 4 (CCU4)

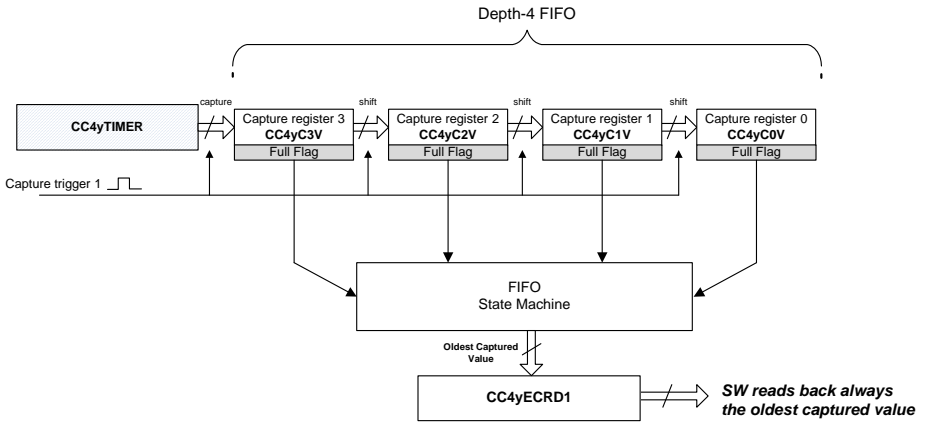


Figure 21-30 Capture Extended Read Back - Depth 4

Capture/Compare Unit 4 (CCU4)

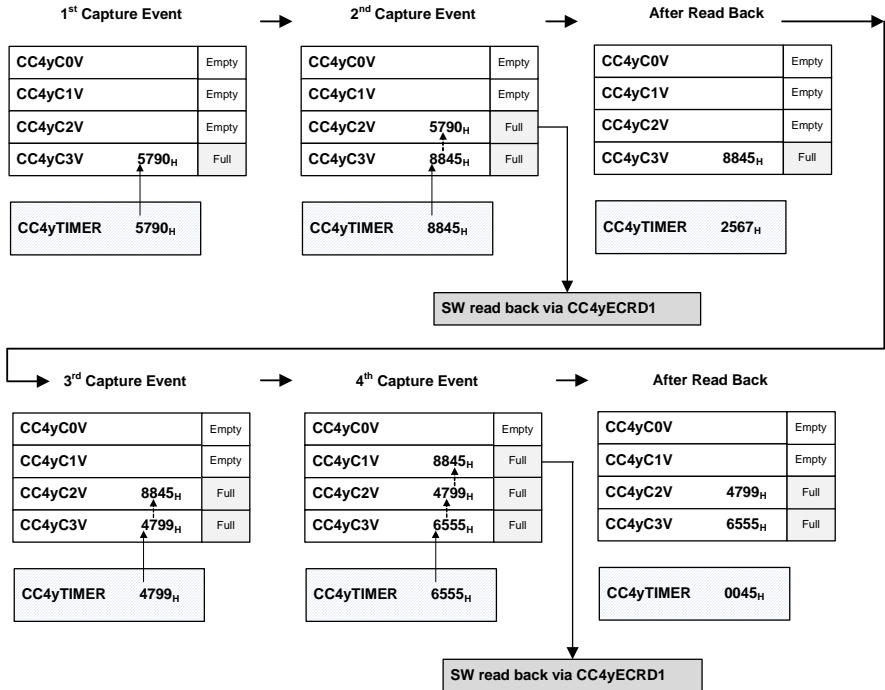


Figure 21-31 Depth 4 software access example

Depth 2 Structure

Each Timer Slice can have two capture structures of depth-2: one used with capture trigger 0 and another with capture trigger 1.

The one linked with capture trigger 0, is accessed via the **CC4yECDR0** while the one linked with the capture trigger 1 is accessed via the **CC4yECDR1**, **Figure 21-32**.

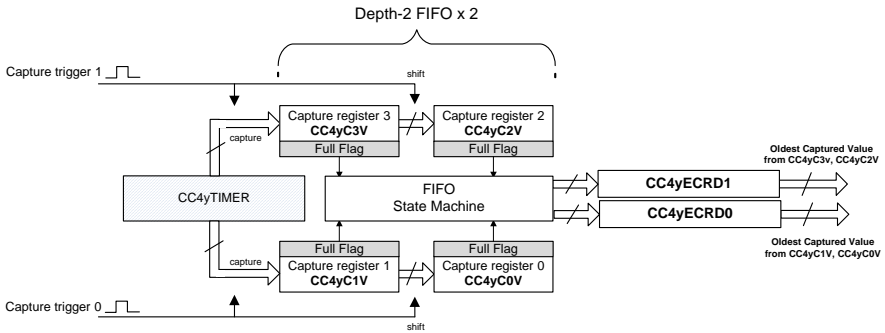


Figure 21-32 Capture Extended Read Back - Depth 2

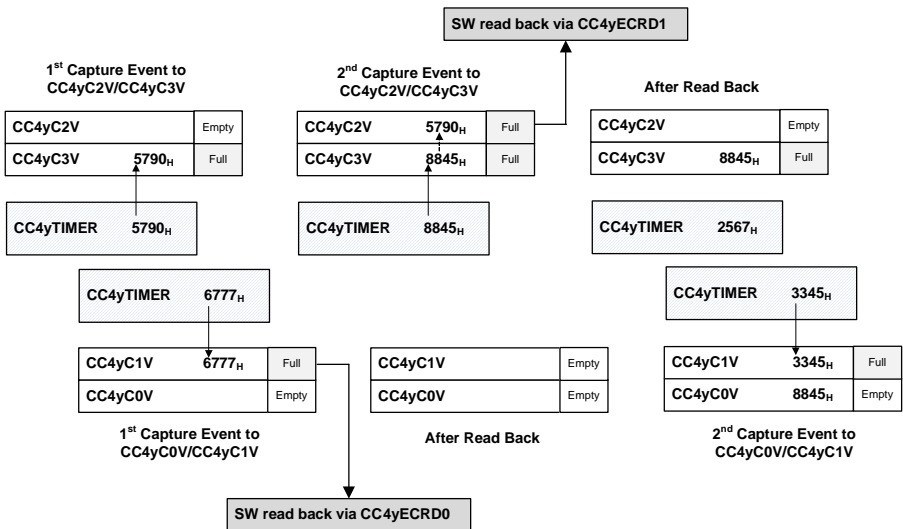


Figure 21-33 Depth 2 software access example

21.2.7.8 External Modulation

An external signal can be used to perform a modulation at the output of each timer slice. To select an external modulation signal, one should map one of the input signals to one of the events, by setting the required value in the **CC4yINS.EVxIS** register and indicating the active level of the signal on the **CC4yINS.EVxLM** register. This event should be then

Capture/Compare Unit 4 (CCU4)

mapped to the modulation functionality by setting the **CC4yCMC.MOS** = 01_B if event 0 is being used, **CC4yCMC.MOS** = 10_B if event 1 or **CC4yCMC.MOS** = 11_B if event 2.

Notice that the modulation function is level active and therefore the active/passive configuration is set only by the **CC4yINS.EVxLM**.

The modulation has two modes of operation:

- modulation event is used to clear the CC4yST bit - **CC4yTC.EMT** = 0_B
- modulation event is used to gate the outputs - **CC4yTC.EMT** = 1_B

On **Figure 21-34**, we have an external signal configured to act as modulation source that clears the CC4yST bit, **CC4yTC.EMT** = 0_B. It was programmed to be an active LOW event and therefore, when this signal is LOW the output value follows the normal ACTIVE/PASSIVE rules.

When the signal is HIGH (inactive state), then the CC4yST bit is cleared and the output is forced into the PASSIVE state. Notice that the values of the status bit, CC4yST and the specific output CCU4x.OUTy are not linked together. One can choose for the output to be active LOW or HIGH through the PSL bit.

The exit of the external modulation inactive state is synchronized with the PWM signal due to the fact that the CC4yST bit is cleared and cannot be set while the modulation signal is inactive.

The entering into inactive state also can be synchronized with the PWM signal, by setting **CC4yTC.EMS** = 1_B. With this all possible glitches at the output are avoided, see **Figure 21-35**.

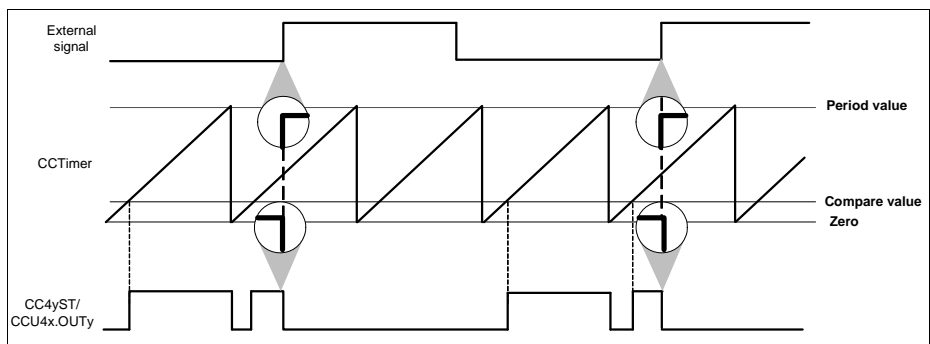


Figure 21-34 External modulation clearing the ST bit - **CC4yTC.EMT = 0_B**

Capture/Compare Unit 4 (CCU4)

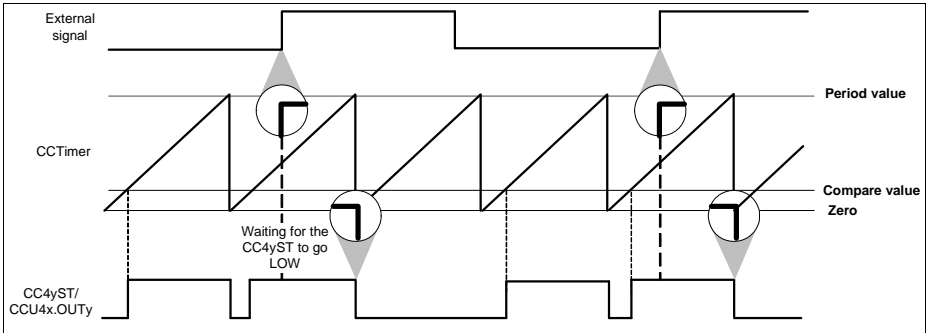


Figure 21-35 External modulation clearing the ST bit - $CC4yTC.EMT = 0_B$,
 $CC4yTC.EMS = 1_B$

On [Figure 21-36](#), the external modulation event was used as gating signal of the outputs, $CC4yTC.EMT = 1_B$. The external signal was configured to be active HIGH, $CC4yINS.EVxLM = 0_B$, which means that when the external signal is HIGH the outputs are set to the PASSIVE state. In this mode, the gating event can also be synchronized with the PWM signal by setting the $CC4yTC.EMS = 1_B$.

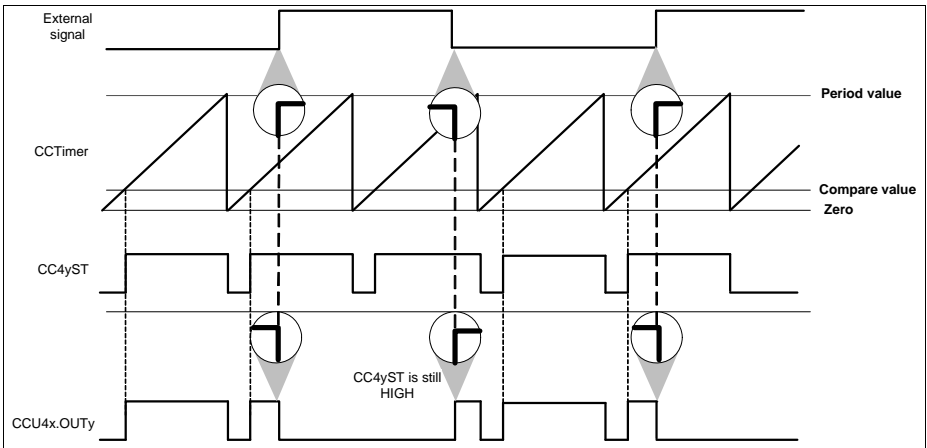


Figure 21-36 External modulation gating the output - $CC4yTC.EMT = 1_B$

21.2.7.9 TRAP Function

The TRAP functionality allows the PWM outputs to react on the state of an input pin. This functionality can be used to switch off the power devices if the TRAP input becomes active.

To select the TRAP functionality, one should map one of the input signals to event number 2, by setting the required value in the **CC4yINS.EV2IS** register and indicating the active level of the signal on the **CC4yINS.EV2LM** register. This event should be then mapped to the trap functionality by setting the **CC4yCMC.TS = 1_B**.

Notice that the trap function is level active and therefore the active/passive configuration is set only by the **CC4yINTS.EV2LM**.

There are two bitfields that can be monitored via software to crosscheck the TRAP function, **Figure 21-37**:

- The TRAP state bit, **CC4yINTS.E2AS**. This bitfield is the TRAP is currently active or not. This bitfield is therefore setting the specific Timer Slice output, into ACTIVE or PASSIVE state.
- The TRAP Flag, **CC4yINTS.TRPF**. This bitfield is used as a remainder in the case that the TRAP condition is cleared automatically via hardware. This field needs to be cleared by the software.

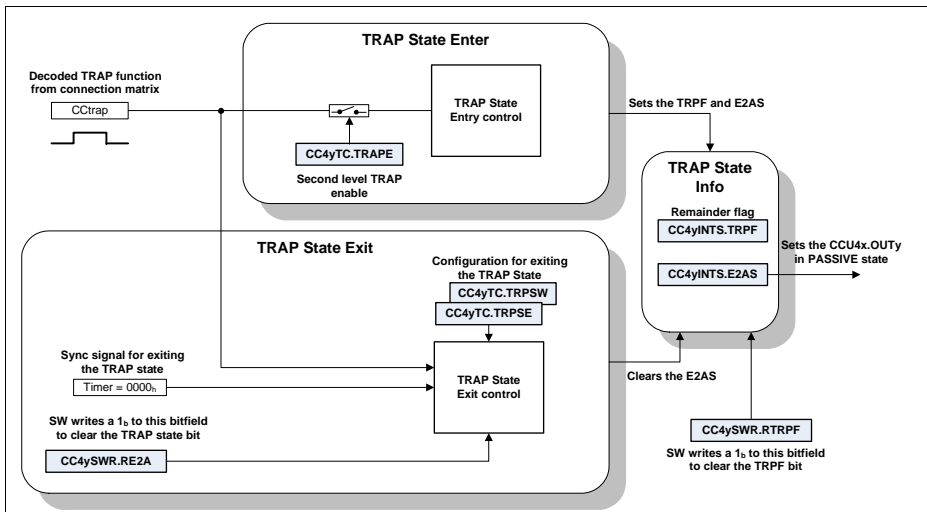


Figure 21-37 Trap control diagram

When a TRAP condition is detected at the selected input pin, both the Trap Flag and the Trap State bit are set to 1_B. The Trap State is entered immediately, by setting the CCU4xOUTy into the programmed PASSIVE state, **Figure 21-38**.

Capture/Compare Unit 4 (CCU4)

Exiting the Trap State can be done in two ways (**CC4yTC**.TRPSW register):

- automatically via HW, when the TRAP signal becomes inactive - **CC4yTC**.TRPSW = 0_B
- by SW only, by clearing the **CC4yINTS**.E2AS. The clearing is only possible if the input TRAP signal is in inactive state - **CC4yTC**.TRPSW = 1_B

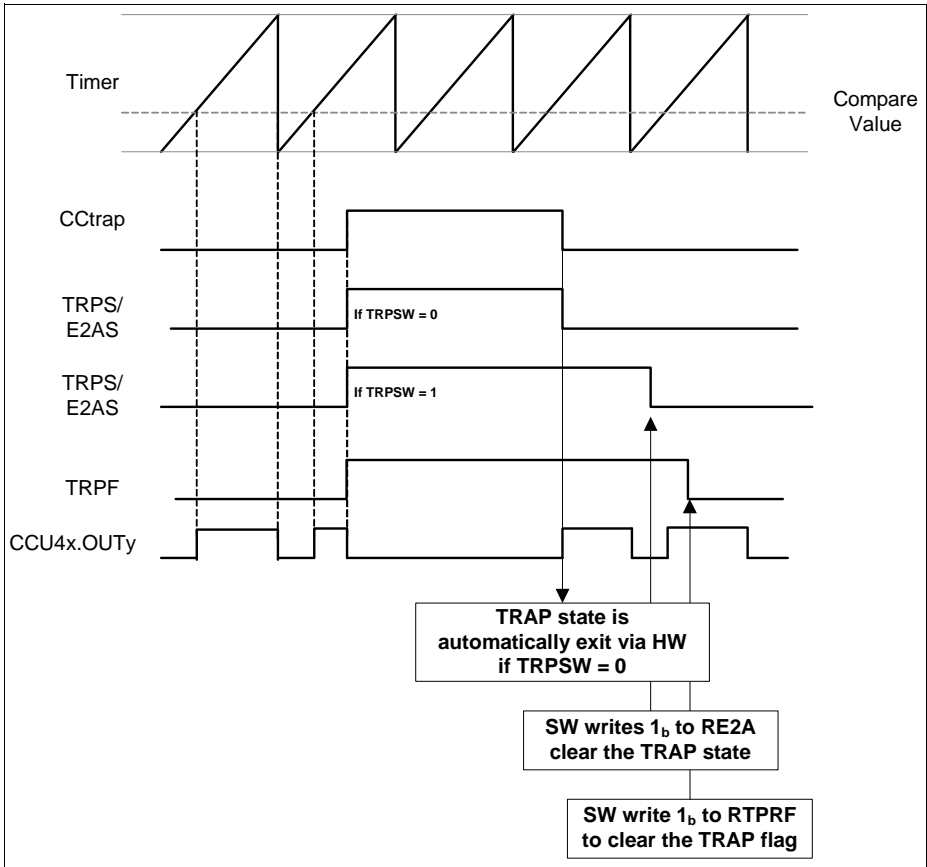


Figure 21-38 Trap timing diagram, **CC4yPSL**.PSL = 0_B (output passive level is 0_B)

It is also possible to synchronize the exiting of the TRAP state with the PWM signal, **Figure 21-39**. This function is enabled when the bitfield **CC4yTC**.TRPSE = 1_B .

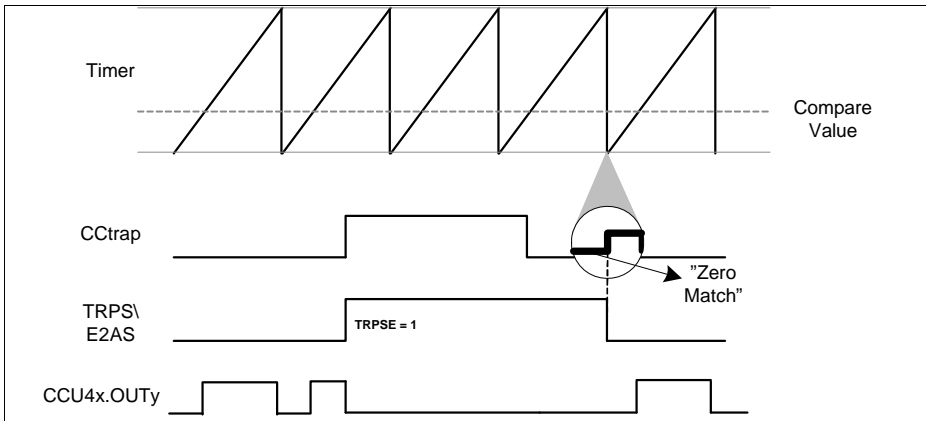


Figure 21-39 Trap synchronization with the PWM signal, **CC4yTC.TRPSE = 1_B**

21.2.7.10 Status Bit Override

For complex timed output control, each Timer Slice has a functionality that enables the override of the status bit (CC4yST) with a value passed through an external signal.

The override of the status bit, can then lead to a change on the output pin, CCU4xOUTy (from inactive to active or vice versa).

To enable this functionality, two signals are needed:

- One signal that acts as a trigger to override the status bit (edge active)
- One signal that contains the value to be set in the status bit (level active)

To use the status bit override functionality, one should map the signal that acts as trigger to the event number 1, by setting the required value in the **CC4yINS.EV1IS** register and indicating the active edge of the signal on the **CC4yINS.EV1EM** register.

The signal that carries the value to be set on the status bit, needs to be mapped to the event number 2, by setting the required value in the **CC4yINS.EV2IS** register. The **CC4yINS.EV2LM** register should be set to 0_B if no inversion on the signal is needed and to 1_B otherwise.

The events should be then mapped to the status bit functionality by setting the **CC4yCMC.OFS = 1_B**.

Figure 21-40 shows the functionality of the status bit override, when the external signal(1) was selected as trigger source (rising edge active) and the external signal(2) was selected as override value.

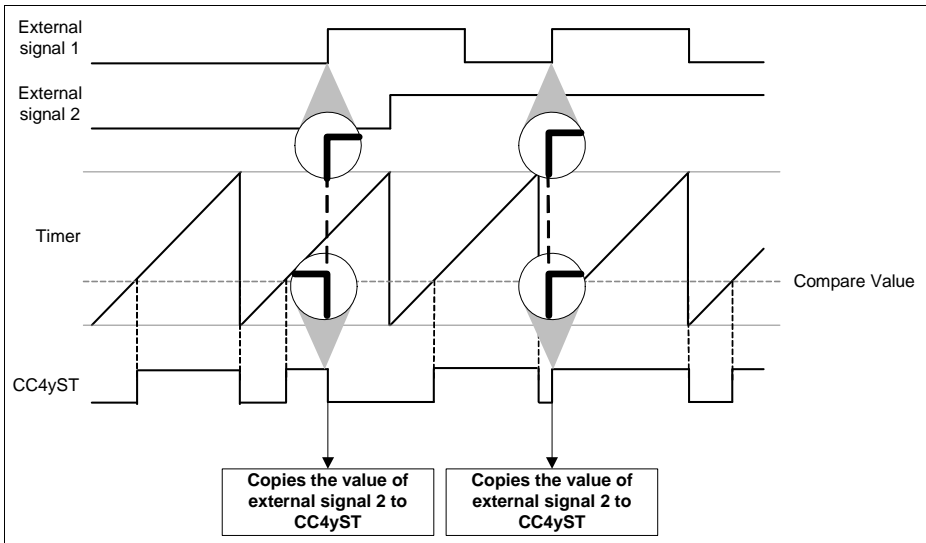


Figure 21-40 Status bit override

21.2.8 Multi-Channel Control

The multi channel control mode is selected individually in each slice by setting the **CC4yTC.MCME** = 1_B.

Within this mode, the output state of the Timer Slices (the ones set in multi channel mode) can be controlled in parallel by a single pattern.

The pattern is controlled via the CCU4 inputs, CCU4x.MCI0, CCU4x.MCI1, CCU4x.MCI2 and CCU4x.MCI3. Each of these inputs is connected directly to the associated slice input, e.g. CCU4x.MCI0 to CC40MCI, CCU4x.MCI1 to CC41MCI.

This pattern can be controlled directly by other module. The connectivity of each device may allow different control possibilities therefore one should address **Section 21.8** to check what modules are connected to these inputs.

When using the Multi Channel support of the CCU4, one can achieve a complete synchronicity between the output state update, CCU4x.OUTy, and the update of a new pattern, **Figure 21-41**. This synchronicity feature can be enabled by using some specific modules and therefore one should address **Section 21.8** to check which module is controlling the CCU4x.MCIy inputs.

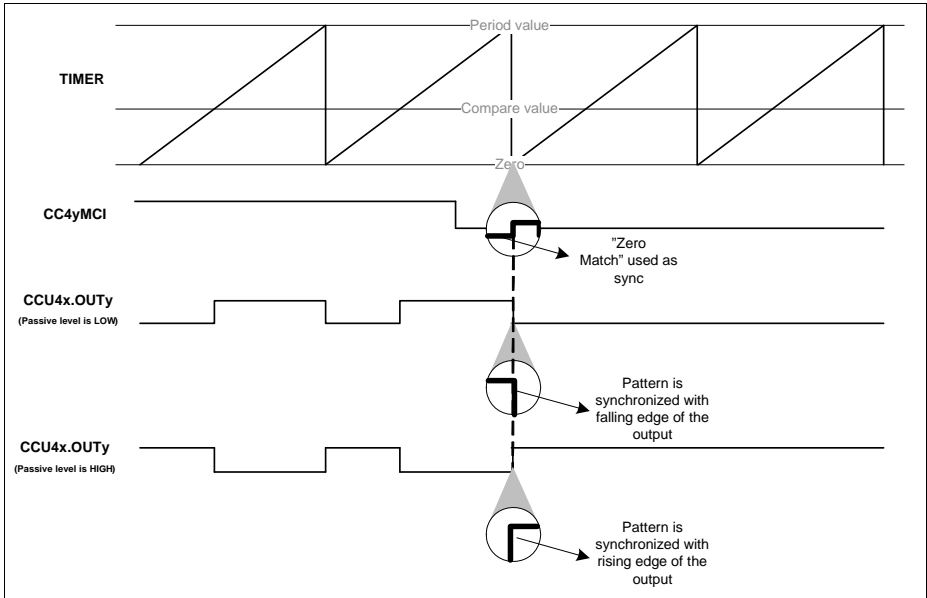


Figure 21-41 Multi channel pattern synchronization

Figure 21-42 shows the usage of the multi channel mode in conjunction with all four Timer Slices inside the CCU4.

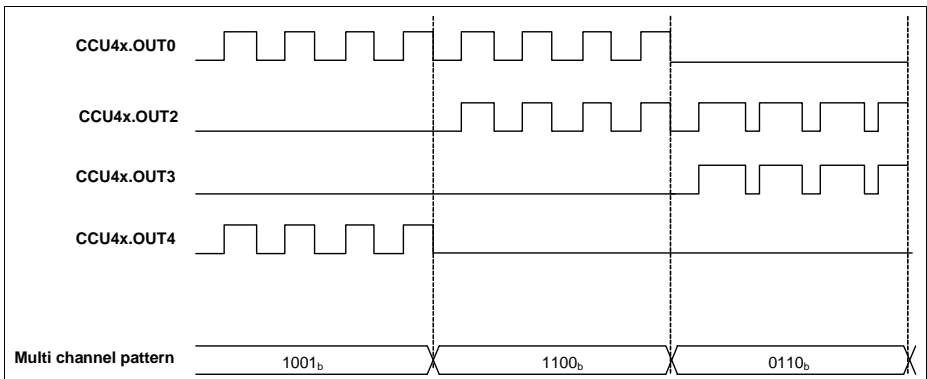


Figure 21-42 Multi Channel mode for multiple Timer Slices

The synchronization between the CCU4 and the module controlling the multi-channel pattern is achieved, by adding a 3 cycle delay on the output path of each Timer Slice

Capture/Compare Unit 4 (CCU4)

(between the status bit, CC4yST and the direct control of the output pin). This path is only selected when **CC4yTC.MCME** = 1_B, see **Figure 21-43**.

The multi pattern input synchronization can be seen on **Figure 21-44**. To achieve a synchronization between the update of the status bit, the sampling of a new multi channel pattern input is controlled by the period match or one match signal.

In a straightforward utilization of this synchronization feature, the module controlling the multi channel pattern signals, receives a sync signal from the CCU4, the CCU4x.PSy. This signal is then used by this module to update the multi-channel pattern. Due to the structure of the synchronization scheme inside the CCU4, the module controlling the multi-channel pattern needs to update this pattern, within 4 clock cycles after the CCU4x.PSy signal is asserted, **Figure 21-44**.

In a normal operation, where no external signal is used to control the counting direction, the signal used to enable the sampling of the pattern is always the period match when in edge aligned and the one match when in center aligned mode. When an external signal is used to control the counting direction, depending if the counter is counting up or counting down, the period match or the one match signal is used, respectively.

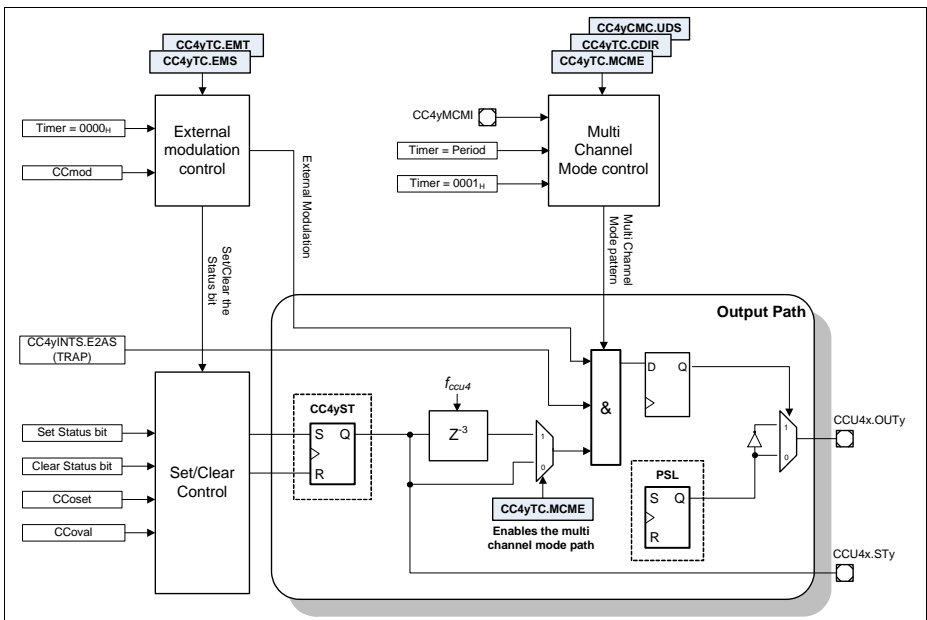


Figure 21-43 CC4y Status bit and Output Path

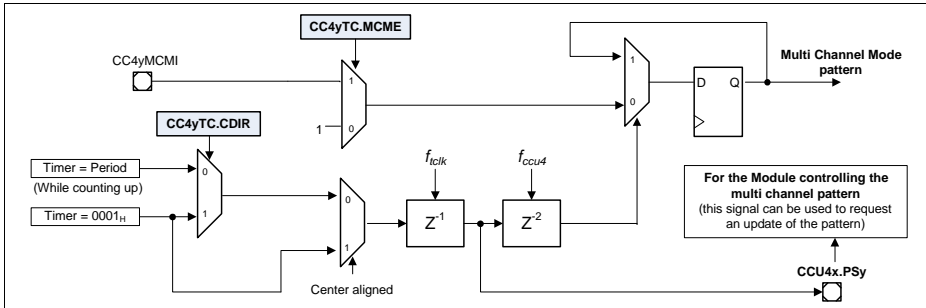


Figure 21-44 Multi Channel Mode Control Logic

21.2.9 Timer Concatenation

The CCU4 offers a very easy mechanism to perform a synchronous timer concatenation. This functionality can be used by setting the **CC4yCMC.TCE** = 1_B . By doing this the user is doing a concatenation of the actual CCU4 slice with the previous one, see [Figure 21-45](#).

Notice that it is not possible to perform concatenation with non adjacent slices and that timer concatenation automatically sets the slice mode into Edge Aligned. It is not possible to perform timer concatenation in Center Aligned mode.

To enable a 64 bit timer, one should set the **CC4yCMC.TCE** = 1_B in all the slices (with the exception of the CC40 due to the fact that it doesn't contain this control register).

To enable a 48 bit timer, one should set the **CC4yCMC.TCE** = 1_B in two adjacent slices and to enable a 32 bit timer, the **CC4yCMC.TCE** is set to 1_B in the slice containing the MSBs. Notice that the timer slice containing the LSBs should always have the TCE bitfield set to 0_B .

Several combinations for timer concatenation can be made inside a CCU4 module:

- one 64 bit timer
- one 48 bit timer plus a 16 timer
- two 32 bit timers
- one 32 bit timer plus two 16 bit timers

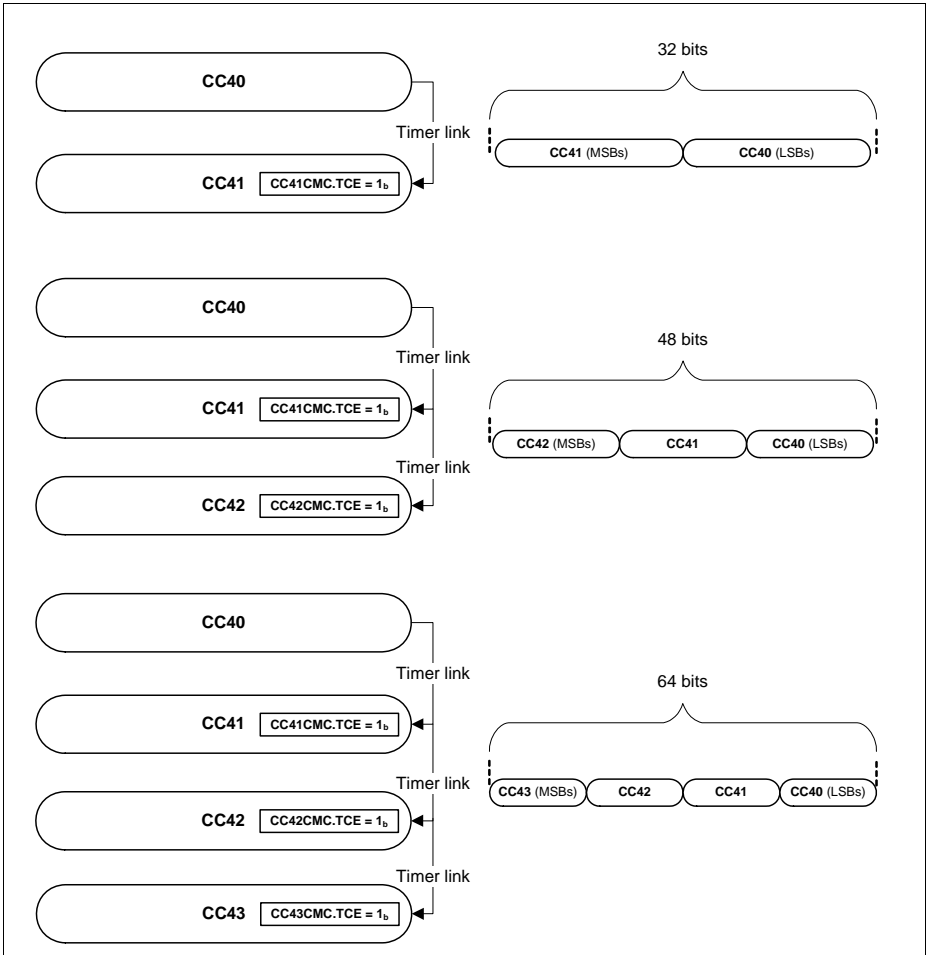


Figure 21-45 Timer Concatenation Example

Each Timer Slice is connected to the adjacent Timer Slices via a dedicated concatenation interface. This interface allows the concatenation of not only the Timer counting operation, but also a synchronous input trigger handling for capturing and loading operations, [Figure 21-46](#).

Note: For all cases CC40 and CC43 are not considered adjacent slices

Capture/Compare Unit 4 (CCU4)

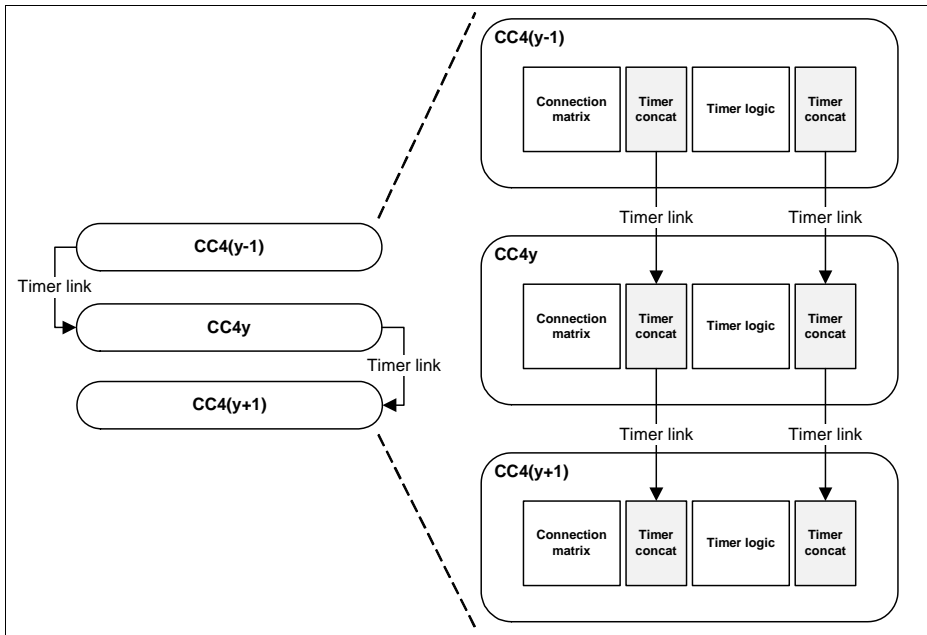


Figure 21-46 Timer Concatenation Link

Seven signals are present in the timer concatenation interface:

- Timer Period match (CC4yPM)
- Timer Zero match (CC4yZM)
- Timer Compare match (CC4yCM)
- Timer counting direction function (CCupd)
- Timer load function (CCload)
- Timer capture function for CC4yC0V and CC4yC1V registers (CCcap0)
- Timer capture function for CC4yC2V and CC4yC3V registers (CCcap1)

The first four signals are used to perform the synchronous timing concatenation at the output of the Timer Logic, like it is seen in [Figure 21-46](#). With this link, the timer length can be easily adjusted to 32, 48 or 64 bits (counting up or counting down).

The last three signals are used to perform a synchronous link between the capture and load functions, for the concatenated timer system. This means that the user can have a capture or load function programmed in the first Timer Slice, and propagate this capture or load trigger synchronously from the LSBs until the MSBs, [Figure 21-47](#).

The capture or load function only needs to be configured in the first Timer Slice (the one holding the LSBs). From the moment that **CC4yCMC.TCE** is set to 1_B, in the following Timer Slices, the link between these functions is done automatically by the hardware.

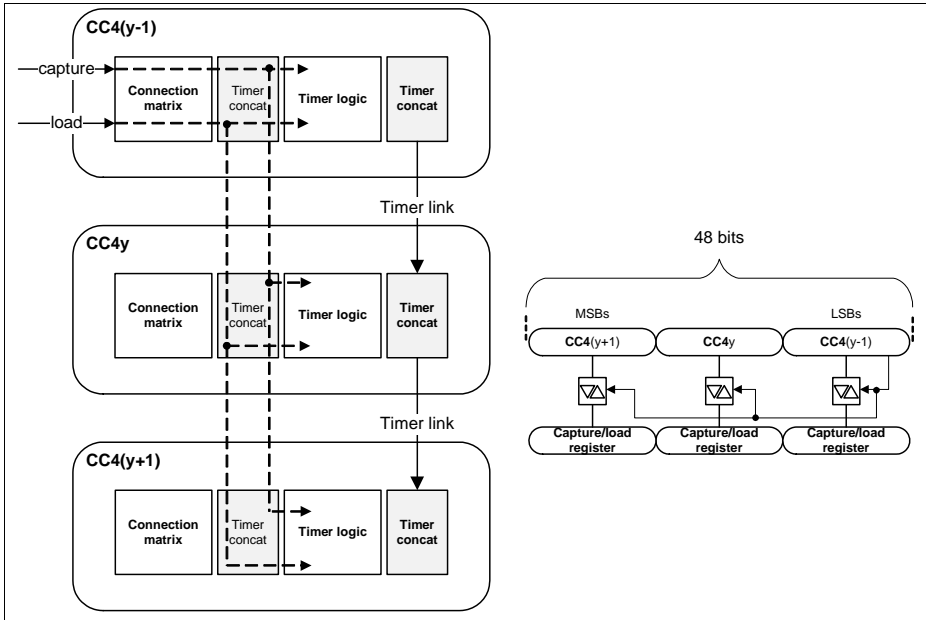


Figure 21-47 Capture/Load Timer Concatenation

The period match (CC4yPM) or zero match (CC4yZM) from the previous Timer Slice (with the immediately next lower index) are used in concatenated mode, as gating signal for the counter. This means that the counting operation of the MSBs only happens when a wrap around condition is detected, avoiding additional DSP operations to extract the counting value.

With the same methodology, the compare match (CC4yCM), zero match and period match are gated with the specific signals from the previous slice. This means that the timing information is propagated throughout all the slices, enabling a completely synchronous match between the LSB and MSB count, see [Figure 21-48](#).

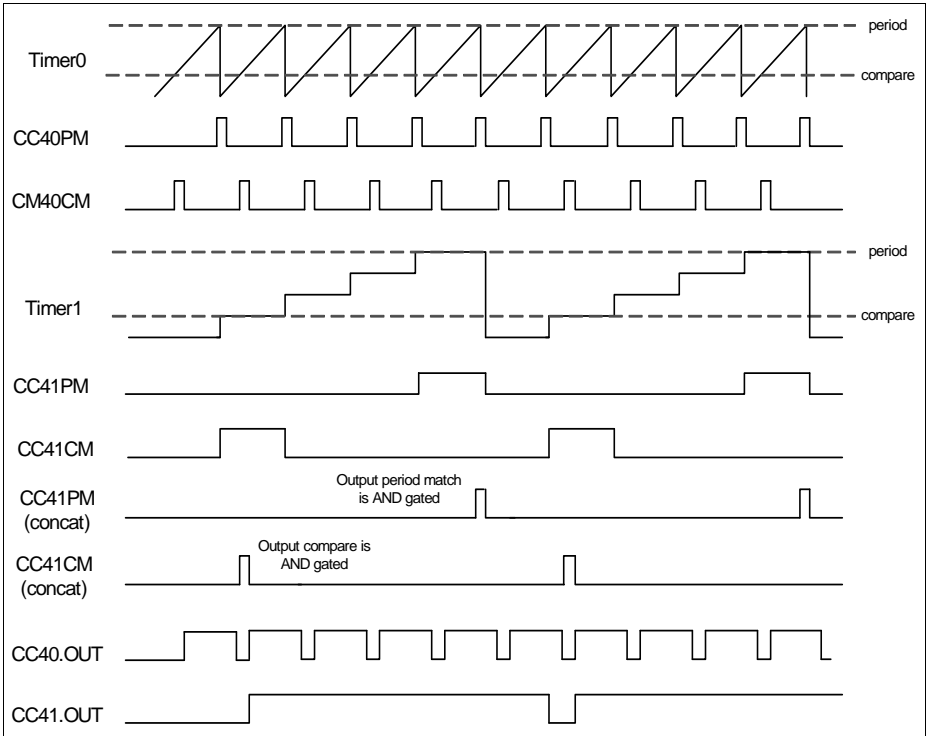


Figure 21-48 32 bit concatenation timing diagram

Note: The counting direction of the concatenated timer needs to be fixed. The timer can count up or count down, but the direction cannot be updated on the fly.

Figure 21-49 gives an overview of the timer concatenation logic. Notice that all the mechanism is controlled solely by the **CC4yCMC.TCE** bitfield.

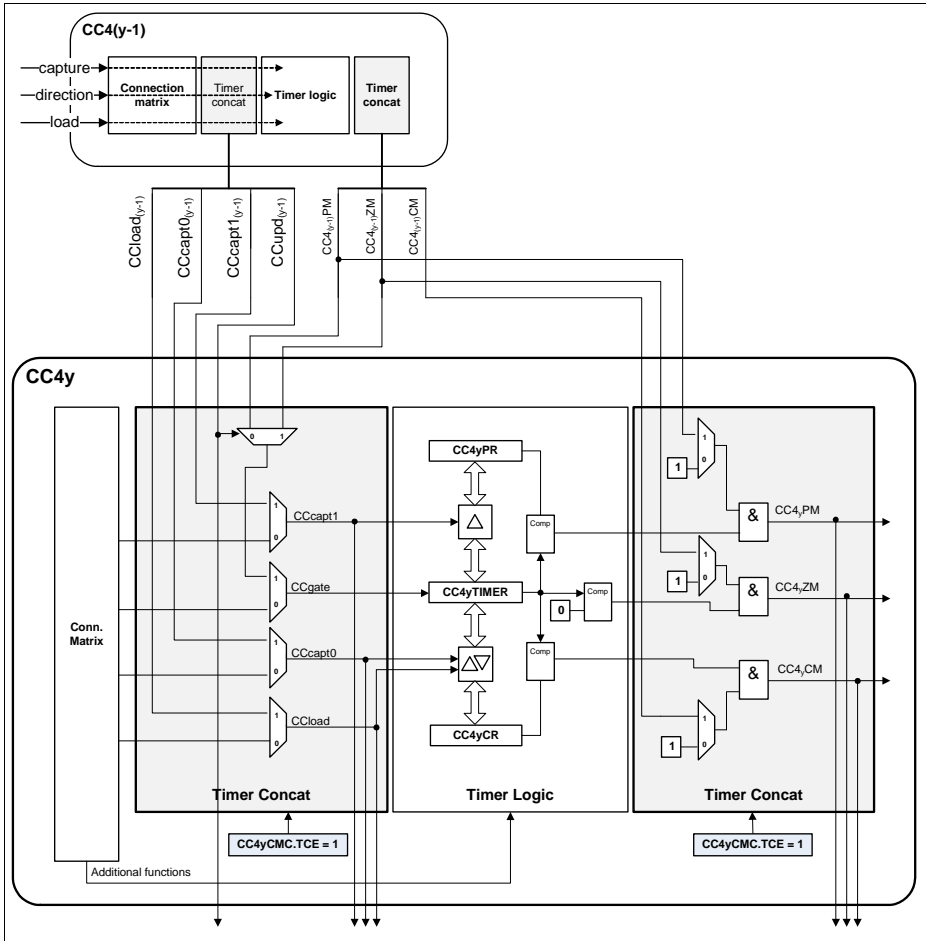


Figure 21-49 Timer concatenation control logic

21.2.10 PWM Dithering

The CCU4 has an automatic PWM dithering insertion function. This functionality can be used with very slow control loops that cannot update the period/compare values in a fast manner, and by that fact the loop can lose precision on long runs. By introducing dither on the PWM signal, the average frequency/duty cycle is then compensated against that error.

Each slice contains a dither control unit, see [Figure 21-50](#).

Capture/Compare Unit 4 (CCU4)

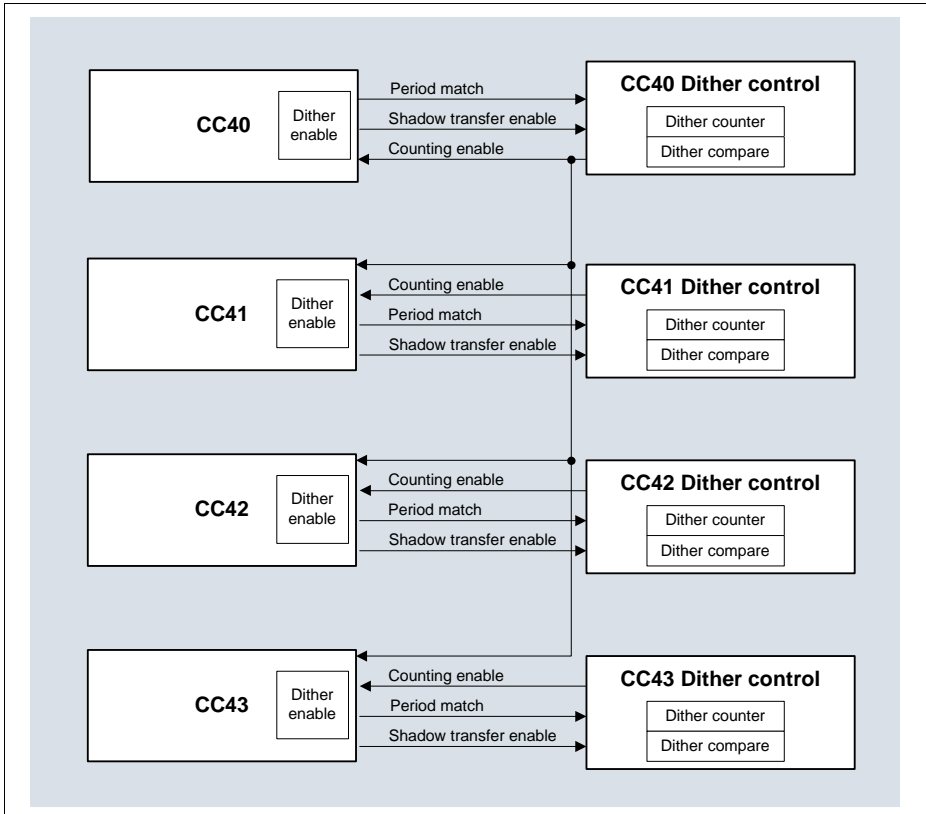


Figure 21-50 Dither structure overview

The dither control unit contains a 4 bit counter and a compare value. The four bit counter is incremented every time that a period match occurs. The counter works in a bit reverse mode so the distribution of increments stays uniform over 16 counter periods, see [Table 21-5](#).

Table 21-5 Dither bit reverse counter

counter[3]	counter[2]	counter[1]	counter[0]
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	0
0	0	0	1
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	1

The counter is then compared against a programmed value, **CC4yDIT.DCV**. If the counter value is smaller than the programmed value, a gating signal is generated that can be used to extend the period, to delay the compare or both (controlled by the **CC4yTC.DITHE** field, see **Table 21-6**) for one clock cycle.

Table 21-6 Dither modes

DITHE[1]	DITH[0]	Mode
0	0	Dither is disabled
0	1	Period is increased by 1 cycle
1	0	Compare match is delayed by 1 cycle
1	1	Period is increased by 1 cycle and compare is delayed by 1 cycle

The dither compare value also has an associated shadow register that enables concurrent update with the period/compare register of CC4y. The control logic for the dithering unit is represented on **Figure 21-51**.

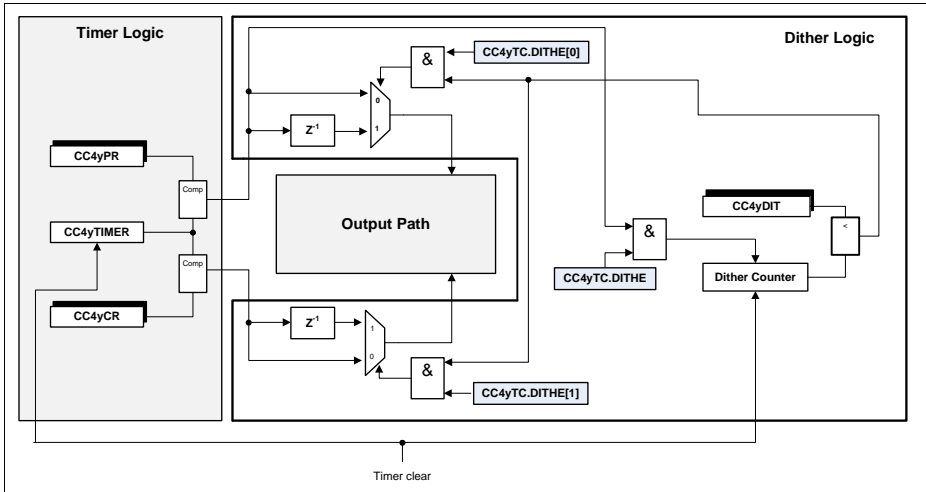


Figure 21-51 Dither control logic

Figure 21-52 to **Figure 21-57** show the effect of the different configurations for the dithering function, **CC4yTC.DITHE**, for both counting schemes, Edge and Center Aligned mode. In each figure, the bit reverse scheme is represented for the dither counter and the compare value was programmed with the value 8_H . In each figure, the variable T , represents the period of the counter, while the variable d indicates the duty cycle (status bit is set HIGH).

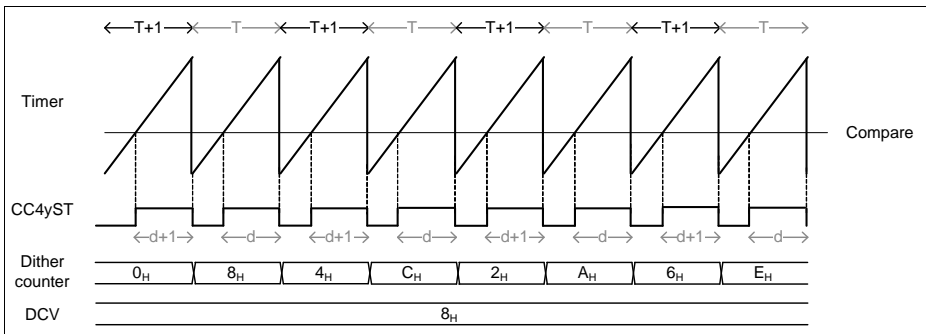


Figure 21-52 Dither timing diagram in edge aligned - **CC4yTC.DITHE = 01_B**

Capture/Compare Unit 4 (CCU4)

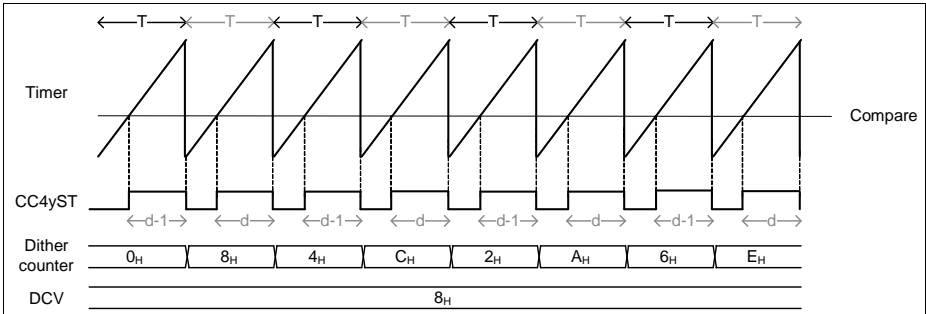


Figure 21-53 Dither timing diagram in edge aligned - $CC4yTC.DITHE = 10_B$

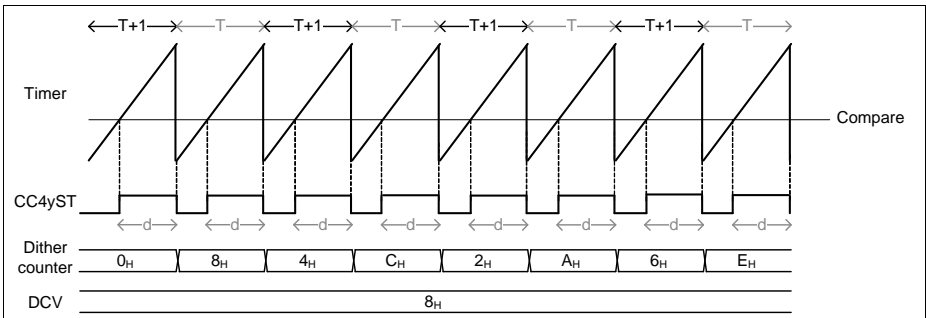


Figure 21-54 Dither timing diagram in edge aligned - $CC4yTC.DITHE = 11_B$

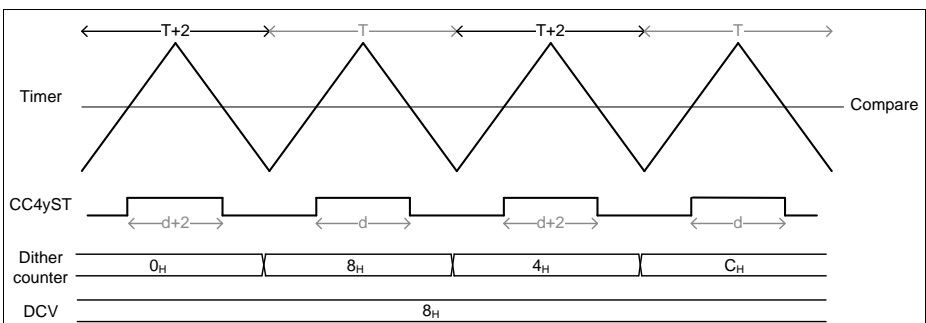


Figure 21-55 Dither timing diagram in center aligned - $CC4yTC.DITHE = 01_B$

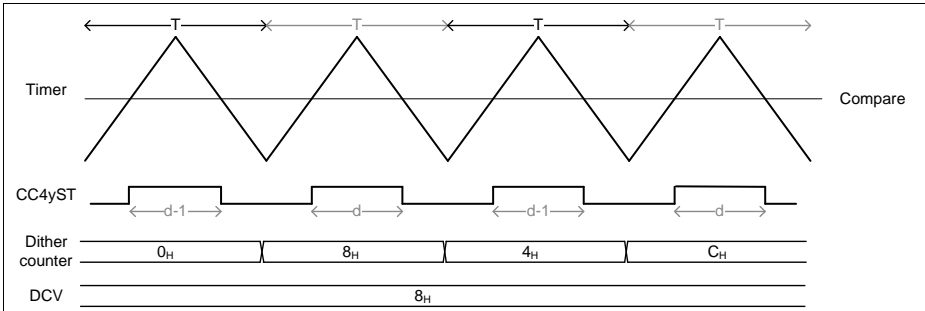


Figure 21-56 Dither timing diagram in center aligned - **CC4yTC.DITHE = 10_B**

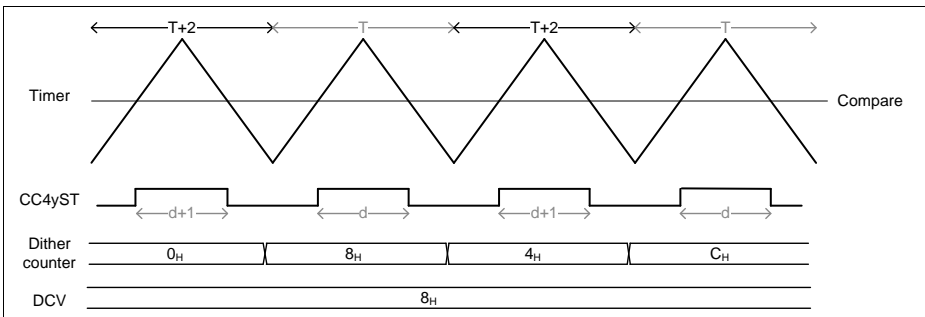


Figure 21-57 Dither timing diagram in center aligned - **CC4yTC.DITHE = 11_B**

Note: When using the dither, is not possible to select a period value of FS when in edge aligned mode. In center aligned mode, the period value must be at least $FS - 2$.

21.2.11 Prescaler

The CCU4 contains a 4 bit prescaler that can be used in two operating modes for each individual slice:

- normal prescaler mode
- floating prescaler mode

The run bit of the prescaler can be set/cleared by SW by writing into the registers, **GIDLC.SPRB** and **GIDLS.CPRB** respectively, and it can also be cleared by the run bit of a specific slice. With the last mechanism, the run bit of the prescaler is cleared one clock cycle after the clear of the run bit of the selected slice. To select which slice can perform this action, one should program the **GCTRL.PRBC** register.

21.2.11.1 Normal Prescaler Mode

In Normal prescaler mode the clock fed to the CC4y counter is a normal fixed division by N, accordingly to the value set in the **CC4yPSC**.PSIV register. The values for the possible division values are listed in **Table 21-7**. The **CC4yPSC**.PSIV value is only modified by a SW access. Notice that each slice has a dedicated prescaler value selector (**CC4yPSC**.PSIV), which means that the user can select different counter clocks for each Timer Slice (CC4y).

Table 21-7 Timer clock division options

CC4yPSC .PSIV	Resulting clock
0000 _B	f_{CCU4}
0001 _B	$f_{CCU4}/2$
0010 _B	$f_{CCU4}/4$
0011 _B	$f_{CCU4}/8$
0100 _B	$f_{CCU4}/16$
0101 _B	$f_{CCU4}/32$
0110 _B	$f_{CCU4}/64$
0111 _B	$f_{CCU4}/128$
1000 _B	$f_{CCU4}/256$
1001 _B	$f_{CCU4}/512$
1010 _B	$f_{CCU4}/1024$
1011 _B	$f_{CCU4}/2048$
1100 _B	$f_{CCU4}/4096$
1101 _B	$f_{CCU4}/8192$
1110 _B	$f_{CCU4}/16384$
1111 _B	$f_{CCU4}/32768$

21.2.11.2 Floating Prescaler Mode

The floating prescaler mode can be used individually in each slice by setting the register **CC4yTC**.FPE = 1_B. With this mode, the user can not only achieve a better precision on the counter clock for compare operations but also reduce the SW read access for the capture mode.

The floating prescaler mode contains additionally to the initial configuration value register, **CC4yPSC**.PSIV, a compare register, **CC4yFPC**.PCMP with an associated shadow register mechanism.

Capture/Compare Unit 4 (CCU4)

Figure 21-58 shows the structure of the prescaler in floating mode when the specific slice is in compare mode (no external signal is used for capture). In this mode, the value of the clock division is incremented by 1_D every time that a timer overflow/underflow (overflow if in Edge Aligned Mode, underflow if in Center Aligned Mode) occurs.

In this mode, the Compare Match from the timer is ANDed with the Compare Match of the prescaler and every time that this event occurs, the value of the clock division is updated with the **CC4yPSC.PSIV** value in the immediately next timer overflow/underflow event.

The shadow transfer of the floating prescaler compare value, **CC4yFPC.PCMP**, is done following the same rules described on **Section 21.2.5.2**.

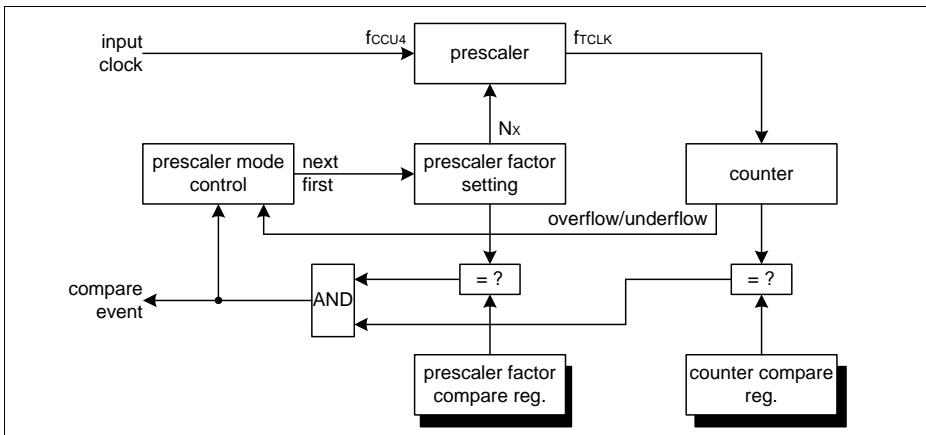


Figure 21-58 Floating prescaler in compare mode overview

When the specific CC4y is operating in capture mode (when at least one external signal is decoded as capture functionality), the actual value of the clock division also needs to be stored every time that a capture event occurs. The floating prescaler can have up to 4 capture registers (the maximum number of capture registers is dictated by the number of capture registers used in the specific slice).

The clock division value continues to be incremented by 1_D every time that a timer overflow (in capture mode, the slice is always operating in Edge Aligned Mode) occurs and it is loaded with the PSIV value every time that a capture triggers is detected.

See the **Section 21.2.12.2** for a full description of the usage of the floating prescaler mode in conjunction with compare and capture modes.

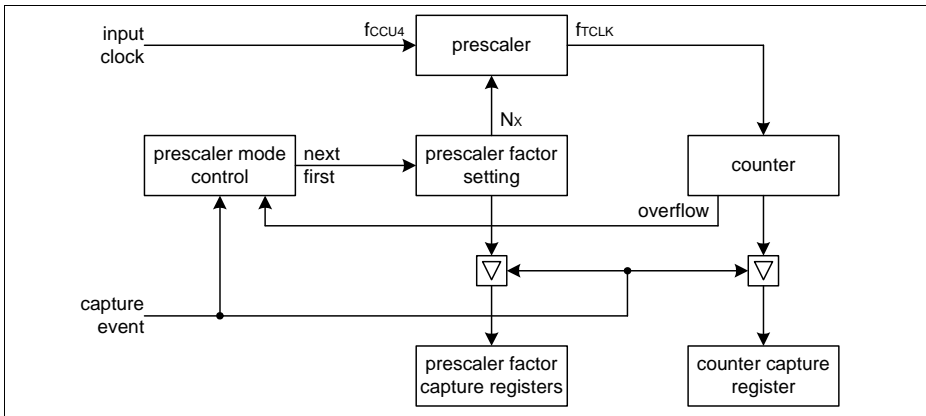


Figure 21-59 Floating Prescaler in capture mode overview

21.2.12 CCU4 Usage

21.2.12.1 PWM Signal Generation

The CCU4 offers a very flexible range in duty cycle configurations. This range is comprised between 0 to 100%.

To generate a PWM signal with a 100% duty cycle in Edge Aligned Mode, one should program the compare value, **CC4yCR.CR**, to 0000_H, see [Figure 21-60](#).

In the same manner a 100% duty cycle signal can be generated in Center Aligned Mode, see [Figure 21-61](#).

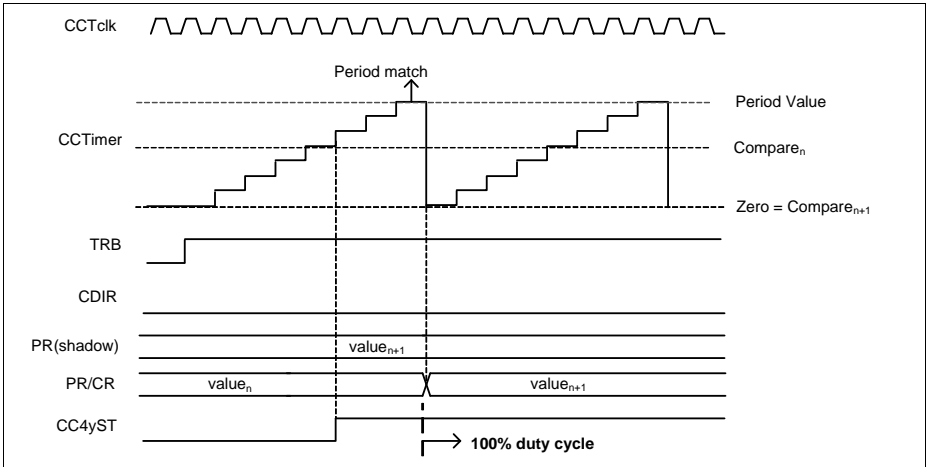


Figure 21-60 PWM with 100% duty cycle - Edge Aligned Mode

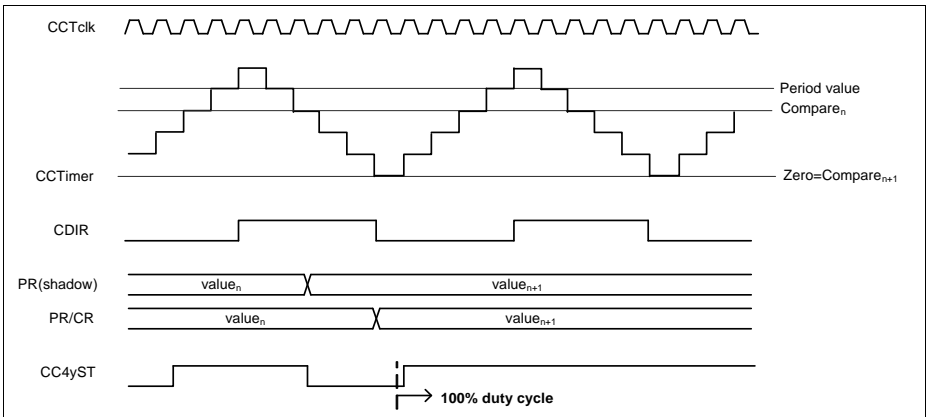


Figure 21-61 PWM with 100% duty cycle - Center Aligned Mode

To generate a PWM signal with 0% duty cycle in Edge Aligned Mode, the compare register should be set with the value programmed into the period value plus 1. In the case that the timer is being used with the full 16 bit capability (counting from 0 to 65535), setting a value bigger than the period value into the compare register is not possible and therefore the smallest duty cycle that can be achieved is 1/FS, see [Figure 21-62](#).

In Center Aligned Mode, the counter is never running from 0 to 65535_D, due to the fact that it has to overshoot for one clock cycle the value set in the period register. Therefore the user never has a FS counter, which means that generating a 0% duty cycle signal is

Capture/Compare Unit 4 (CCU4)

always possible by setting a value in the compare register bigger than the one programmed into the period register, see [Figure 21-63](#).

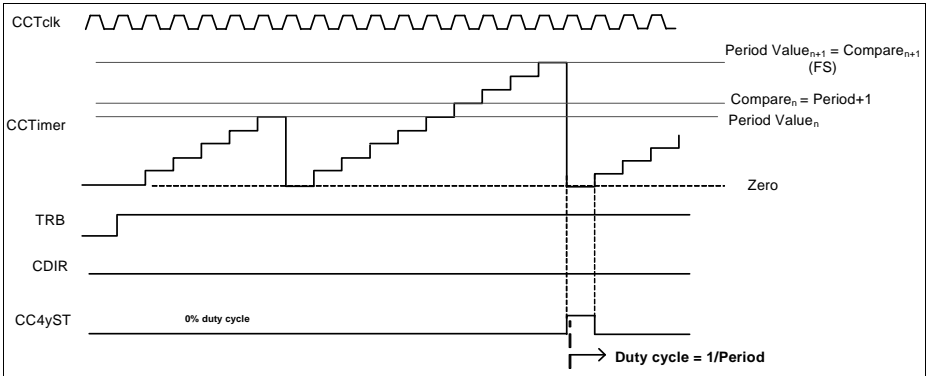


Figure 21-62 PWM with 0% duty cycle - Edge Aligned Mode

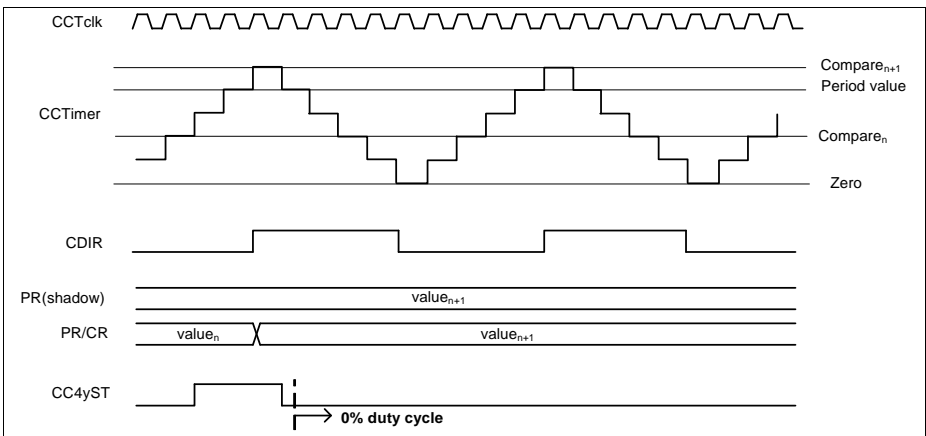


Figure 21-63 PWM with 0% duty cycle - Center Aligned Mode

21.2.12.2 Prescaler Usage

In Normal Prescaler Mode, the frequency of the f_{clk} fed to the specific CC4y is chosen from the [Table 21-7](#), by setting the [CC4yPSC.PSIV](#) with the required value.

In Floating Prescaler Mode, the frequency of the f_{clk} can be modified over a selected timeframe, within the values specified in [Table 21-7](#). This mechanism is specially useful if, when in capture mode, the dynamic of the capture triggers is very slow or unknown.

Capture/Compare Unit 4 (CCU4)

In Capture Mode, the Floating Prescaler value is incremented by 1 every time that a timer overflow happens and it is set with the initial programmed value when a capture event happens, see **Figure 21-64**.

When using the Floating Prescaler Mode in Capture Mode, the timer should be cleared each time that a capture event happens, **CC4yTC.CAPC** = 11_B. By operating the Capture mode in conjunction with the Floating Prescaler, even for capture signals that have a periodicity bigger than 16 bits, it is possible to use just a single CCU4 Timer Slice without monitoring the interrupt event of the timer overflow, cycle by cycle. For this the user just needs to know what is the timer captured value and the actual prescaler configuration at the time that the capture event occurred. These values are contained in each CC4yCxV register.

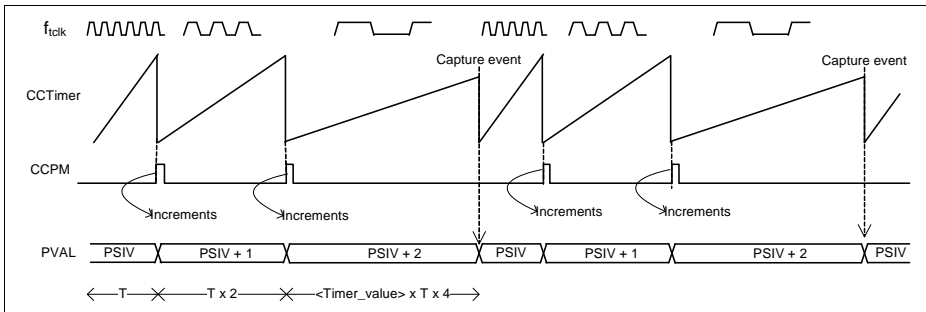


Figure 21-64 Floating Prescaler capture mode usage

When in Compare Mode, the Floating Prescaler function may be used to achieve a fractional PWM frequency or to perform some frequency modulation.

The same incrementing by 1_D mechanism is done every time that an overflow/underflow of the Timer occurs and the actual Prescaler value, doesn't match the one programmed into the **CC4yFPC.PCMP** register.

When a Compare Match from the Timer occurs and the actual Prescaler value is equal to the one programmed on the **CC4yFPC.PCMP** register, then the Prescaler value is set with the initial value, **CC4yPSC.PSIV**, when the next occurrence of a timer overflow/underflow.

In **Figure 21-65**, the Compare value of the Floating Prescaler was set to PSIV + 2. Every time that a timer overflow occurs, the value of the Prescaler is incremented by 1, which means that if we give f_{tclk} as the reference frequency for the **CC4yPSC.PSIV** value, we have $f_{tclk}/2$ for **CC4yPSC.PSIV + 1** and $f_{tclk}/4$ for **CC4yPSC.PSIV + 2**.

The period over time of the counter becomes:

$$Period = (1/f_{tclk} + 2/f_{tclk} + 4/f_{tclk}) / 3 \quad (21.10)$$

Capture/Compare Unit 4 (CCU4)

The same mechanism is used in Center Aligned Mode, but to keep the rising arcade and falling arcade always symmetrical, instead of the overflow of the timer, the underflow is used, see [Figure 21-66](#).

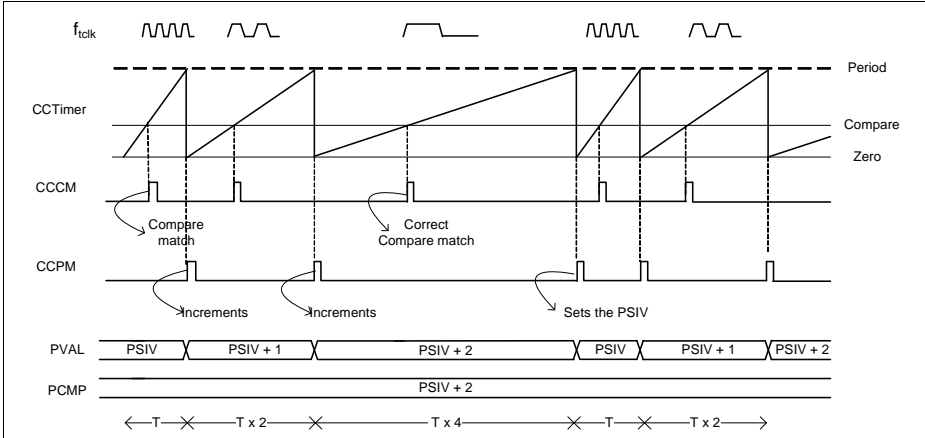


Figure 21-65 Floating Prescaler compare mode usage - Edge Aligned

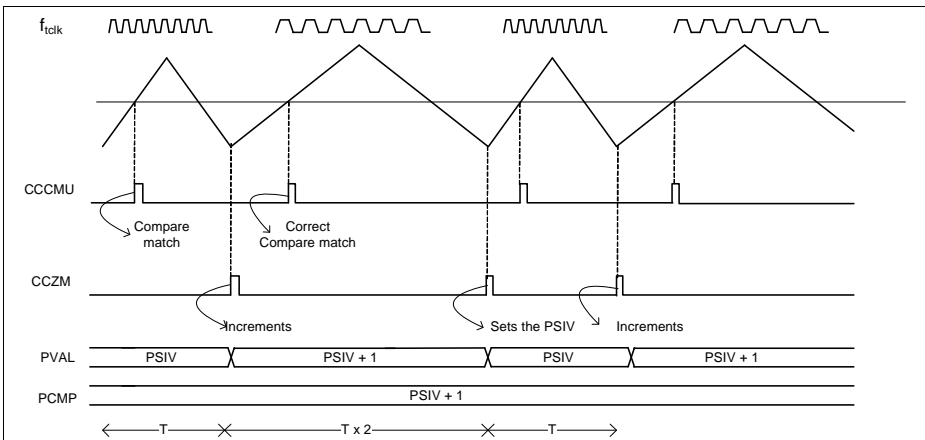


Figure 21-66 Floating Prescaler compare mode usage - Center Aligned

21.2.12.3 PWM Dither

The Dither functionality can be used to achieve a very fine precision on the periodicity of the output state in compare mode. The value set in the dither compare register, **CC4yDIT.DCV** is crosschecked against the actual value of the dither counter and every

Capture/Compare Unit 4 (CCU4)

time that the dither counter is smaller than the comparison value one of the follows actions is taken:

- The period is extended for 1 clock cycle - **CC4yTC.DITHE** = 01_B; in edge aligned mode
- The period is extended for 2 clock cycles - **CC4yTC.DITHE** = 01_B; in center aligned mode
- The comparison match while counting up (**CC4yTCST.CDIR** = 0_B) is delayed (this means that the status bit is going to stay in the SET state 1 cycle less) for 1 clock cycle - **CC4yTC.DITHE** = 10_B;
- The period is extended for 1 clock cycle and the comparison match while counting up is delayed for 1 clock cycle - **CC4yTC.DITHE** = 11_B; in edge aligned mode
- The period is extended for 2 clock cycles and the comparison match while counting up is delayed for 1 clock cycle; center aligned mode

The bit reverse counter distributes the number programmed in the **CC4yDIT.DCV** throughout a window of 16 timer periods.

Table 21-8 describes the bit reverse distribution versus the programmed value on the **CC4yDIT.DCV** field. The fields marked as '0' indicate that in that counter period, one of the above described actions, is going to be performed.

Table 21-8 Bit reverse distribution

Dither counter	DCV															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
4	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
C	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
2	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
6	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
E	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
5	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
3	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
B	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0

Table 21-8 Bit reverse distribution (cont'd)

Dither counter	DCV															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
7	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
F	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The bit reverse distribution versus the programmed **CC4yDIT.DCV** value results in the following values for the Period and duty cycle:

$$\text{DITHE} = 01_{\text{B}}$$

$$\text{Period} = [(16 - \text{DCV}) \times T + \text{DCV} \times (T + 1)]/16; \text{ in Edge Aligned Mode} \quad (21.11)$$

$$\text{Duty cycle} = [(16 - \text{DCV}) \times d/T + \text{DCV} \times (d+1)/(T + 1)]/16; \text{ in Edge Aligned Mode} \quad (21.12)$$

$$\text{Period} = [(16 - \text{DCV}) \times T + \text{DCV} \times (T + 2)]/16; \text{ in Center Aligned Mode} \quad (21.13)$$

$$\text{Duty cycle} = [(16 - \text{DCV}) \times d/T + \text{DCV} \times (d+2)/(T + 2)]/16; \text{ in Center Aligned Mode} \quad (21.14)$$

$$\text{DITHE} = 10_{\text{B}}$$

$$\text{Period} = T ; \text{ in Edge Aligned Mode} \quad (21.15)$$

$$\text{Duty cycle} = [(16 - \text{DCV}) \times d/T + \text{DCV} \times (d-1)/T]/16; \text{ in Edge Aligned Mode} \quad (21.16)$$

$$\text{Period} = T ; \text{ in Center Aligned Mode} \quad (21.17)$$

$$\text{Duty cycle} = [(16 - \text{DCV}) \times d/T + \text{DCV} \times (d-1)/T]/16; \text{ in Center Aligned Mode} \quad (21.18)$$

$$\text{DITHE} = 11_{\text{B}}$$

$$\text{Period} = [(16 - \text{DCV}) \times T + \text{DCV} \times (T + 1)]/16; \text{ in Edge Aligned Mode} \quad (21.19)$$

$$\text{Duty cycle} = [(16 - \text{DCV}) \times d/T + \text{DCV} \times d/(T + 1)]/16; \text{ in Edge Aligned Mode} \quad (21.20)$$

$$\text{Period} = [(16 - \text{DCV}) \times T + \text{DCV} \times (T + 2)]/16; \text{ in Center Aligned Mode} \quad (21.21)$$

$$\text{Duty cycle} = [(16 - \text{DCV}) \times d/T + \text{DCV} \times (d+1)/(T + 2)]/16; \text{ in Center Aligned Mode} \quad (21.22)$$

where:

T - Original period of the signal, see [Section 21.2.5.1](#)

d - Original duty cycle of the signal, see [Section 21.2.5.1](#)

21.2.12.4 Capture Mode Usage

Each Timer Slice can make use of 2 or 4 capture registers. Using only 2 capture registers means that only 1 Event was linked to a captured trigger. To use the four capture registers, both capture triggers need to be mapped into an Event (it can be the same signal with different edges selected or two different signals) or the **CC4yTC.SCE** field needs to be set to 1, which enables the linking of the 4 capture registers.

The internal slice mechanism for capturing is the same for the capture trigger 1 or capture trigger 0.

Different Capture Events - SCE = 0_B

Capture trigger 1 (CCcapt1) is appointed to the capture register 2, **CC4yC2V** and capture register 3, **CC4yC3V**, while trigger 0 (CCcapt0) is appointed to capture register 1, **CC4yC1V** and 0, **CC4yC0V**.

In each CCcapt0 event, the timer value is stored into **CC4yC1V** and the value of the **CC4yC1V** is transferred into the **CC4yC0V**.

In each CCcapt1 event, the timer value is stored into capture register **CC4yC3V** and the value of the capture register **CC4yC3V** is transferred into **CC4yC2V**.

The capture/transfer mechanism only happens if the specific register is not full. A capture register becomes full when receives a new value and becomes empty after the SW has read back the value.

The full flag is cleared every time that the SW reads back the **CC4yC0V**, **CC4yC1V**, **CC4yC2V** or **CC4yC3V** register. The SW can be informed of a new capture trigger by enabling the interrupt source linked to the specific Event. This means that every time that a capture is made an interrupt pulse is generated.

In the case that the Floating Prescaler Mode is being used, the actual value of the clock division is also stored in the capture register (CC4yCxV).

Figure 21-67 shows an example of how the capture/transfer may be used in a Timer Slice that is using an external signal as count function (to measure the velocity of a rotating device), and an equidistant capture trigger that is used to dictate the timestamp for the velocity calculation (two Timer waveforms are plotted, one that exemplifies the clearing of the timer in each capture event and another without the clearing function active).

Capture/Compare Unit 4 (CCU4)

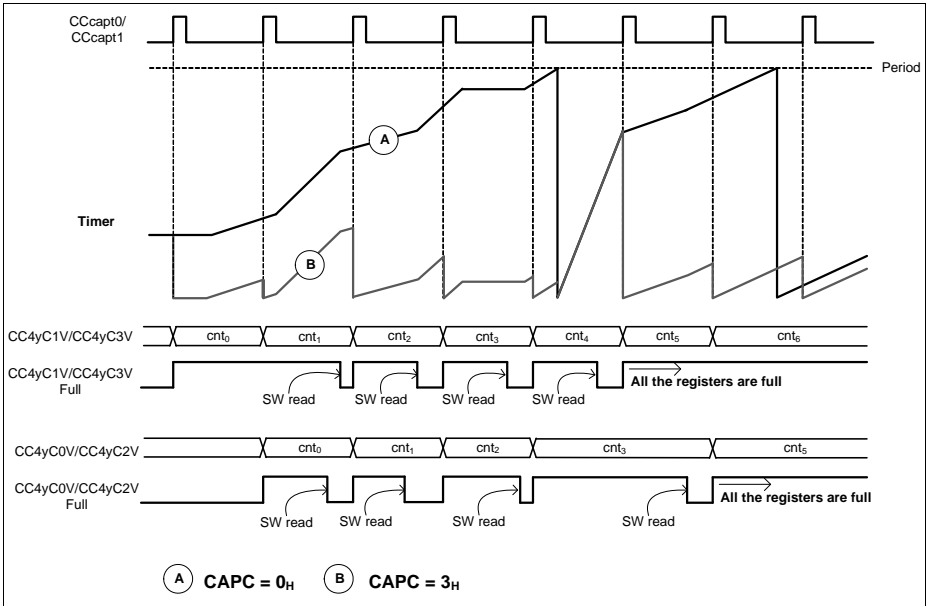


Figure 21-67 Capture mode usage - single channel

Same Capture Event - SCE = 1_B

If the **CC4yTC.SCE** is set to 1_B, all the four capture registers are chained together, emulating a fifo with a depth of 4. In this case, only the capture trigger 1, **CCcapt1**, is used to perform a capture event.

As an example for this mode, one can consider the case where one Timer Slice is being used in capture mode with SCE = 1_B, with another external signal that controls the counting. This timer slice can be incremented at different speeds, depending on the frequency of the counting signal.

An additional Timer Slice is used to control the capture trigger, dictating the time stamp for the capturing.

A simple scheme for this can be seen in **Figure 21-68**. The **CC40ST** output of slice 0 was used as capture trigger in the **CC41** slice (active on rising and falling edge). The **CC40ST** output is used as known timebase marker, while the slice timer used for capture is being controlled by external events, e.g. external count.

Due to the fact that we have available 4 capture registers, every time that the SW reads back the complete set of values, 3 speed profiles can be measured.

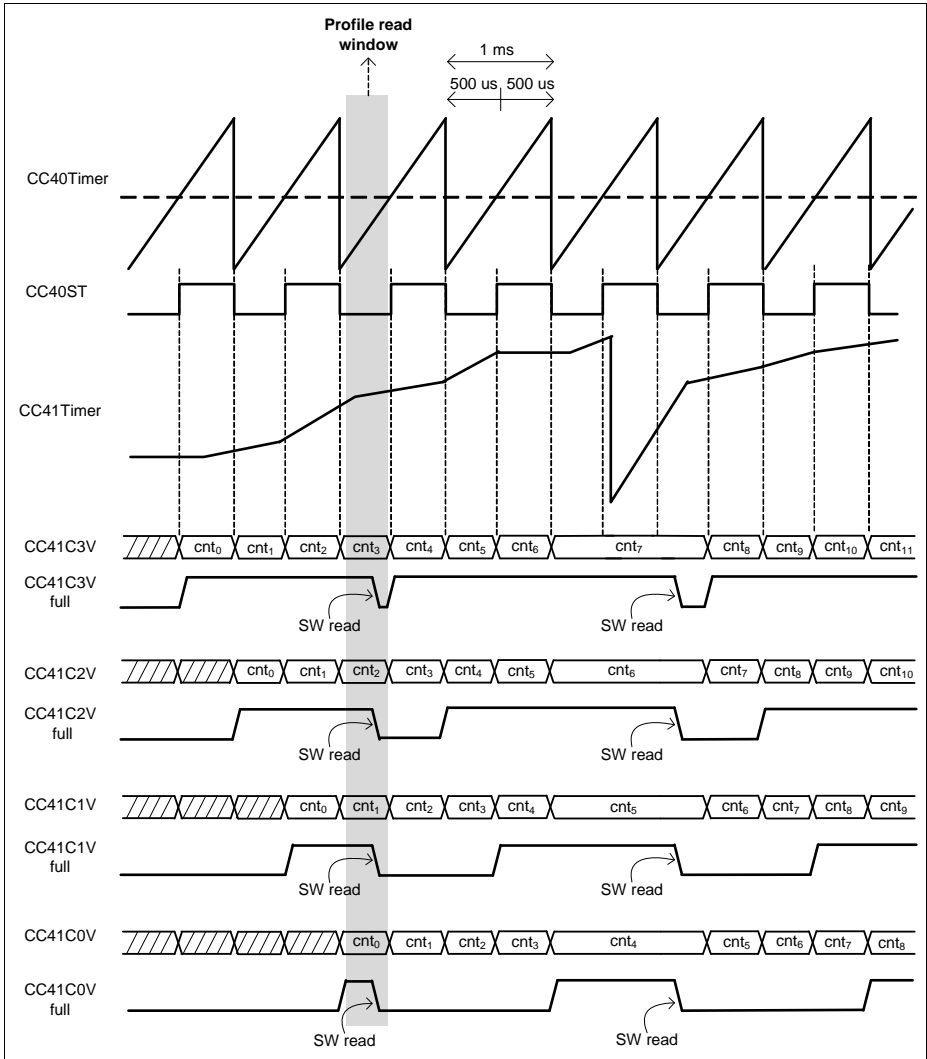


Figure 21-68 Three Capture profiles - $CC4yTC.SCE = 1_B$

To calculate the three different profiles in [Figure 21-68](#), the 4 capture registers need to be read during the pointed read window. After that, the profile calculation is done:

$$\text{Profile 1} = CC41C1V_{info} - CC41C0V_{info}$$

$$\text{Profile 2} = CC41C2V_{info} - CC41C1V_{info}$$

Profile 3= CC41C3V_{info} - CC41C2V_{info}

Note: This is an example and therefore several Timer Slice configurations and software loops can be implemented.

High Dynamics Capturing

In some cases the dynamics of the capture trigger(s) may vary greatly over time. This will impose that the software needs to be prepared for the worst case scenario, where the frequency of the capture triggers may be very high. In applications where cycle-by-cycle calculation is needed (calculation in each capture trigger), then this constraints needs to be met by the software. Nevertheless for applications where a cycle-by-cycle calculation is not needed, the software can read back the FIFO data register in a periodic base and fetch all the data that has been captured so far, **Figure 21-69**.

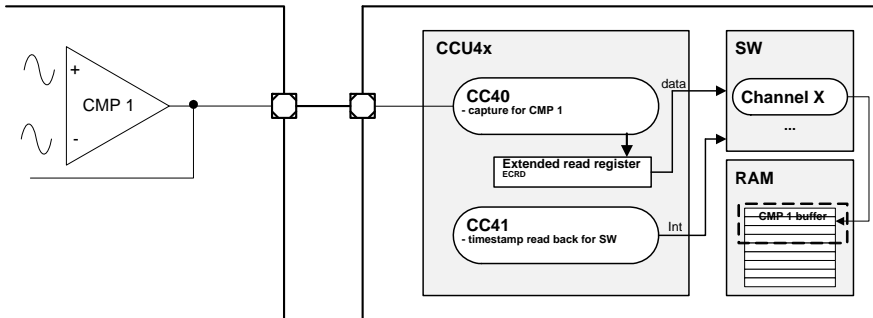


Figure 21-69 High dynamics capturing with software controlled timestamp

In this scenario, the software/CPU will read back the complete set of capture registers (2 or 4 depending on the chosen configuration), every time that an interrupt is triggered from the timestamp timer (the periodicity of this timer can also be adjusted on-the-fly).

Due to the fact that every capture register offers a full flag status bit, the software/CPU can always read back the complete set of registers. At the time of the data processing, this full flag is then checked, indicating if this value needs to be processed or not.

This FIFO read back functionality can also be used for applications that impose a heavy load on the system, which may not guarantee fixed access times to read back the captured data.

Capture/Compare Unit 4 (CCU4)

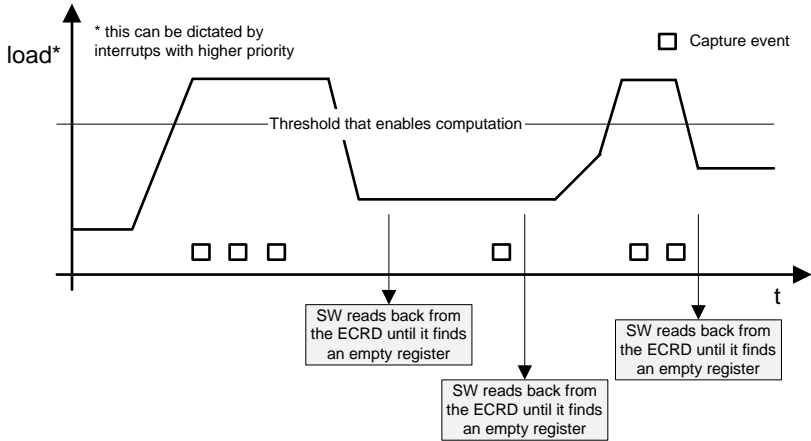


Figure 21-70 Extended read back during high load

Capture Grouping

In applications where multiple capture Timers are needed and the priority of the capture routines, does not imply that a cycle-by-cycle calculation needs to be done for every event, it may be suitable to group all the timers in the same CCU4x unit, [Figure 21-71](#).

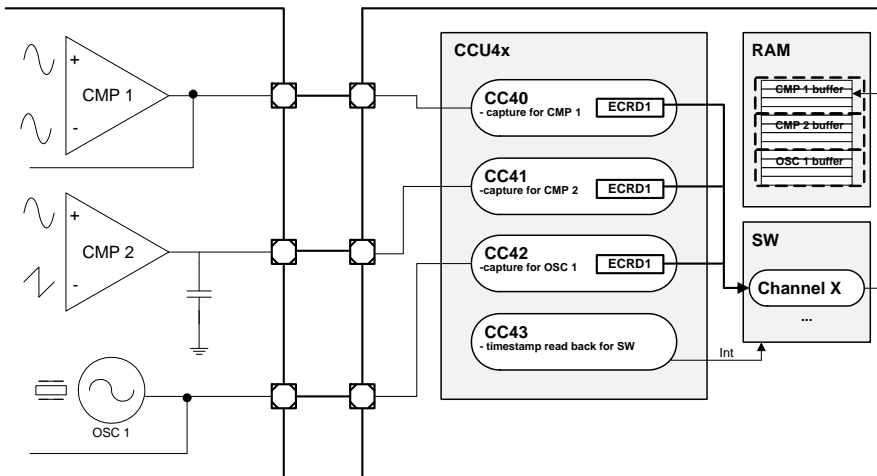


Figure 21-71 Capture grouping with extended read back

Capture/Compare Unit 4 (CCU4)

By setting the ECM bitfield for the Timer Slices used for capturing, the extended read back mode enables the reading back of data always in the proper capture order (from oldest to newest data). A timestamp timer is then used to trigger the Software/CPU to read back all the capture data present in the Timer Slices.

Every time that the interrupt is sensed, the Software/CPU (in this example) reads back the complete set of capture registers (via the ECRD address) for all Timer Slices. Due to the fact that each data read has a full flag indicator, the Software/CPU can read back the complete set of capture registers from all timers. This allows a fixed memory allocation that is as big as the number of captured registers, **Figure 21-72** (in this example 4 capture registers for each Timer Slice are being used).

The additional lost value bitfield (LCV) on the header of each ECRD data, will indicate if any capture trigger was lost between read operations (this can happen if the capture triggers are faster than the routine that is reading back the values).

Capture/Compare Unit 4 (CCU4)

MEMORY	1st Read Back			2nd Read Back			2nd Read Back		
	Timer 2 previous data	Previous		Timer 2 previous data	Previous		Timer 2 previous data	Previous	
	Timer 2 previous data	Previous		Timer 2 previous data	Previous		Timer 2 previous data	Previous	
	Timer 2 previous data	Previous		Timer 2 previous data	Previous		Timer 2 previous data	Previous	
	Timer 2 previous data	Previous		Timer 2 previous data	Previous		Timer 2 previous data	Previous	
	Timer 1 previous data	Previous		Timer 1 previous data	Previous		Timer 1 previous data	Previous	
	Timer 1 previous data	Previous		Timer 1 previous data	Previous		Timer 1 previous data	Previous	
	Timer 1 previous data	Previous		Timer 1 previous data	Previous		Timer 1 previous data	Previous	
	Timer 1 previous data	Previous		Timer 1 previous data	Previous		Timer 1 previous data	Previous	
	Timer 0 previous data	Previous		Timer 0 previous data	Previous		Timer 0 previous data	Previous	
	Timer 0 previous data	Previous		Timer 0 previous data	Empty		Timer 0	xxxx_H	Empty
	Timer 0 previous data	Previous		Timer 0	5888_H	Full	Timer 0	5888_H	Full
Timer 0	5790_H	Full	Timer 0	5790_H	Full	Timer 0	5790_H	Full	
...									
	10th Read Back			11th Read Back			12th Read Back		
Timer 2 previous data	Previous		Timer 2 previous data	Previous		Timer 2	xxxx_H	Empty	
Timer 2 previous data	Previous		Timer 2	xxxx_H	Empty	Timer 2	xxxx_H	Empty	
Timer 2	xxxx_H	Empty	Timer 2	xxxx_H	Empty	Timer 2	xxxx_H	Empty	
Timer 2	0009_H	Full	Timer 2	0009_H	Full	Timer 2	0009_H	Full	
Timer 1	0FCC_H	Full	Timer 1	0FCC_H	Full	Timer 1	0FCC_H	Full	
Timer 1	0FC0_H	Full	Timer 1	0FC0_H	Full	Timer 1	0FC0_H	Full	
Timer 1	0F09_H	Full	Timer 1	0F09_H	Full	Timer 1	0F09_H	Full	
Timer 1	0EFF_H	Full	Timer 1	0EFF_H	Full	Timer 1	0EFF_H	Full	
Timer 0	xxxx_H	Empty	Timer 0	xxxx_H	Empty	Timer 0	xxxx_H	Empty	
Timer 0	xxxx_H	Empty	Timer 0	xxxx_H	Empty	Timer 0	xxxx_H	Empty	
Timer 0	5888_H	Full	Timer 0	5888_H	Full	Timer 0	5888_H	Full	
Timer 0	5790_H	Full	Timer 0	5790_H	Full	Timer 0	5790_H	Full	

Figure 21-72 Memory structure for extended read back

21.3 Service Request Generation

Each CCU4 slice has an interrupt structure as the one in [Figure 21-73](#). The register **CC4yINTS** is the status register for the interrupt sources. Each dedicated interrupt

Capture/Compare Unit 4 (CCU4)

source can be set or cleared by SW, by writing into the specific bit in the **CC4ySWS** and **CC4ySWR** registers respectively.

Each interrupt source can be enabled/disabled via the **CC4yINTE** register. An enabled interrupt source will always generate a pulse on the service request line even if the specific status bit was not cleared. **Table 21-9** describes the interrupt sources of each CCU4 slice.

The interrupt sources, Period Match while counting up and one Match while counting down are ORed together. The same mechanism is applied to the Compare Match while counting up and Compare Match while counting down.

The interrupt sources for the external events are directly linked with the configuration set on the **CC4yINS.EVxEM**. If an event is programmed to be active on both edges, that means that service request pulse is going to be generated when any transition on the external signal is detected. If the event is linked with a level function, the **CC4yINS.EVxEM** still can be programmed to enable a service request pulse. The TRAP event doesn't need any of extra configuration for generating the service request pulse when the slice enters the TRAP state.

Table 21-9 Interrupt sources

Signal	Description
CCINEV0_E	Event 0 edge(s) information from event selector. Used when an external signal should trigger an interrupt.
CCINEV1_E	Event 1 edge(s) information from event selector. Used when an external signal should trigger an interrupt.
CCINEV2_E	Event 2 edge(s) information from event selector. Used when an external signal should trigger an interrupt.
CCPM_U	Period Match while counting up
CCCM_U	Compare Match while counting up
CCCM_D	Compare Match while counting down
CCOM_D	One Match while counting down
Trap state set	Entering Trap State. Will set the E2AS

Capture/Compare Unit 4 (CCU4)

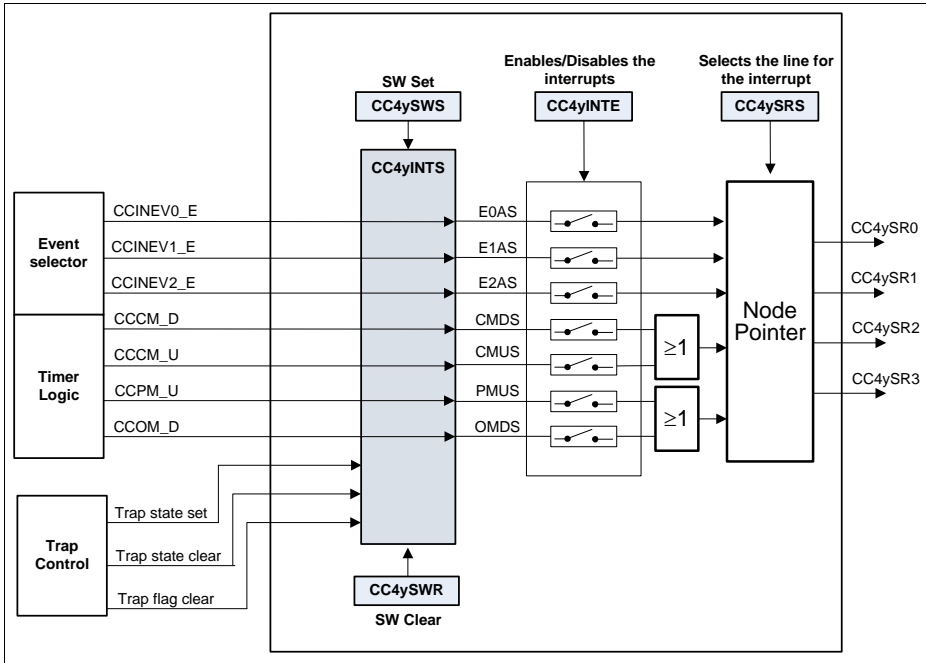


Figure 21-73 Slice interrupt structure overview

Each of the interrupt events can then be forwarded to one of the slice's four service request lines, [Figure 21-74](#). The value set on the **CC4ySRS** controls which interrupt event is mapped into which service request line.

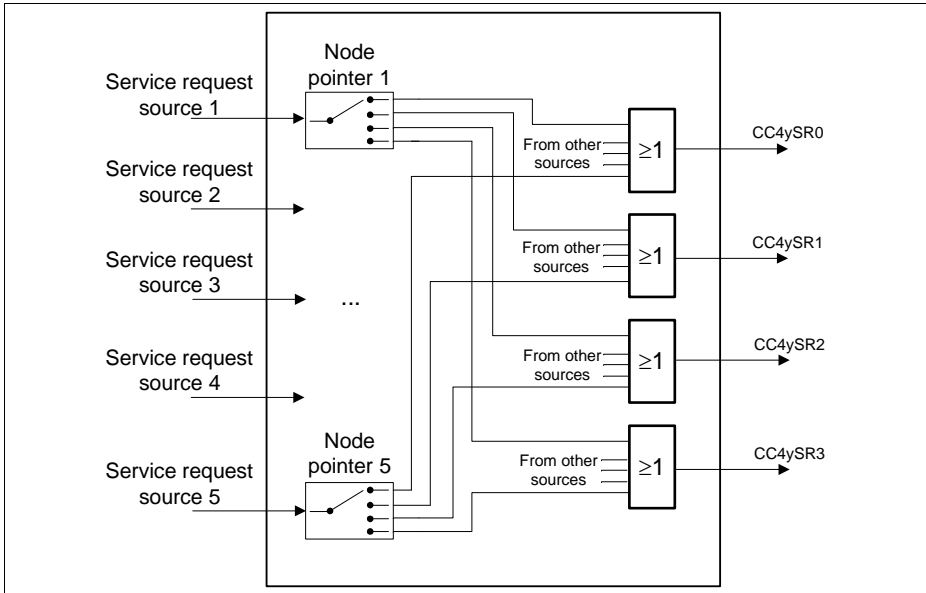


Figure 21-74 Slice Interrupt Node Pointer overview

The four service request lines of each slice are OR together inside the kernel of the CCU4, see [Figure 21-75](#). This means that there are only four service request lines per CCU4, that can have in each line interrupt requests coming from different slices.

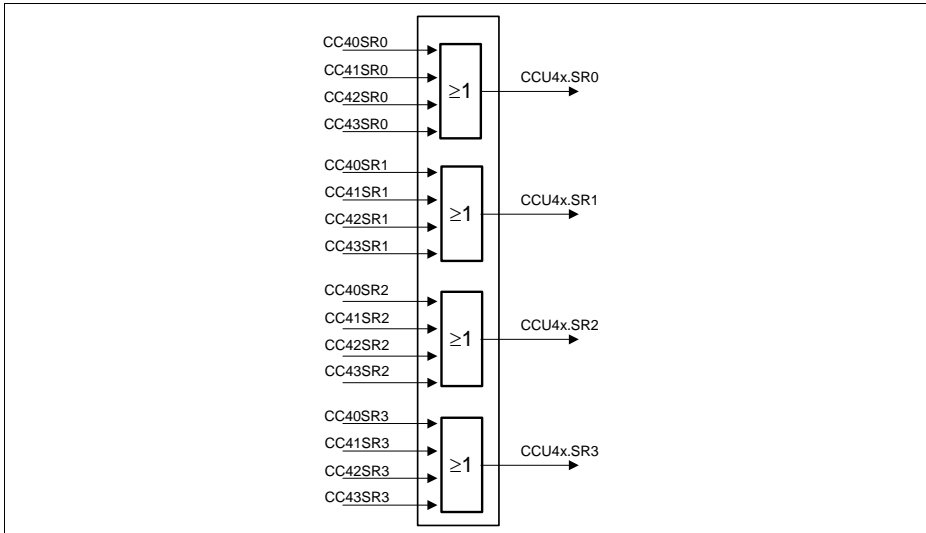


Figure 21-75 CCU4 service request overview

21.4 Debug Behavior

In suspend mode, the functional clocks for all slices as well the prescaler are stopped. The registers can still be accessed by the CPU (read only). This mode is useful for debugging purposes, e.g. where the current device status should be frozen in order to get a snapshot of the internal values. In suspend mode, all the slice timers are stopped. The suspend mode is non-intrusive concerning the register bits. This means register bits are not modified by hardware when entering or leaving the suspend mode.

Entry into suspend mode can be configured at the kernel level by means of the field **GCTRL.SUSCFG**.

The module is only functional after the suspend signal becomes inactive.

21.5 Power, Reset and Clock

The following sections describe the operating conditions, characteristics and timing requirements for the CCU4. All the timing information is related to the module clock, f_{CCU4} .

21.5.1 Clocks

Module Clock

The module clock of the CCU4 module is described in the SCU chapter as f_{CCU} .

The bus interface clock of the CCU4 module is described in the SCU chapter as f_{PERIPH} .

Capture/Compare Unit 4 (CCU4)

The module clock for the CCU4 is controlled via a specific control bit inside the SCU (System Control Unit), register CLKSET.

It is possible to disable the module clock for the CCU4 via the **GSTAT** register, nevertheless, there may be a dependency of the f_{ccu4} through the different CCU4 instances. One should address the SCU Chapter for a complete description of the product clock scheme.

If module clock dependencies exist through different IP instances, then one can disable the module clock internally inside the specific CCU4, by disabling the prescaler (**GSTAT.PRB** = 0_B).

External Clock

It is possible to use an external clock as source for the prescaler, and consequently for all the timer Slices, CC4y. This external source can be connected to one of the CCU4x.CLK[C...A] inputs.

This external source is nevertheless synchronized against f_{ccu4} .

Table 21-10 External clock operating conditions

Parameter	Symbol	Values			Unit	Note / Test Condition
		Min.	Typ.	Max.		
Frequency	f_{eclk}	–	–	$f_{ccu4}/4$	MHz	
ON time	ton_{eclk}	$2T_{ccu4}^{1)2)}$	–	–	ns	
OFF time	$toff_{eclk}$	$2T_{ccu4}^{1)2)}$	–	–	ns	Only the rising edge is used

1) Only valid if the signal was not previously synchronized/generated with the fccu4 clock (or a synchronous clock)

2) 50% duty cycle is not obligatory

21.5.2 Module Reset

Each CCU4 has one reset source. This reset source is handled at system level and it can be generated independently via a system control register, PRSET0/PRSET1 (address SCU chapter for a full description).

After reset release, the complete IP is set to default configuration. The default configuration for each register field is addressed on **Section 21.7**.

21.5.3 Power

The CCU4 is inside the power core domain, therefore no special considerations about power up or power down sequences need to be taken. For an explanation about the different power domains, please address the SCU (System Control Unit) chapter.

An internal power down mode for the CCU4, can be achieved by disabling the clock inside the CCU4 itself. For this one should set the **GSTAT** register with the default reset value (via the idle mode set register, **GIDLS**).

21.6 Initialization and System Dependencies

21.6.1 Initialization Sequence

The initialization sequence for an application that is using the CCU4, should be the following:

- 1st Step:** Apply reset to the CCU4, via the specific SCU bitfield on the PRSET0/PRSET1 register.
- 2nd Step:** Release reset of the CCU4, via the specific SCU bitfield on the PRCLR0/PRCLR1 register
- 3rd Step:** Enable the CCU4 clock via the specific SCU register, CLKSET.
- 4th Step:** Enable the prescaler block, by writing 1_B to the **GIDLC**.SPRB field.
- 5th Step:** Configure the global CCU4 register **GCTRL**
- 6th Step:** Configure all the registers related to the required Timer Slice(s) functions, including the interrupt/service request configuration.
- 7th Step:** If needed, configure the startup value for a specific Compare Channel Status, of a Timer Slice, by writing 1_B to the specific **GCSS**.SyTS.
- 8th Step:** Enable the specific timer slice(s), CC4y, by writing 1_B to the specific **GIDLC**.CSyl.
- 9th Step:** For all the Timer Slices that should be started synchronously via SW, the specific system register localized in the SCU, CCUCON, that enables a synchronous timer start should be addressed. The SCU.GSC4x input signal needs to be configured previously as a start function, see **Section 21.2.7.1**.

21.6.2 System Dependencies

Each CCU4 may have different dependencies regarding module and bus clock frequencies. This dependencies should be addressed in the SCU and System Architecture Chapters.

Capture/Compare Unit 4 (CCU4)

Dependencies between several peripherals, regarding different clock operating frequencies may also exist. This should be addressed before configuring the connectivity between the CCU4 and some other peripheral.

The following topics must be taken into consideration for good CCU4 and system operation:

- CCU4 module clock must be at maximum two times faster than the module bus interface clock
- Module input triggers for the CCU4 must not exceed the module clock frequency (if the triggers are generated internally in the device)
- Module input triggers for the CCU4 must not exceed the frequency dictated in [Section 21.5.1](#)
- Frequency of the CCU4 outputs used as triggers/functions on other modules, must be crosschecked on the end point
- Applying and removing CCU4 from reset, can cause unwanted operations in other modules. This can occur if the modules are using CCU4 outputs as triggers/functions.

21.7 Registers

Registers Overview

The absolute register address is calculated by adding:
Module Base Address + Offset Address

Table 21-11 Registers Address Space

Module	Base Address	End Address	Note
CCU40	4000C000 _H	4000FFFF _H	
CCU41	40010000 _H	40013FFF _H	

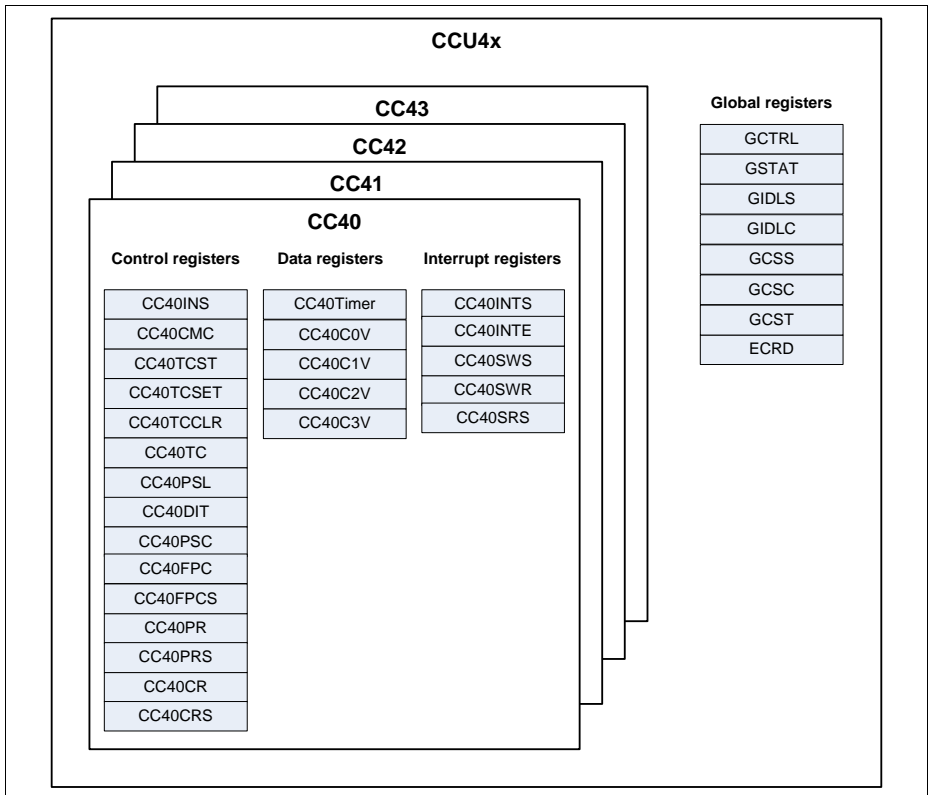


Figure 21-76 CCU4 registers overview

Table 21-12 Register Overview of CCU4

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	

CCU4 Global Registers

GCTRL	Module General Control Register	0000 _H	U, PV	U, PV	Page 21-87
GSTAT	General Slice Status Register	0004 _H	U, PV	BE	Page 21-90
GIDLS	General Idle Enable Register	0008 _H	U, PV	U, PV	Page 21-91
GIDLC	General Idle Disable Register	000C _H	U, PV	U, PV	Page 21-93
GCSS	General Channel Set Register	0010 _H	U, PV	U, PV	Page 21-94
GCSC	General Channel Clear Register	0014 _H	U, PV	U, PV	Page 21-96
GCST	General Channel Status Register	0018 _H	U, PV	BE	Page 21-99
MIDR	Module Identification Register	0080 _H	U, PV	BE	Page 21-102

CC40 Registers

CC40INS	Input Selector Unit Configuration	0100 _H	U, PV	U, PV	Page 21-102
CC40CMC	Connection Matrix Configuration	0104 _H	U, PV	U, PV	Page 21-104
CC40TST	Timer Run Status	0108 _H	U, PV	BE	Page 21-107
CC40TCSET	Timer Run Set	010C _H	U, PV	U,PV	Page 21-108
CC40TCCLR	Timer Run Clear	0110 _H	U, PV	U, PV	Page 21-109
CC40TC	General Timer Configuration	0114 _H	U, PV	U, PV	Page 21-110
CC40PSL	Output Passive Level Configuration	0118 _H	U, PV	U, PV	Page 21-115
CC40DIT	Dither Configuration	011C _H	U, PV	BE	Page 21-115
CC40DITS	Dither Shadow Register	0120 _H	U, PV	U, PV	Page 21-116
CC40PSC	Prescaler Configuration	0124 _H	U, PV	U, PV	Page 21-117
CC40FPC	Prescaler Compare Value	0128 _H	U, PV	U, PV	Page 21-118

Capture/Compare Unit 4 (CCU4)
Table 21-12 Register Overview of CCU4 (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
CC40FPCS	Prescaler Shadow Compare Value	012C _H	U, PV	U, PV	Page 21-119
CC40PR	Timer Period Value	0130 _H	U, PV	BE	Page 21-119
CC40PRS	Timer Period Shadow Value	0134 _H	U, PV	U, PV	Page 21-120
CC40CR	Timer Compare Value	0138 _H	U, PV	BE	Page 21-121
CC40CRS	Timer Compare Shadow Value	013C _H	U, PV	U, PV	Page 21-122
CC40TIMER	Timer Current Value	0170 _H	U, PV	U, PV	Page 21-123
CC40C0V	Capture Register 0 Value	0174 _H	U, PV	BE	Page 21-123
CC40C1V	Capture Register 1 Value	0178 _H	U, PV	BE	Page 21-124
CC40C2V	Capture Register 2 Value	017C _H	U, PV	BE	Page 21-125
CC40C3V	Capture Register 3 Value	0180 _H	U, PV	BE	Page 21-126
CC40INTS	Interrupt Status	01A0 _H	U, PV	BE	Page 21-127
CC40INTE	Interrupt Enable	01A4 _H	U, PV	U, PV	Page 21-129
CC40SRS	Interrupt Configuration	01A8 _H	U, PV	U, PV	Page 21-131
CC40SWS	Interrupt Status Set	01AC _H	U, PV	U, PV	Page 21-132
CC40SWR	Interrupt Status Clear	01B0 _H	U, PV	U, PV	Page 21-134
CC40ECD0	Extended Read Back 0	01B8 _H	U, PV	BE	Page 21-135
CC40ECD1	Extended Read Back 1	01BC _H	U, PV	BE	Page 21-137

CC41 Registers

CC41INS	Input Selector Unit Configuration	0200 _H	U, PV	U, PV	Page 21-102
CC41CMC	Connection Matrix Configuration	0204 _H	U, PV	U, PV	Page 21-104
CC41TST	Timer Run Status	0208 _H	U, PV	BE	Page 21-107
CC41TCSET	Timer Run Set	020C _H	U, PV	U, PV	Page 21-108
CC41TCCLR	Timer Run Clear	0210 _H	U, PV	U, PV	Page 21-109
CC41TC	General Timer Configuration	0214 _H	U, PV	U, PV	Page 21-110
CC41PSL	Output Passive Level Configuration	0218 _H	U, PV	U, PV	Page 21-115

Capture/Compare Unit 4 (CCU4)
Table 21-12 Register Overview of CCU4 (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
CC41DIT	Dither Configuration	021C _H	U, PV	BE	Page 21-115
CC41DITS	Dither Shadow Register	0220 _H	U, PV	U, PV	Page 21-116
CC41PSC	Prescaler Configuration	0224 _H	U, PV	U, PV	Page 21-117
CC41FPC	Prescaler Compare Value	0228 _H	U, PV	U, PV	Page 21-118
CC41FPCS	Prescaler Shadow Compare Value	022C _H	U, PV	U, PV	Page 21-119
CC41PR	Timer Period Value	0230 _H	U, PV	BE	Page 21-119
CC41PRS	Timer Period Shadow Value	0234 _H	U, PV	U, PV	Page 21-120
CC41CR	Timer Compare Value	0238 _H	U, PV	BE	Page 21-121
CC41CRS	Timer Compare Shadow Value	023C _H	U, PV	U, PV	Page 21-122
CC41TIMER	Timer Current Value	0270 _H	U, PV	U, PV	Page 21-123
CC41C0V	Capture Register 0 Value	0274 _H	U, PV	BE	Page 21-123
CC41C1V	Capture Register 1 Value	0278 _H	U, PV	BE	Page 21-124
CC41C2V	Capture Register 2 Value	027C _H	U, PV	BE	Page 21-125
CC41C3V	Capture Register 3 Value	0280 _H	U, PV	BE	Page 21-126
CC41INTS	Interrupt Status	02A0 _H	U, PV	BE	Page 21-127
CC41INTE	Interrupt Enable	02A4 _H	U, PV	U, PV	Page 21-129
CC41SRS	Interrupt Configuration	02A8 _H	U, PV	U, PV	Page 21-131
CC41SWS	Interrupt Status Set	02AC _H	U, PV	U, PV	Page 21-132
CC41SWR	Interrupt Status Clear	02B0 _H	U, PV	U, PV	Page 21-134
CC41ECD0	Extended Read Back 0	02B8 _H	U, PV	BE	Page 21-135
CC41ECD1	Extended Read Back 1	02BC _H	U, PV	BE	Page 21-137

CC42 Registers

CC42INS	Input Selector Unit Configuration	0300 _H	U, PV	U, PV	Page 21-102
CC42CMC	Connection Matrix Configuration	0304 _H	U, PV	U, PV	Page 21-104
CC42TST	Timer Run Status	0308 _H	U, PV	BE	Page 21-107
CC42TCSET	Timer Run Set	030C _H	U, PV	U, PV	Page 21-108

Capture/Compare Unit 4 (CCU4)

Table 21-12 Register Overview of CCU4 (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
CC42TCCLR	Timer Run Clear	0310 _H	U, PV	U, PV	Page 21-109
CC42TC	General Timer Configuration	0314 _H	U, PV	U, PV	Page 21-110
CC42PSL	Output Passive Level Configuration	0318 _H	U, PV	U, PV	Page 21-115
CC42DIT	Dither Configuration	031C _H	U, PV	BE	Page 21-115
CC42DITS	Dither Shadow Register	0320 _H	U, PV	U, PV	Page 21-116
CC42PSC	Prescaler Configuration	0324 _H	U, PV	U, PV	Page 21-117
CC42FPC	Prescaler Compare Value	0328 _H	U, PV	U, PV	Page 21-118
CC42FPCS	Prescaler Shadow Compare Value	032C _H	U, PV	U, PV	Page 21-119
CC42PR	Timer Period Value	0330 _H	U, PV	BE	Page 21-119
CC42PRS	Timer Period Shadow Value	0334 _H	U, PV	U, PV	Page 21-120
CC42CR	Timer Compare Value	0338 _H	U, PV	BE	Page 21-121
CC42CRS	Timer Compare Shadow Value	033C _H	U, PV	U, PV	Page 21-122
CC42TIMER	Timer Current Value	0370 _H	U, PV	U, PV	Page 21-123
CC42C0V	Capture Register 0 Value	0374 _H	U, PV	BE	Page 21-123
CC42C1V	Capture Register 1 Value	0378 _H	U, PV	BE	Page 21-124
CC42C2V	Capture Register 2 Value	037C _H	U, PV	BE	Page 21-125
CC42C3V	Capture Register 3 Value	0380 _H	U, PV	BE	Page 21-126
CC42INTS	Interrupt Status	03A0 _H	U, PV	BE	Page 21-127
CC42INTE	Interrupt Enable	03A4 _H	U, PV	U, PV	Page 21-129
CC42SRS	Interrupt Configuration	03A8 _H	U, PV	U, PV	Page 21-131
CC42SWS	Interrupt Status Set	03AC _H	U, PV	U, PV	Page 21-132
CC42SWR	Interrupt Status Clear	03B0 _H	U, PV	U, PV	Page 21-134
CC42ECD0	Extended Read Back 0	03B8 _H	U, PV	BE	Page 21-135
CC42ECD1	Extended Read Back 1	03BC _H	U, PV	BE	Page 21-137

CC43 Registers

CC43INS	Input Selector Unit Configuration	0400 _H	U, PV	U, PV	Page 21-102
---------	-----------------------------------	-------------------	-------	-------	-----------------------------

Capture/Compare Unit 4 (CCU4)
Table 21-12 Register Overview of CCU4 (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
CC43CMC	Connection Matrix Configuration	0404 _H	U, PV	U, PV	Page 21-104
CC43TST	Timer Run Status	0408 _H	U, PV	BE	Page 21-107
CC43TCSET	Timer Run Set	040C _H	U, PV	U,PV	Page 21-108
CC43TCCLR	Timer Run Clear	0410 _H	U, PV	U, PV	Page 21-109
CC43TC	General Timer Configuration	0414 _H	U, PV	U, PV	Page 21-110
CC43PSL	Output Passive Level Configuration	0418 _H	U, PV	U, PV	Page 21-115
CC43DIT	Dither Configuration	041C _H	U, PV	BE	Page 21-115
CC43DITS	Dither Shadow Register	0420 _H	U, PV	U, PV	Page 21-116
CC43PSC	Prescaler Configuration	0424 _H	U, PV	U, PV	Page 21-117
CC43FPC	Prescaler Compare Value	0428 _H	U, PV	U, PV	Page 21-118
CC43FPCS	Prescaler Shadow Compare Value	042C _H	U, PV	U, PV	Page 21-119
CC43PR	Timer Period Value	0430 _H	U, PV	BE	Page 21-119
CC43PRS	Timer Period Shadow Value	0434 _H	U, PV	U, PV	Page 21-120
CC43CR	Timer Compare Value	0438 _H	U, PV	BE	Page 21-121
CC43CRS	Timer Compare Shadow Value	043C _H	U, PV	U, PV	Page 21-122
CC43TIMER	Timer Current Value	0470 _H	U, PV	U, PV	Page 21-123
CC43C0V	Capture Register 0 Value	0474 _H	U, PV	BE	Page 21-123
CC43C1V	Capture Register 1 Value	0478 _H	U, PV	BE	Page 21-124
CC43C2V	Capture Register 2 Value	047C _H	U, PV	BE	Page 21-125
CC43C3V	Capture Register 3 Value	0480 _H	U, PV	BE	Page 21-126
CC43INTS	Interrupt Status	04A0 _H	U, PV	BE	Page 21-127
CC43INTE	Interrupt Enable	04A4 _H	U, PV	U, PV	Page 21-129
CC43SRS	Interrupt Configuration	04A8 _H	U, PV	U, PV	Page 21-131
CC43SWS	Interrupt Status Set	04AC _H	U, PV	U, PV	Page 21-132
CC43SWR	Interrupt Status Clear	04B0 _H	U, PV	U, PV	Page 21-134

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
PRBC	[2:0]	rw	<p>Prescaler Clear Configuration</p> <p>This register controls how the prescaler Run Bit and internal registers are cleared.</p> <p>000_B SW only</p> <p>001_B GSTAT.PRB and prescaler registers are cleared when the Run Bit of CC40 is cleared.</p> <p>010_B GSTAT.PRB and prescaler registers are cleared when the Run Bit of CC41 is cleared.</p> <p>011_B GSTAT.PRB and prescaler registers are cleared when the Run Bit of CC42 is cleared.</p> <p>100_B GSTAT.PRB and prescaler registers are cleared when the Run Bit of CC43 is cleared.</p>
PCIS	[5:4]	rw	<p>Prescaler Input Clock Selection</p> <p>00_B Module clock</p> <p>01_B CCU4x.ECLKA</p> <p>10_B CCU4x.ECLKB</p> <p>11_B CCU4x.ECLKC</p>
SUSCFG	[9:8]	rw	<p>Suspend Mode Configuration</p> <p>This field controls the entering in suspend mode for all the CAPCOM4 slices.</p> <p>00_B Suspend request ignored. The module never enters in suspend</p> <p>01_B Stops all the running slices immediately. Safe stop is not applied.</p> <p>10_B Stops the block immediately and clamps all the outputs to PASSIVE state. Safe stop is applied.</p> <p>11_B Waits for the roll over of each slice to stop and clamp the slices outputs. Safe stop is applied.</p>
MSE0	10	rw	<p>Slice 0 Multi Channel shadow transfer enable</p> <p>When this field is set, a shadow transfer of slice 0 can be requested not only by SW but also via the CCU4x.MCSS input.</p> <p>0_B Shadow transfer can only be requested by SW</p> <p>1_B Shadow transfer can be requested via SW and via the CCU4x.MCSS input.</p>

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
MSE1	11	rw	<p>Slice 1 Multi Channel shadow transfer enable When this field is set, a shadow transfer of slice 1 can be requested not only by SW but also via the CCU4x.MCSS input.</p> <p>0_B Shadow transfer can only be requested by SW 1_B Shadow transfer can be requested via SW and via the CCU4x.MCSS input.</p>
MSE2	12	rw	<p>Slice 2 Multi Channel shadow transfer enable When this field is set, a shadow transfer of slice 2 can be requested not only by SW but also via the CCU4x.MCSS input.</p> <p>0_B Shadow transfer can only be requested by SW 1_B Shadow transfer can be requested via SW and via the CCU4x.MCSS input.</p>
MSE3	13	rw	<p>Slice 3 Multi Channel shadow transfer enable When this field is set, a shadow transfer of slice 3 can be requested not only by SW but also via the CCU4x.MCSS input.</p> <p>0_B Shadow transfer can only be requested by SW 1_B Shadow transfer can be requested via SW and via the CCU4x.MCSS input.</p>
MSDE	[15:14]	rw	<p>Multi Channel shadow transfer request configuration This field configures the type of shadow transfer requested via the CCU4x.MCSS input. The field CC4yTC.MSEy needs to be set in order for this configuration to have any effect.</p> <p>00_B Only the shadow transfer for period and compare values is requested 01_B Shadow transfer for the compare, period and prescaler compare values is requested 10_B Reserved 11_B Shadow transfer for the compare, period, prescaler and dither compare values is requested</p>

Capture/Compare Unit 4 (CCU4)

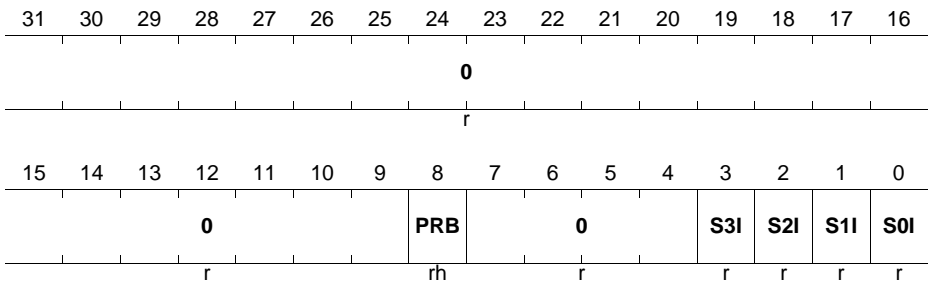
Field	Bits	Type	Description
0	3, [7:6], [31:16]	r	Reserved A read always returns 0.

GSTAT

The register contains the status of the prescaler and each timer slice (idle mode or running).

GSTAT

Global Status Register (0004_H) **Reset Value: 000000F_H**



Field	Bits	Type	Description
S0I	0	r	CC40 IDLE status This bit indicates if the CC40 slice is in IDLE mode or not. In IDLE mode the clocks for the CC40 slice are stopped. 0 _B Running 1 _B Idle
S1I	1	r	CC41 IDLE status This bit indicates if the CC41 slice is in IDLE mode or not. In IDLE mode the clocks for the CC41 slice are stopped. 0 _B Running 1 _B Idle

Capture/Compare Unit 4 (CCU4)

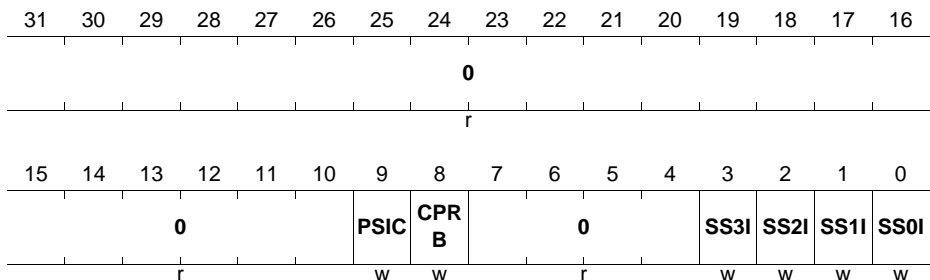
Field	Bits	Type	Description
S2I	2	r	CC42 IDLE status This bit indicates if the CC42 slice is in IDLE mode or not. In IDLE mode the clocks for the CC42 slice are stopped. 0 _B Running 1 _B Idle
S3I	3	r	CC43 IDLE status This bit indicates if the CC43 slice is in IDLE mode or not. In IDLE mode the clocks for the CC43 slice are stopped. 0 _B Running 1 _B Idle
PRB	8	rh	Prescaler Run Bit 0 _B Prescaler is stopped 1 _B Prescaler is running
0	[7:4], [31:9]	r	Reserved Read always returns 0.

GIDLS

Through this register one can set the prescaler and the specific timer slices into idle mode.

GIDLS

Global Idle Set (0008_H) Reset Value: 00000000_H



Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
SS0I	0	w	CC40 IDLE mode set Writing a 1 _B to this bit sets the CC40 slice in IDLE mode. The clocks for the slice are stopped when in IDLE mode. When entering IDLE, the internal slice registers are not cleared. A read access always returns 0.
SS1I	1	w	CC41 IDLE mode set Writing a 1 _B to this bit sets the CC41 slice in IDLE mode. The clocks for the slice are stopped when in IDLE mode. When entering IDLE, the internal slice registers are not cleared. A read access always returns 0.
SS2I	2	w	CC42 IDLE mode set Writing a 1 _B to this bit sets the CC42 slice in IDLE mode. The clocks for the slice are stopped when in IDLE mode. When entering IDLE, the internal slice registers are not cleared. A read access always returns 0.
SS3I	3	w	CC43 IDLE mode set Writing a 1 _B to this bit sets the CC43 slice in IDLE mode. The clocks for the slice are stopped when in IDLE mode. When entering IDLE, the internal slice registers are not cleared. A read access always returns 0.
CPRB	8	w	Prescaler Run Bit Clear Writing a 1 _B into this register clears the Run Bit of the prescaler. Prescaler internal registers are not cleared. A read always returns 0.
PSIC	9	w	Prescaler clear Writing a 1 _B to this register clears the prescaler counter. It also loads the PSIV into the PVAL field for all Timer Slices. This performs a re alignment of the timer clock for all Slices. The Run Bit of the prescaler is not cleared. A read always returns 0.
0	[7:4], [31:10]	r	Reserved Read always returns 0.

Capture/Compare Unit 4 (CCU4)

GIDLC

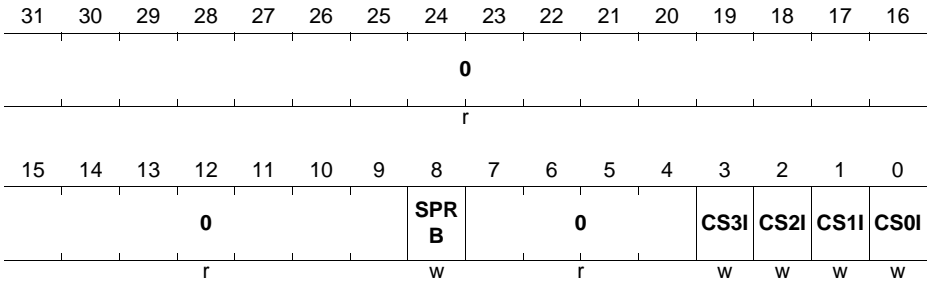
Through this register one can remove the prescaler and the specific timer slices from idle mode.

GIDLC

Global Idle Clear

(000C_H)

Reset Value: 00000000_H



Field	Bits	Type	Description
CS0I	0	w	CC40 IDLE mode clear Writing a 1 _B to this bit removes the CC40 from IDLE mode. A read access always returns 0.
CS1I	1	w	CC41 IDLE mode clear Writing a 1 _B to this bit removes the CC41 from IDLE mode. A read access always returns 0.
CS2I	2	w	CC42 IDLE mode clear Writing a 1 _B to this bit removes the CC42 from IDLE mode. A read access always returns 0.
CS3I	3	w	CC43 IDLE mode clear Writing a 1 _B to this bit removes the CC43 from IDLE mode. A read access always returns 0.
SPRB	8	w	Prescaler Run Bit Set Writing a 1 _B into this register sets the Run Bit of the prescaler. A read always returns 0.
0	[7:4], [31:9]	r	Reserved Read always returns 0.

GCSS

Through this register one can request a shadow transfer for the specific timer slice(s) and set the status bit for each of the compare channels.

Capture/Compare Unit 4 (CCU4)

GCSS

Global Channel Set

(0010_H)

Reset Value: 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												S3S TS	S2S TS	S1S TS	S0S TS
r												w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3P SE	S3D SE	S3S E	0	S2P SE	S2D SE	S2S E	0	S1P SE	S1D SE	S1S E	0	S0P SE	S0D SE	S0S E
r	w	w	w	r	w	w	w	r	w	w	w	r	w	w	w

Field	Bits	Type	Description
S0SE	0	w	Slice 0 shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S0SS field, enabling then a shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S0DSE	1	w	Slice 0 Dither shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S0DSS field, enabling then a shadow transfer for the Dither compare value. A read always returns 0.
S0PSE	2	w	Slice 0 Prescaler shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S0PSS field, enabling then a shadow transfer for the prescaler compare value. A read always returns 0.
S1SE	4	w	Slice 1 shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S1SS field, enabling then a shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S1DSE	5	w	Slice 1 Dither shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S1DSS field, enabling then a shadow transfer for the Dither compare value. A read always returns 0.

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
S1PSE	6	w	Slice 1 Prescaler shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S1PSS field, enabling then a shadow transfer for the prescaler compare value. A read always returns 0.
S2SE	8	w	Slice 2 shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S2SS field, enabling then a shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S2DSE	9	w	Slice 2 Dither shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S2DSS field, enabling then a shadow transfer for the Dither compare value. A read always returns 0.
S2PSE	10	w	Slice 2 Prescaler shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S2PSS field, enabling then a shadow transfer for the prescaler compare value. A read always returns 0.
S3SE	12	w	Slice 3 shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S3SS field, enabling then a shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S3DSE	13	w	Slice 3 Dither shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S3DSS field, enabling then a shadow transfer for the Dither compare value. A read always returns 0.
S3PSE	14	w	Slice 3 Prescaler shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S3PSS field, enabling then a shadow transfer for the prescaler compare value. A read always returns 0.
S0STS	16	w	Slice 0 status bit set Writing a 1 _B into this field sets the status bit of slice 0 (GCST.CC40ST) to 1 _B . A read always returns 0.

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
S1STS	17	w	Slice 1 status bit set Writing a 1 _B into this field sets the status bit of slice 1 (GCST.CC41ST) to 1 _B . A read always returns 0.
S2STS	18	w	Slice 2 status bit set Writing a 1 _B into this field sets the status bit of slice 2 (GCST.CC42ST) to 1 _B . A read always returns 0.
S3STS	19	w	Slice 3 status bit set Writing a 1 _B into this field sets the status bit of slice 3 (GCST.CC43ST) to 1 _B . A read always returns 0.
0	3, 7, 11, 15, [31:20]	r	Reserved Read always returns 0.

GCSC

Through this register one can reset a shadow transfer request for the specific timer slice and clear the status bit for each the compare channels.

GCSC

Global Channel Clear (0014_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												S3S TC	S2S TC	S1S TC	S0S TC
r												w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3P SC	S3D SC	S3S C	0	S2P SC	S2D SC	S2S C	0	S1P SC	S1D SC	S1S C	0	S0P SC	S0D SC	S0S C
r	w	w	w	r	w	w	w	r	w	w	w	r	w	w	w

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
S0SC	0	w	Slice 0 shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S0SS field, canceling any pending shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S0DSC	1	w	Slice 0 Dither shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S0DSS field, canceling any pending shadow transfer for the Dither compare value. A read always returns 0.
S0PSC	2	w	Slice 0 Prescaler shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S0PSS field, canceling any pending shadow transfer for the prescaler compare value. A read always returns 0.
S1SC	4	w	Slice 1 shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S1SS field, canceling any pending shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S1DSC	5	w	Slice 1 Dither shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S1DSS field, canceling any pending shadow transfer for the Dither compare value. A read always returns 0.
S1PSC	6	w	Slice 1 Prescaler shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S1PSS field, canceling any pending shadow transfer for the prescaler compare value. A read always returns 0.
S2SC	8	w	Slice 2 shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S2SS field, canceling any pending shadow transfer for the Period, Compare and Passive level values. A read always returns 0.

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
S2DSC	9	w	Slice 2 Dither shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S2DSS field, canceling any pending shadow transfer for the Dither compare value. A read always returns 0.
S2PSC	10	w	Slice 2 Prescaler shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S2PSS field, canceling any pending shadow transfer for the prescaler compare value. A read always returns 0.
S3SC	12	w	Slice 3 shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S3SS field, canceling any pending shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S3DSC	13	w	Slice 3 Dither shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S3DSS field, canceling any pending shadow transfer for the Dither compare value. A read always returns 0.
S3PSC	14	w	Slice 3 Prescaler shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S3PSS field, canceling any pending shadow transfer for the prescaler compare value. A read always returns 0.
S0STC	16	w	Slice 0 status bit clear Writing a 1 _B into this field clears the status bit of slice 0 (GCST.CC40ST) to 0 _B . A read always returns 0.
S1STC	17	w	Slice 1 status bit clear Writing a 1 _B into this field clears the status bit of slice 1 (GCST.CC41ST) to 0 _B . A read always returns 0.
S2STC	18	w	Slice 2 status bit clear Writing a 1 _B into this field clears the status bit of slice 2 (GCST.CC42ST) to 0 _B . A read always returns 0.

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
S0PSS	2	rh	Slice 0 Prescaler shadow transfer status 0_B Prescaler shadow transfer has not been requested 1_B Prescaler shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S1SS	4	rh	Slice 1 shadow transfer status 0_B Shadow transfer has not been requested 1_B Shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S1DSS	5	rh	Slice 1 Dither shadow transfer status 0_B Dither shadow transfer has not been requested 1_B Dither shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S1PSS	6	rh	Slice 1 Prescaler shadow transfer status 0_B Prescaler shadow transfer has not been requested 1_B Prescaler shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S2SS	8	rh	Slice 2 shadow transfer status 0_B Shadow transfer has not been requested 1_B Shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S2DSS	9	rh	Slice 2 Dither shadow transfer status 0_B Dither shadow transfer has not been requested 1_B Dither shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.

Capture/Compare Unit 4 (CCU4)

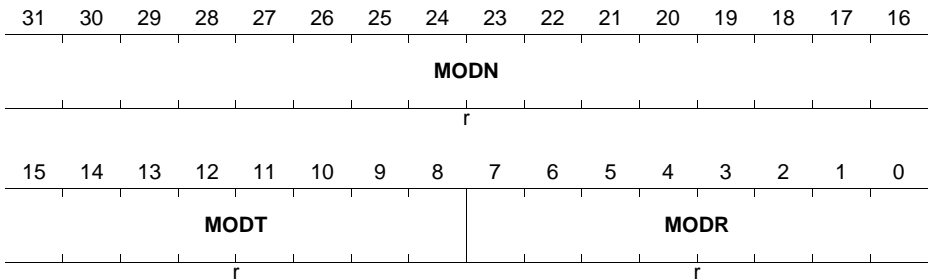
Field	Bits	Type	Description
S2PSS	10	rh	Slice 2 Prescaler shadow transfer status 0_B Prescaler shadow transfer has not been requested 1_B Prescaler shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S3SS	12	rh	Slice 3 shadow transfer status 0_B Shadow transfer has not been requested 1_B Shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S3DSS	13	rh	Slice 3 Dither shadow transfer status 0_B Dither shadow transfer has not been requested 1_B Dither shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S3PSS	14	rh	Slice 3 Prescaler shadow transfer status 0_B Prescaler shadow transfer has not been requested 1_B Prescaler shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
CC40ST	16	rh	Slice 0 status bit
CC41ST	17	rh	Slice 1 status bit
CC42ST	18	rh	Slice 2 status bit
CC43ST	19	rh	Slice 3 status bit
0	3, 7, 11, 15, [31:20]	r	Reserved Read always returns 0.

MIDR

This register contains the module identification number.

MIDR

Module Identification (0080_H) Reset Value: 00A6C0XX_H



Field	Bits	Type	Description
MODR	[7:0]	r	Module Revision This bit field indicates the revision number of the module implementation (depending on the design step). The given value of 00 _H is a placeholder for the actual number.
MODT	[15:8]	r	Module Type
MODN	[31:16]	r	Module Number

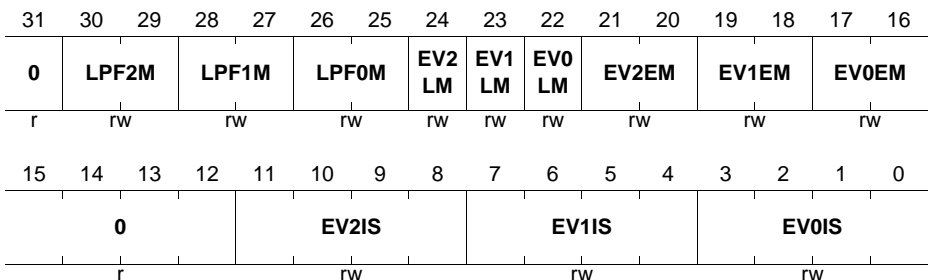
21.7.2 Slice (CC4y) Registers

CC4yINS

The register contains the configuration for the input selector.

CC4yINS (y = 0 - 3)

Input Selector Configuration (0100_H + 0100_H * y) Reset Value: 00000000_H



Field	Bits	Type	Description
EV0IS	[3:0]	rw	Event 0 signal selection This field selects which pins is used for the event 0. 0000 _B CCU4x.INyA 0001 _B CCU4x.INyB 0010 _B CCU4x.INyC 0011 _B CCU4x.INyD 0100 _B CCU4x.INyE 0101 _B CCU4x.INyF 0110 _B CCU4x.INyG 0111 _B CCU4x.INyH 1000 _B CCU4x.INyI 1001 _B CCU4x.INyJ 1010 _B CCU4x.INyK 1011 _B CCU4x.INyL 1100 _B CCU4x.INyM 1101 _B CCU4x.INyN 1110 _B CCU4x.INyO 1111 _B CCU4x.INyP
EV1IS	[7:4]	rw	Event 1 signal selection Same as EV0IS description
EV2IS	[11:8]	rw	Event 2 signal selection Same as EV0IS description
EV0EM	[17:16]	rw	Event 0 Edge Selection 00 _B No action 01 _B Signal active on rising edge 10 _B Signal active on falling edge 11 _B Signal active on both edges
EV1EM	[19:18]	rw	Event 1 Edge Selection Same as EV0EM description
EV2EM	[21:20]	rw	Event 2 Edge Selection Same as EV0EM description
EV0LM	22	rw	Event 0 Level Selection 0 _B Active on HIGH level 1 _B Active on LOW level
EV1LM	23	rw	Event 1 Level Selection Same as EV0LM description

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
EV2LM	24	rw	Event 2 Level Selection Same as EV0LM description
LPF0M	[26:25]	rw	Event 0 Low Pass Filter Configuration This field sets the number of consecutive counts for the Low Pass Filter of Event 0. The input signal value needs to remain stable for this number of counts (f_{CCU4}), so that a level/transition is accepted. 00 _B LPF is disabled 01 _B 3 clock cycles of f_{CCU4} 10 _B 5 clock cycles of f_{CCU4} 11 _B 7 clock cycles of f_{CCU4}
LPF1M	[28:27]	rw	Event 1 Low Pass Filter Configuration Same description as LPF0M
LPF2M	[30:29]	rw	Event 2 Low Pass Filter Configuration Same description as LPF0M
0	[15:12] , 31	r	Reserved Read always returns 0.

CC4yCMC

The register contains the configuration for the connection matrix.

CC4yCMC (y = 0 - 3)

Connection Matrix Control (0104_H + 0100_H * y) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											TCE	MOS	TS	OFS	
r											rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTS		LDS		UDS		GATES		CAP1S		CAP0S		ENDS		STRTS	
rw		rw		rw		rw		rw		rw		rw		rw	

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
STRTS	[1:0]	rw	<p>External Start Functionality Selector Selects the Event that is going to be linked with the external start functionality.</p> <p>00_B External Start Function deactivated 01_B External Start Function triggered by Event 0 10_B External Start Function triggered by Event 1 11_B External Start Function triggered by Event 2</p>
ENDS	[3:2]	rw	<p>External Stop Functionality Selector Selects the Event that is going to be linked with the external stop functionality.</p> <p>00_B External Stop Function deactivated 01_B External Stop Function triggered by Event 0 10_B External Stop Function triggered by Event 1 11_B External Stop Function triggered by Event 2</p>
CAP0S	[5:4]	rw	<p>External Capture 0 Functionality Selector Selects the Event that is going to be linked with the external capture for capture registers number 1 and 0.</p> <p>00_B External Capture 0 Function deactivated 01_B External Capture 0 Function triggered by Event 0 10_B External Capture 0 Function triggered by Event 1 11_B External Capture 0 Function triggered by Event 2</p>
CAP1S	[7:6]	rw	<p>External Capture 1 Functionality Selector Selects the Event that is going to be linked with the external capture for capture registers number 3 and 2.</p> <p>00_B External Capture 1 Function deactivated 01_B External Capture 1 Function triggered by Event 0 10_B External Capture 1 Function triggered by Event 1 11_B External Capture 1 Function triggered by Event 2</p>

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
GATES	[9:8]	rw	<p>External Gate Functionality Selector Selects the Event that is going to be linked with the counter gating function. This function is used to gate the timer increment/decrement procedure.</p> <p>00_B External Gating Function deactivated 01_B External Gating Function triggered by Event 0 10_B External Gating Function triggered by Event 1 11_B External Gating Function triggered by Event 2</p>
UDS	[11:10]	rw	<p>External Up/Down Functionality Selector Selects the Event that is going to be linked with the Up/Down counting direction control.</p> <p>00_B External Up/Down Function deactivated 01_B External Up/Down Function triggered by Event 0 10_B External Up/Down Function triggered by Event 1 11_B External Up/Down Function triggered by Event 2</p>
LDS	[13:12]	rw	<p>External Timer Load Functionality Selector Selects the Event that is going to be linked with the timer load function.</p> <p>00_B - External Load Function deactivated 01_B - External Load Function triggered by Event 0 10_B - External Load Function triggered by Event 1 11_B - External Load Function triggered by Event 2</p>
CNTS	[15:14]	rw	<p>External Count Selector Selects the Event that is going to be linked with the count function. The counter is going to be increment/decremented each time that a specific transition on the event is detected.</p> <p>00_B External Count Function deactivated 01_B External Count Function triggered by Event 0 10_B External Count Function triggered by Event 1 11_B External Count Function triggered by Event 2</p>
OFS	16	rw	<p>Override Function Selector This field enables the ST bit override functionality.</p> <p>0_B Override functionality disabled 1_B Status bit trigger override connected to Event 1; Status bit value override connected to Event 2</p>

Capture/Compare Unit 4 (CCU4)

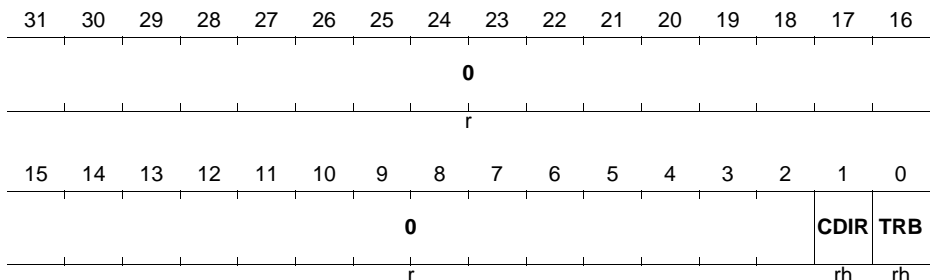
Field	Bits	Type	Description
TS	17	rw	Trap Function Selector This field enables the trap functionality. 0 _B Trap function disabled 1 _B TRAP function connected to Event 2
MOS	[19:18]	rw	External Modulation Functionality Selector Selects the Event that is going to be linked with the external modulation function. 00 _B - Modulation Function deactivated 01 _B - Modulation Function triggered by Event 0 10 _B - Modulation Function triggered by Event 1 11 _B - Modulation Function triggered by Event 2
TCE	20	rw	Timer Concatenation Enable This bit enables the timer concatenation with the previous slice. 0 _B Timer concatenation is disabled 1 _B Timer concatenation is enabled <i>Note: In CC40 this field doesn't exist. This is a read only reserved field. Read access always returns 0.</i>
0	[31:21]	r	Reserved A read always returns 0

CC4yTCST

The register holds the status of the timer (running/stopped) and the information about the counting direction (up/down).

CC4yTCST (y = 0 - 3)

Slice Timer Status (0108_H + 0100_H * y) **Reset Value: 00000000_H**



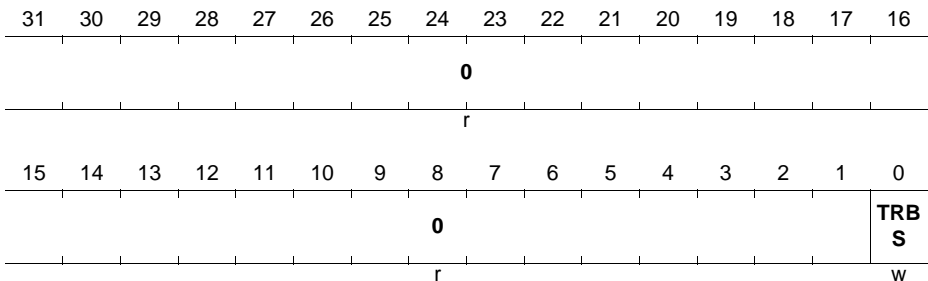
Field	Bits	Type	Description
TRB	0	rh	Timer Run Bit This field indicates if the timer is running. 0 _B Timer is stopped 1 _B Timer is running
CDIR	1	rh	Timer Counting Direction This field indicates if the timer is being increment or decremented 0 _B Timer is counting up 1 _B Timer is counting down
0	[31:2]	r	Reserved Read always returns 0

CC4yTCSET

Through this register it is possible to start the timer.

CC4yTCSET (y = 0 - 3)

Slice Timer Run Set (010C_H + 0100_H * y) **Reset Value: 00000000_H**



Field	Bits	Type	Description
TRBS	0	w	Timer Run Bit set Writing a 1 _B into this field sets the run bit of the timer. Read always returns 0.
0	[31:1]	r	Reserved Read always returns 0

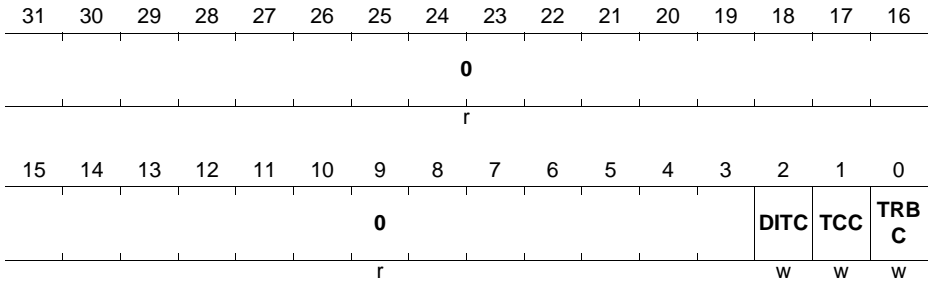
Capture/Compare Unit 4 (CCU4)

CC4yTCCLR

Through this register it is possible to stop and clear the timer, and clearing also the dither counter

CC4yTCCLR (y = 0 - 3)

Slice Timer Clear (0110_H + 0100_H * y) **Reset Value: 00000000_H**



Field	Bits	Type	Description
TRBC	0	w	Timer Run Bit Clear Writing a 1 _B into this field clears the run bit of the timer. The timer is not cleared. Read always returns 0.
TCC	1	w	Timer Clear Writing a 1 _B into this field clears the timer to 0000 _H . Read always returns 0.
DITC	2	w	Dither Counter Clear Writing a 1 _B into this field clears the dither counter to 0 _H . Read always returns 0.
0	[31:3]	r	Reserved Read always returns 0

CC4yTC

This register holds the several possible configurations for the timer operation.

Capture/Compare Unit 4 (CCU4)

CC4yTC (y = 0 - 3)

Slice Timer Control

(0114_H + 0100_H * y)

Reset Value: 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						MCM E	EMT	EMS	TRP SW	TRP SE	0			TRA PE	FPE
r						rw	rw	rw	rw	rw	r			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIM	DITHE	CCS	SCE	STR M	ENDM	0	CAPC		ECM	CMO D	CLS T	TSS M	TCM		
rw	rw	rw	rw	rw	rw	r	rw		rw	rh	rw	rw	rw		

Field	Bits	Type	Description
TCM	0	rw	<p>Timer Counting Mode</p> <p>This field controls the actual counting scheme of the timer.</p> <p>0_B Edge aligned mode 1_B Center aligned mode</p> <p><i>Note: When using an external signal to control the counting direction, the counting scheme is always edge aligned.</i></p>
TSSM	1	rw	<p>Timer Single Shot Mode</p> <p>This field controls the single shot mode. This is applicable in edge and center aligned modes.</p> <p>0_B Single shot mode is disabled 1_B Single shot mode is enabled</p>
CLST	2	rw	<p>Shadow Transfer on Clear</p> <p>Setting this bit to 1_B enables a shadow transfer when a timer clearing action is performed.</p> <p>Notice that the shadow transfer enable bitfields on the GCST register still need to be set to 1_B via software.</p>

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
CMOD	3	rh	<p>Capture Compare Mode</p> <p>This field indicates in which mode the slice is operating. The default value is compare mode. The capture mode is automatically set by the HW when an external signal is mapped to a capture trigger.</p> <p>0_B Compare Mode 1_B Capture Mode</p>
ECM	4	rw	<p>Extended Capture Mode</p> <p>This field control the Capture mode of the specific slice. It only has effect if the CMOD bit is 1_B.</p> <p>0_B Normal Capture Mode. Clear of the Full Flag of each capture register is done by accessing the registers individually only.</p> <p>1_B Extended Capture Mode. Clear of the Full Flag of each capture register is done not only by accessing the individual registers but also by accessing the ECRD register. When reading the ECRD register, only the capture register full flag pointed by the ECRD.VPTR is cleared.</p>
CAPC	[6:5]	rw	<p>Clear on Capture Control</p> <p>00_B Timer is never cleared on a capture event 01_B Timer is cleared on a capture event into capture registers 2 and 3. (When SCE = 1_B, Timer is always cleared in a capture event) 10_B Timer is cleared on a capture event into capture registers 0 and 1. (When SCE = 1_B, Timer is always cleared in a capture event) 11_B Timer is always cleared in a capture event.</p>

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
ENDM	[9:8]	rw	<p>Extended Stop Function Control</p> <p>This field controls the extended functions of the external Stop signal.</p> <p>00_B Clears the timer run bit only (default stop)</p> <p>01_B Clears the timer only (flush)</p> <p>10_B Clears the timer and run bit (flush/stop)</p> <p>11_B Reserved</p> <p><i>Note: When using an external up/down signal the flush operation sets the timer with zero if the counter is counting up and with the Period value if the counter is being decremented.</i></p>
STRM	10	rw	<p>Extended Start Function Control</p> <p>This field controls the extended functions of the external Start signal.</p> <p>0_B Sets run bit only (default start)</p> <p>1_B Clears the timer and sets run bit (flush/start)</p> <p><i>Note: When using an external up/down signal the flush operation sets the timer with zero if the counter is being incremented and with the Period value if the counter is being decremented.</i></p>
SCE	11	rw	<p>Equal Capture Event enable</p> <p>0_B Capture into CC4yC0V/CC4yC1V registers control by CCycapt0 and capture into CC4yC3V/CC4yC2V control by CCycapt1</p> <p>1_B Capture into CC4yC0V/CC4yC1V and CC4yC3V/CC4yC2V control by CCycapt1</p>
CCS	12	rw	<p>Continuous Capture Enable</p> <p>0_B The capture into a specific capture register is done with the rules linked with the full flags, described at Section 21.2.7.6.</p> <p>1_B The capture into the capture registers is always done regardless of the full flag status (even if the register has not been read back).</p>

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
DITHE	[14:13]	rw	<p>Dither Enable This field controls the dither mode for the slice. See Section 21.2.10.</p> <p>00_B Dither is disabled 01_B Dither is applied to the Period 10_B Dither is applied to the Compare 11_B Dither is applied to the Period and Compare</p>
DIM	15	rw	<p>Dither input selector This fields selects if the dither control signal is connected to the dither logic of the specific slice of is connected to the dither logic of slice 0. Notice that even if this field is set to 1_B, the field DITHE still needs to be programmed.</p> <p>0_B Slice is using its own dither unit 1_B Slice is connected to the dither unit of slice 0.</p>
FPE	16	rw	<p>Floating Prescaler enable Setting this bit to 1_B enables the floating prescaler mode.</p> <p>0_B Floating prescaler mode is disabled 1_B Floating prescaler mode is enabled</p>
TRAPE	17	rw	<p>TRAP enable Setting this bit to 1_B enables the TRAP action at the output pin. After mapping an external signal to the TRAP functionality, the user must set this field to 1_B to activate the effect of the TRAP on the output pin. Writing a 0_B into this field disables the effect of the TRAP function regardless of the state of the input signal.</p> <p>0_B TRAP functionality has no effect on the output 1_B TRAP functionality affects the output</p>
TRPSE	21	rw	<p>TRAP Synchronization Enable Writing a 1_B into this bit enables a synchronous exiting with the PWM signal of the trap state.</p> <p>0_B Exiting from TRAP state isn't synchronized with the PWM signal 1_B Exiting from TRAP state is synchronized with the PWM signal</p>

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
TRPSW	22	rw	<p>TRAP State Clear Control</p> <p>0_B The slice exits the TRAP state automatically when the TRAP condition is not present</p> <p>1_B The TRAP state can only be exited by a SW request.</p>
EMS	23	rw	<p>External Modulation Synchronization</p> <p>Setting this bit to 1_B enables the synchronization of the external modulation functionality with the PWM period.</p> <p>0_B External Modulation functionality is not synchronized with the PWM signal</p> <p>1_B External Modulation functionality is synchronized with the PWM signal</p>
EMT	24	rw	<p>External Modulation Type</p> <p>This field selects if the external modulation event is clearing the CC4yST bit or if is gating the outputs.</p> <p>0_B External Modulation functionality is clearing the CC4yST bit.</p> <p>1_B External Modulation functionality is gating the outputs.</p>
MCME	25	rw	<p>Multi Channel Mode Enable</p> <p>0_B Multi Channel Mode is disabled</p> <p>1_B Multi Channel Mode is enabled</p>
0	7, [20:18], [31:26]	r	<p>Reserved</p> <p>Read always returns 0</p>

CC4yPSL

This register holds the configuration for the output passive level control.

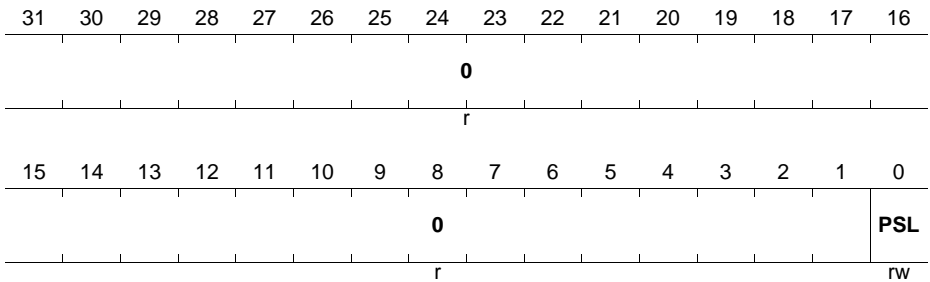
Capture/Compare Unit 4 (CCU4)

CC4yPSL (y = 0 - 3)

Passive Level Config

(0118_H + 0100_H * y)

Reset Value: 00000000_H



Field	Bits	Type	Description
PSL	0	rw	Output Passive Level This field controls the passive level of the output pin. 0 _B Passive Level is LOW 1 _B Passive Level is HIGH A write always addresses the shadow register, while a read always returns the current used value.
0	[31:1]	r	Reserved A read access always returns 0

CC4yDIT

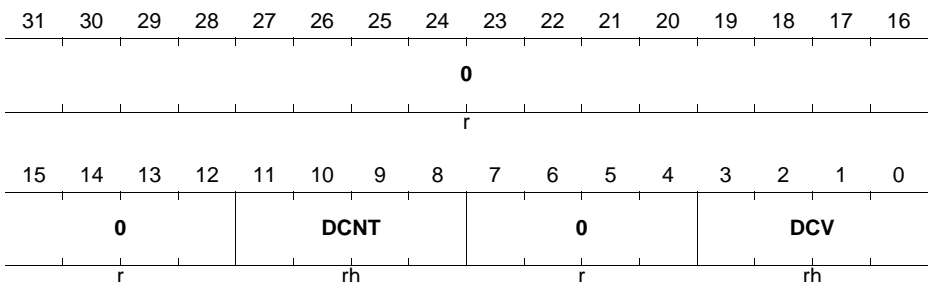
This register holds the current dither compare and dither counter values.

CC4yDIT (y = 0 - 3)

Dither Config

(011C_H + 0100_H * y)

Reset Value: 00000000_H



Capture/Compare Unit 4 (CCU4)

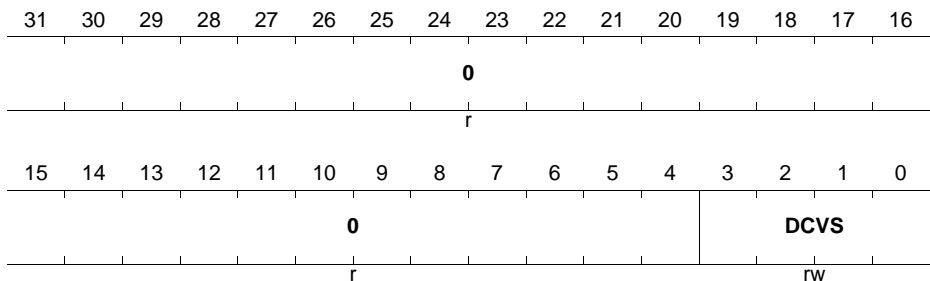
Field	Bits	Type	Description
DCV	[3:0]	rh	Dither compare Value This field contains the value used for the dither comparison. This value is updated when a shadow transfer occurs with the CC4yDITS.DCVS .
DCNT	[11:8]	rh	Dither counter actual value
0	[7:4], [31:12]	r	Reserved Read always returns 0.

CC4yDITS

This register contains the value that is going to be loaded into the **CC4yDIT.DCV** when the next shadow transfer occurs.

CC4yDITS (y = 0 - 3)

Dither Shadow Register **(0120_H + 0100_H * y)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
DCVS	[3:0]	rw	Dither Shadow Compare Value This field contains the value that is going to be set on the dither compare value, CC4yDIT.DCV , within the next shadow transfer.
0	[31:4]	r	Reserved Read always returns 0.

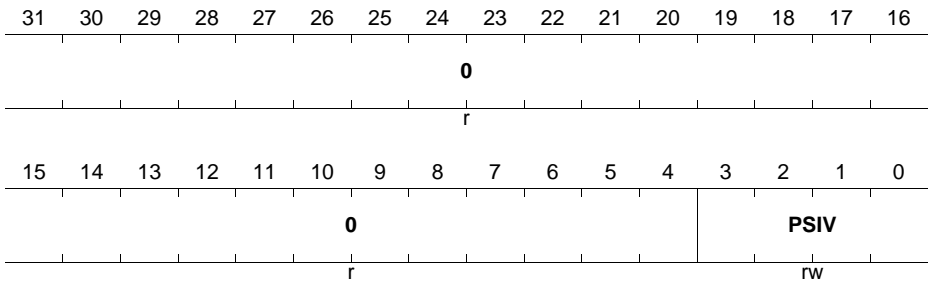
CC4yPSC

This register contains the value that is loaded into the prescaler during restart.

Capture/Compare Unit 4 (CCU4)

CC4yPSC (y = 0 - 3)

Prescaler Control (0124_H + 0100_H * y) **Reset Value: 00000000_H**



Field	Bits	Type	Description
PSIV	[3:0]	rw	Prescaler Initial Value This field contains the value that is applied to the Prescaler at startup. When floating prescaler mode is used, this value is applied when a timer compare match AND prescaler compare match occurs or when a capture event is triggered.
0	[31:4]	r	Reserved Read always returns 0.

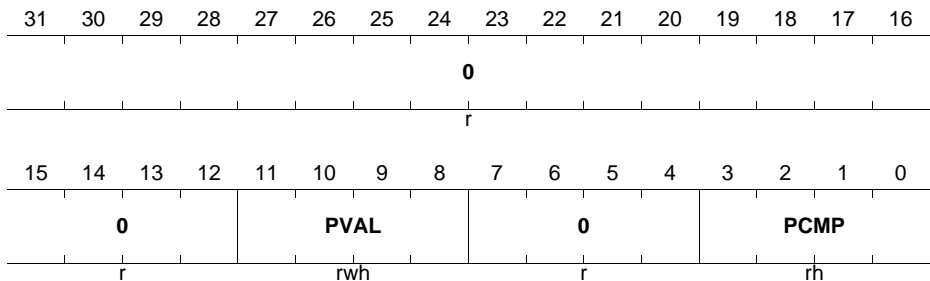
CC4yFPC

This register contains the value used for the floating prescaler compare and the actual prescaler division value.

Capture/Compare Unit 4 (CCU4)

CC4yFPC (y = 0 - 3)

Floating Prescaler Control $(0128_H + 0100_H * y)$ **Reset Value: 00000000_H**



Field	Bits	Type	Description
PCMP	[3:0]	rh	Floating Prescaler Compare Value This field contains comparison value used in floating prescaler mode. The comparison is triggered by the Timer Compare match event. See Section 21.2.11.2 .
PVAL	[11:8]	rwh	Actual Prescaler Value See Table 21-7 . Writing into this register is only possible when the prescaler is stopped. When the floating prescaler mode is not used, this value is equal to the CC4yPSC.PSIV .
0	[7:4], [15:12], [31:16]	r	Reserved Read always returns 0.

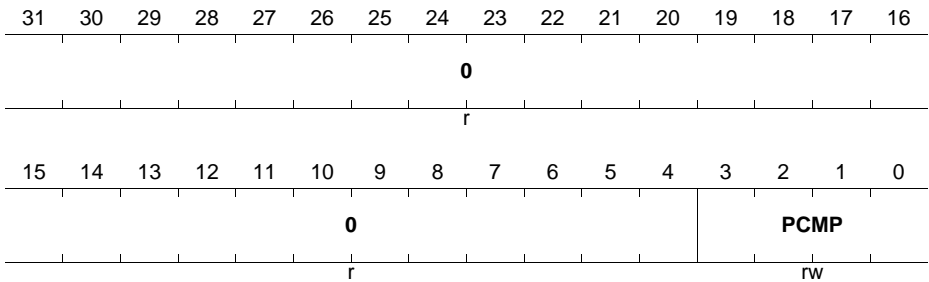
CC4yFPCS

This register contains the value that is going to be transferred to the [CC4yFPC.PCMP](#) field within the next shadow transfer update.

Capture/Compare Unit 4 (CCU4)

CC4yFPCS (y = 0 - 3)

Floating Prescaler Shadow $(012C_H + 0100_H * y)$ **Reset Value: 00000000_H**



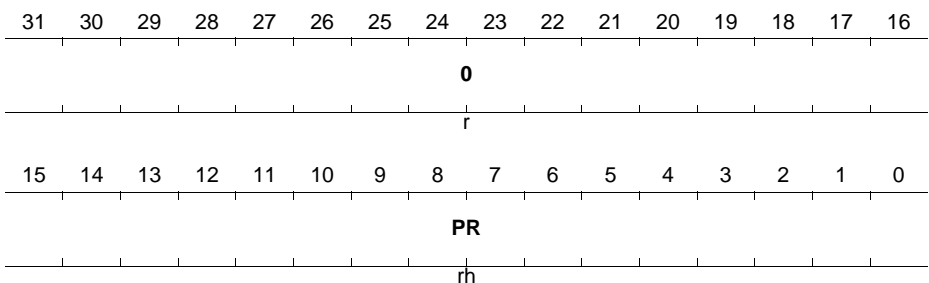
Field	Bits	Type	Description
PCMP	[3:0]	rw	Floating Prescaler Shadow Compare Value This field contains the value that is going to be set on the CC4yFPC.PCMP within the next shadow transfer. See Table 21-7 .
0	[31:4]	r	Reserved Read always returns 0.

CC4yPR

This register contains the actual value for the timer period.

CC4yPR (y = 0 - 3)

Timer Period Value $(0130_H + 0100_H * y)$ **Reset Value: 00000000_H**



Capture/Compare Unit 4 (CCU4)

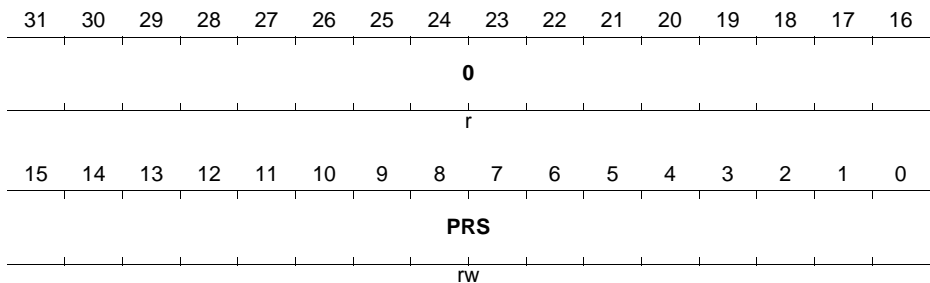
Field	Bits	Type	Description
PR	[15:0]	rh	Period Register Contains the value of the timer period. <i>Note: In Capture Mode when a external signal is selected for capturing the timer value into the capture registers 2 and 3, PR is not accessible for writing. A read always returns 0.</i>
0	[31:16]	r	Reserved A read always returns 0.

CC4yPRS

This register contains the value for the timer period that is going to be transferred into the **CC4yPR.PR** field when the next shadow transfer occurs.

CC4yPRS (y = 0 - 3)

Timer Shadow Period Value **(0134_H + 0100_H * y)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
PRS	[15:0]	rw	Period Register Contains the value of the timer period, that is going to be passed into the CC4yPR.PR field when the next shadow transfer occurs. <i>Note: In Capture Mode when a external signal is selected for capturing the timer value into the capture registers 2 and 3, the PRS is not accessible for writing. A read always returns 0.</i>

Capture/Compare Unit 4 (CCU4)

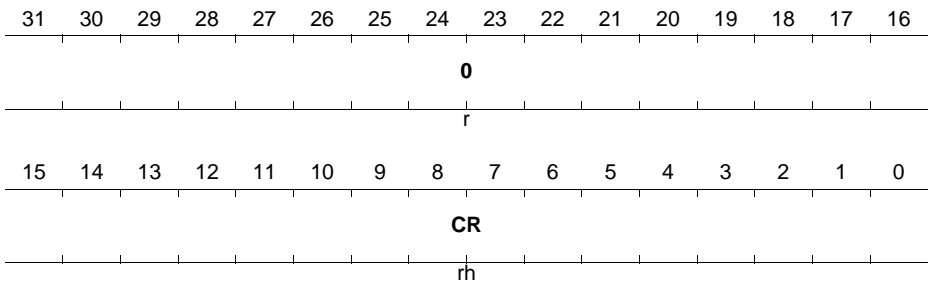
Field	Bits	Type	Description
0	[31:16]	r	Reserved A read always returns 0.

CC4yCR

This register contains the value for the timer comparison.

CC4yCR (y = 0 - 3)

Timer Compare Value (0138_H + 0100_H * y) **Reset Value: 00000000_H**



Field	Bits	Type	Description
CR	[15:0]	rh	Compare Register Contains the value for the timer comparison. <i>Note: In Capture Mode when a external signal is selected for capturing the timer value into the capture registers 0 and 1, a read always returns 0.</i>
0	[31:16]	r	Reserved A read always returns 0.

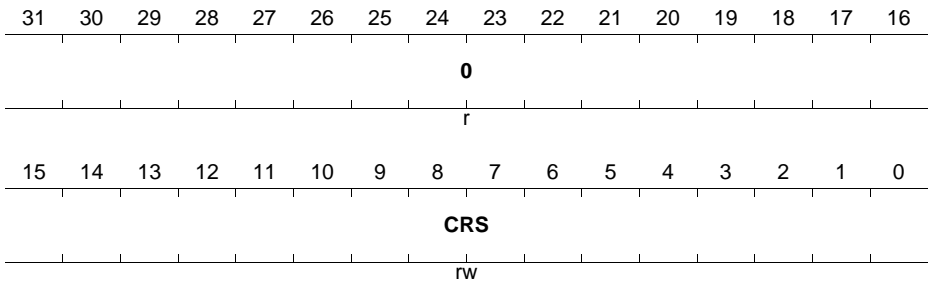
CC4yCRS

This register contains the value that is going to be loaded into the **CC4yCR.CR** field when the next shadow transfer occurs.

Capture/Compare Unit 4 (CCU4)

CC4yCRS (y = 0 - 3)

Timer Shadow Compare Value (013C_H + 0100_H * y) Reset Value: 00000000_H



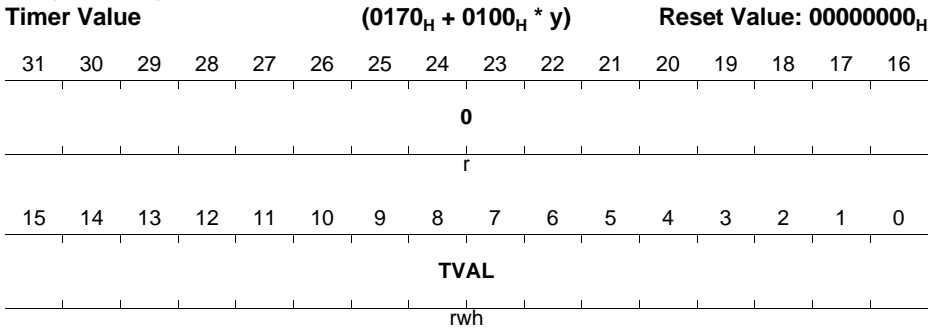
Field	Bits	Type	Description
CRS	[15:0]	rw	<p>Compare Register</p> <p>Contains the value for the timer comparison, that is going to be passed into the CC4yCR.CR field when the next shadow transfer occurs.</p> <p><i>Note: In Capture Mode when a external signal is selected for capturing the timer value into the capture registers 0 and 1, a read always returns 0.</i></p>
0	[31:16]	r	<p>Reserved</p> <p>A read always returns 0.</p>

CC4yTIMER

This register contains the current value of the timer.

Capture/Compare Unit 4 (CCU4)

CC4yTIMER (y = 0 - 3)

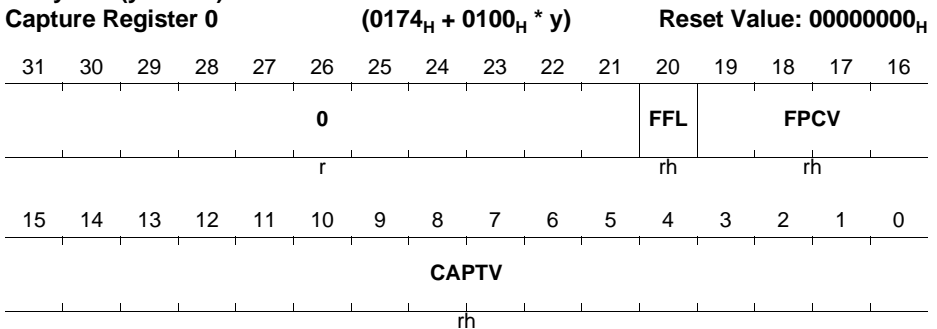


Field	Bits	Type	Description
TVALL	[15:0]	rwh	Timer Value This field contains the actual value of the timer. A write access is only possible when the timer is stopped.
0	[31:16]	r	Reserved A read access always returns 0

CC4yC0V

This register contains the values associated with the Capture 0 field.

CC4yC0V (y = 0 - 3)



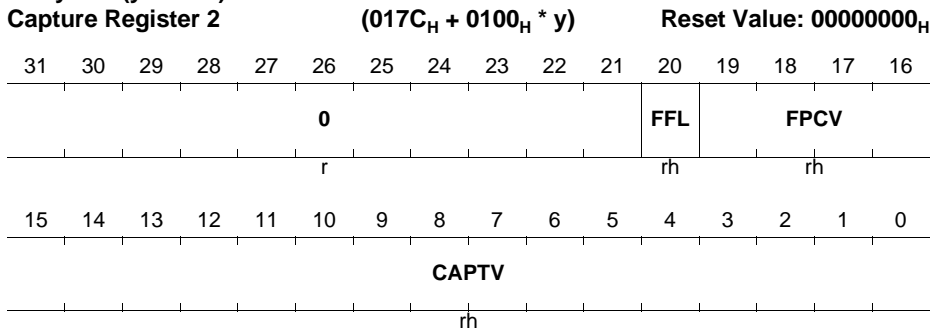
Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
CAPTV	[15:0]	rh	Capture Value This field contains the capture register 1 value. See Figure 21-27 . In compare mode a read access always returns 0.
FPCV	[19:16]	rh	Prescaler Value This field contains the prescaler value at the time of the capture event into the capture register 1. In compare mode a read access always returns 0.
FFL	20	rh	Full Flag This bit indicates if a new value was capture into the capture register 1 after the last read access. See Figure 21-27 . In compare mode a read access always returns 0. 0_B No new value was captured into the specific capture register 1_B A new value was captured into the specific register
0	[31:21]	r	Reserved A read always returns 0

CC4yC2V

This register contains the values associated with the Capture 2 field.

CC4yC2V (y = 0 - 3)



Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
CAPTV	[15:0]	rh	Capture Value This field contains the capture register 3 value. See Figure 21-27 . In compare mode a read access always returns 0.
FPCV	[19:16]	rh	Prescaler Value This field contains the prescaler value at the time of the capture event into the capture register 3. In compare mode a read access always returns 0.
FFL	20	rh	Full Flag This bit indicates if a new value was capture into the capture register 3 after the last read access. See Figure 21-27 . In compare mode a read access always returns 0. 0 _B No new value was captured into the specific capture register 1 _B A new value was captured into the specific register
0	[31:21]	r	Reserved A read always returns 0

CC4yINTS

This register contains the status of all interrupt sources.

CC4yINTS (y = 0 - 3)

Interrupt Status										(01A0 _H + 0100 _H * y)				Reset Value: 00000000 _H			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
0																	
r																	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0				TRP	E2A	E1A	E0A	0				CMD	CMU	OMD	PMU		
r				rh	rh	rh	rh	r				rh	rh	rh	rh		

Field	Bits	Type	Description
PMUS	0	rh	<p>Period Match while Counting Up</p> <p>0_B Period match while counting up not detected 1_B Period match while counting up detected</p>
OMDS	1	rh	<p>One Match while Counting Down</p> <p>0_B One match while counting down not detected 1_B One match while counting down detected</p>
CMUS	2	rh	<p>Compare Match while Counting Up</p> <p>0_B Compare match while counting up not detected 1_B Compare match while counting up detected</p>
CMDS	3	rh	<p>Compare Match while Counting Down</p> <p>0_B Compare match while counting down not detected 1_B Compare match while counting down detected</p>
E0AS	8	rh	<p>Event 0 Detection Status</p> <p>Depending on the user selection on the CC4yINS.EV0EM, this bit can be set when a rising, falling or both transitions are detected.</p> <p>0_B Event 0 not detected 1_B Event 0 detected</p>
E1AS	9	rh	<p>Event 1 Detection Status</p> <p>Depending on the user selection on the CC4yINS.EV1EM, this bit can be set when a rising, falling or both transitions are detected.</p> <p>0_B Event 1 not detected 1_B Event 1 detected</p>
E2AS	10	rh	<p>Event 2 Detection Status</p> <p>Depending on the user selection on the CC4yINS.EV1EM, this bit can be set when a rising, falling or both transitions are detected.</p> <p>0_B Event 2 not detected 1_B Event 2 detected</p> <p><i>Note: If this event is linked with the TRAP function, this field is automatically cleared when the slice exits the Trap State.</i></p>
TRPF	11	rh	<p>Trap Flag Status</p> <p>This field contains the status of the Trap Flag.</p>

Capture/Compare Unit 4 (CCU4)

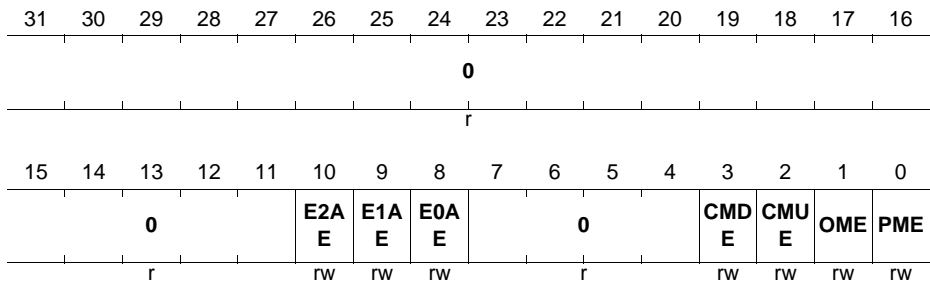
Field	Bits	Type	Description
0	[7:4], [31:12]	r	Reserved A read always returns 0.

CC4yINTE

Through this register it is possible to enable or disable the specific interrupt source(s).

CC4yINTE (y = 0 - 3)

Interrupt Enable Control **(01A4_H + 0100_H * y)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
PME	0	rw	Period match while counting up enable Setting this bit to 1 _B enables the generation of an interrupt pulse every time a period match while counting up occurs. 0 _B Period Match interrupt is disabled 1 _B Period Match interrupt is enabled
OME	1	rw	One match while counting down enable Setting this bit to 1 _B enables the generation of an interrupt pulse every time an one match while counting down occurs. 0 _B One Match interrupt is disabled 1 _B One Match interrupt is enabled

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
CMUE	2	rw	<p>Compare match while counting up enable Setting this bit to 1_B enables the generation of an interrupt pulse every time a compare match while counting up occurs.</p> <p>0_B Compare Match while counting up interrupt is disabled</p> <p>1_B Compare Match while counting up interrupt is enabled</p>
CMDE	3	rw	<p>Compare match while counting down enable Setting this bit to 1_B enables the generation of an interrupt pulse every time a compare match while counting down occurs.</p> <p>0_B Compare Match while counting down interrupt is disabled</p> <p>1_B Compare Match while counting down interrupt is enabled</p>
E0AE	8	rw	<p>Event 0 interrupt enable Setting this bit to 1_B enables the generation of an interrupt pulse every time that Event 0 is detected.</p> <p>0_B Event 0 detection interrupt is disabled</p> <p>1_B Event 0 detection interrupt is enabled</p>
E1AE	9	rw	<p>Event 1 interrupt enable Setting this bit to 1_B enables the generation of an interrupt pulse every time that Event 1 is detected.</p> <p>0_B Event 1 detection interrupt is disabled</p> <p>1_B Event 1 detection interrupt is enabled</p>
E2AE	10	rw	<p>Event 2 interrupt enable Setting this bit to 1_B enables the generation of an interrupt pulse every time that Event 2 is detected.</p> <p>0_B Event 2 detection interrupt is disabled</p> <p>1_B Event 2 detection interrupt is enabled</p>
0	[7:4], [31:11]	r	<p>Reserved A read always returns 0</p>

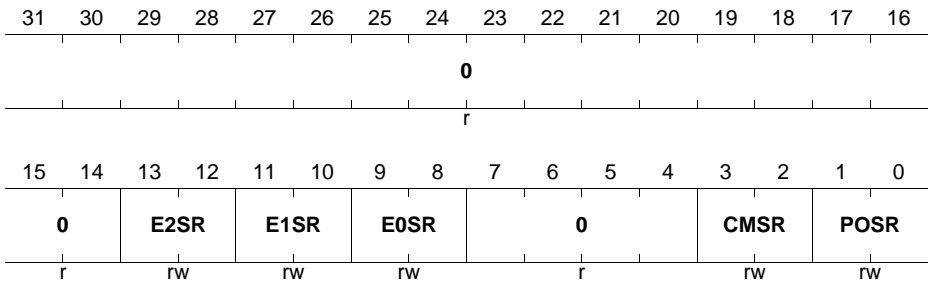
CC4ySRS

Through this register it is possible to select to which service request line each interrupt source is forwarded.

Capture/Compare Unit 4 (CCU4)

CC4ySRS (y = 0 - 3)

Service Request Selector (01A8_H + 0100_H * y) Reset Value: 00000000_H



Field	Bits	Type	Description
POSR	[1:0]	rw	<p>Period/One match Service request selector This field selects to which slice Service request line, the interrupt(s) generated by the Period match while counting up and One match while counting down are going to be forward.</p> <p>00_B Forward to CC4ySR0 01_B Forward to CC4ySR1 10_B Forward to CC4ySR2 11_B Forward to CC4ySR3</p>
CMSR	[3:2]	rw	<p>Compare match Service request selector This field selects to which slice Service request line, the interrupt(s) generated by the Compare match while counting up and Compare match while counting down are going to be forward.</p> <p>00_B Forward to CC4ySR0 01_B Forward to CC4ySR1 10_B Forward to CC4ySR2 11_B Forward to CC4ySR3</p>
E0SR	[9:8]	rw	<p>Event 0 Service request selector This field selects to which slice Service request line, the interrupt generated by the Event 0 detection is going to be forward.</p> <p>00_B Forward to CC4ySR0 01_B Forward to CC4ySR1 10_B Forward to CC4ySR2 11_B Forward to CC4ySR3</p>

Capture/Compare Unit 4 (CCU4)

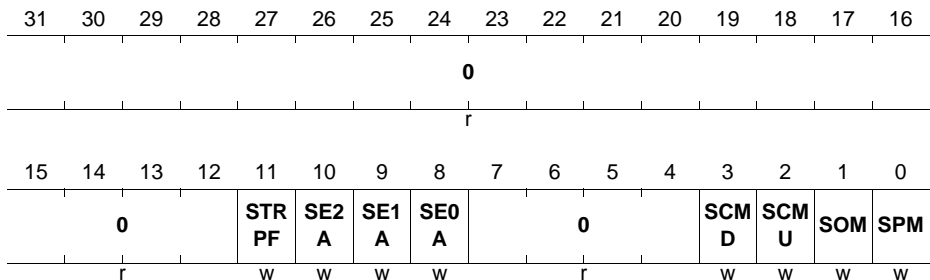
Field	Bits	Type	Description
E1SR	[11:10]	rw	Event 1 Service request selector This field selects to which slice Service request line, the interrupt generated by the Event 1 detection is going to be forward. 00 _B Forward to CC4ySR0 01 _B Forward to CC4ySR1 10 _B Forward to CC4ySR2 11 _B Forward to CC4ySR3
E2SR	[13:12]	rw	Event 2 Service request selector This field selects to which slice Service request line, the interrupt generated by the Event 2 detection is going to be forward. 00 _B Forward to CC4ySR0 01 _B Forward to CC4ySR1 10 _B Forward to CC4ySR2 11 _B Forward to CC4ySR3
0	[7:4], [31:14]	r	Reserved Read always returns 0.

CC4ySWS

Through this register it is possible for the SW to set a specific interrupt status flag.

CC4ySWS (y = 0 - 3)

Interrupt Status Set (01AC_H + 0100_H * y) **Reset Value: 00000000_H**



Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
SPM	0	w	Period match while counting up set Writing a 1 _B into this field sets the CC4yINTS.PMUS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SOM	1	w	One match while counting down set Writing a 1 _B into this bit sets the CC4yINTS.OMDS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SCMU	2	w	Compare match while counting up set Writing a 1 _B into this field sets the CC4yINTS.CMUS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SCMD	3	w	Compare match while counting down set Writing a 1 _B into this bit sets the CC4yINTS.CMDS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SE0A	8	w	Event 0 detection set Writing a 1 _B into this bit sets the CC4yINTS.E0AS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SE1A	9	w	Event 1 detection set Writing a 1 _B into this bit sets the CC4yINTS.E1AS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SE2A	10	w	Event 2 detection set Writing a 1 _B into this bit sets the CC4yINTS.E2AS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
STRPF	11	w	Trap Flag status set Writing a 1 _B into this bit sets the CC4yINTS.TRPF bit. A read always returns 0.
0	[7:4], [31:12]	r	Reserved Read always returns 0

CC4ySWR

Through this register it is possible for the SW to clear a specific interrupt status flag.

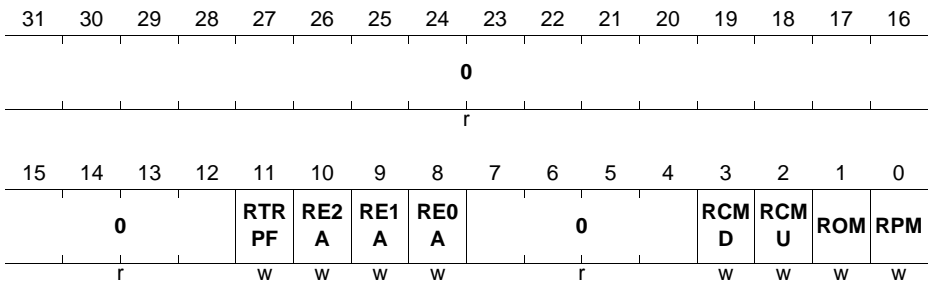
Capture/Compare Unit 4 (CCU4)

CC4ySWR (y = 0 - 3)

Interrupt Status Clear

(01B0_H + 0100_H * y)

Reset Value: 00000000_H



Field	Bits	Type	Description
RPM	0	w	Period match while counting up clear Writing a 1 _B into this field clears the CC4yINTS .PMUS bit. A read always returns 0.
ROM	1	w	One match while counting down clear Writing a 1 _B into this bit clears the CC4yINTS .OMDS bit. A read always returns 0.
RCMU	2	w	Compare match while counting up clear Writing a 1 _B into this field clears the CC4yINTS .CMUS bit. A read always returns 0.
RCMD	3	w	Compare match while counting down clear Writing a 1 _B into this bit clears the CC4yINTS .CMDS bit. A read always returns 0.
RE0A	8	w	Event 0 detection clear Writing a 1 _B into this bit clears the CC4yINTS .E0AS bit. A read always returns 0.
RE1A	9	w	Event 1 detection clear Writing a 1 _B into this bit clears the CC4yINTS .E1AS bit. A read always returns 0.
RE2A	10	w	Event 2 detection clear Writing a 1 _B into this bit clears the CC4yINTS .E2AS bit. A read always returns 0.

Capture/Compare Unit 4 (CCU4)

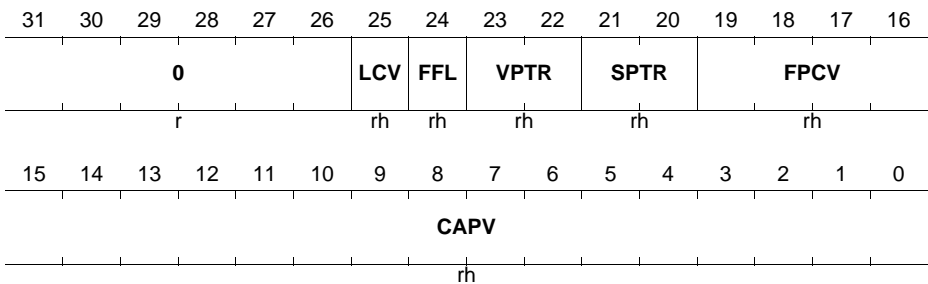
Field	Bits	Type	Description
RTRPF	11	w	Trap Flag status clear Writing a 1 _B into this bit clears the CC4yINTS .TRPF bit. Not valid if CC4yTC .TRPEN = 1 _B and the Trap State is still active. A read always returns 0.
0	[7:4], [31:12]	r	Reserved Read always returns 0

CC4yECRD0

Through this register it is possible to read back the FIFO structure of the capture function that is linked with the capture trigger 0. The read back is only valid if the **CC4yTC**.ECM = 1_B.

CC4yECRD0 (y = 0 - 3)

Extended Read Back 0 (01B8_H + 0100_H * y) **Reset Value: 00000000_H**



Field	Bits	Type	Description
CAPV	[15:0]	rh	Timer Capture Value This field contains the timer captured value
FPCV	[19:16]	rh	Prescaler Capture value This field contains the value of the prescaler clock division associated with the specific CAPV field

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
SPTR	[21:20]	rh	Slice pointer This field indicates the slice index in which the value was captured. 00 _B CC40 01 _B CC41 10 _B CC42 11 _B CC43
VPTR	[23:22]	rh	Capture register pointer This field indicates the capture register index in which the value was captured. 00 _B Capture register 0 01 _B Capture register 1 10 _B Capture register 2 11 _B Capture register 3
FFL	24	rh	Full Flag This bit indicates if the associated capture register contains a new value. 0 _B No new value was captured into this register 1 _B A new value has been captured into this register
LCV	25	rh	Lost Capture Value This field indicates if between two reads of the ECRD0 a capture trigger occurred while the FIFO structure was full. If a capture trigger occurred between two reads than a capture value was lost. This field is automatically cleared by the HW whenever a read to the ECRD occurs. 0 _B No capture was lost 1 _B A capture was lost
0	[31:26]	r	Reserved Read always returns 0

CC4yECRD1

Through this register it is possible to read back the FIFO structure of the capture function that is linked with the capture trigger 1. The read back is only valid if the **CC4yTC.ECM** = 1_B.

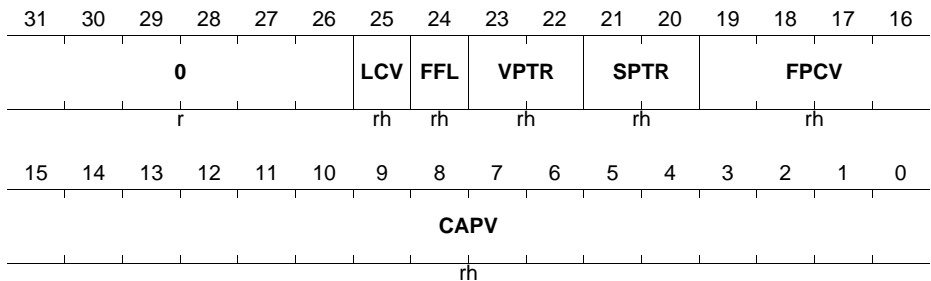
Capture/Compare Unit 4 (CCU4)

CC4yECRD1 (y = 0 - 3)

Extended Read Back 1

(01BC_H + 0100_H * y)

Reset Value: 00000000_H



Field	Bits	Type	Description
CAPV	[15:0]	rh	Timer Capture Value This field contains the timer captured value
FPCV	[19:16]	rh	Prescaler Capture value This field contains the value of the prescaler clock division associated with the specific CAPV field
SPTR	[21:20]	rh	Slice pointer This field indicates the slice index in which the value was captured. 00 _B CC40 01 _B CC41 10 _B CC42 11 _B CC43
VPTR	[23:22]	rh	Capture register pointer This field indicates the capture register index in which the value was captured. 00 _B Capture register 0 01 _B Capture register 1 10 _B Capture register 2 11 _B Capture register 3
FFL	24	rh	Full Flag This bit indicates if the associated capture register contains a new value. 0 _B No new value was captured into this register 1 _B A new value has been captured into this register

Capture/Compare Unit 4 (CCU4)

Field	Bits	Type	Description
LCV	25	rh	Lost Capture Value This field indicates if between two reads of the ECRD0 a capture trigger occurred while the FIFO structure was full. If a capture trigger occurred between two reads than a capture value was lost. This field is automatically cleared by the HW whenever a read to the ECRD occurs. 0 _B No capture was lost 1 _B A capture was lost
0	[31:26]	r	Reserved Read always returns 0

21.8 Interconnects

The tables that refer to the “global pins” are the ones that contain the inputs/outputs of each module that are common to all slices.

The GPIO connections are available at the Ports chapter.

21.8.1 CCU40 pins

Table 21-13 CCU40 Pin Connections

Global Inputs/Outputs	I/O	Connected To	Description
CCU40.MCLK	I	SCU.CCUCLK	Kernel clock
CCU40.CLKA	I (s)	ERU1.IOUT0	another count source for the prescaler
CCU40.CLKB	I (s)	ERU1.IOUT1	another count source for the prescaler
CCU40.CLKC	I (s)	0	another count source for the prescaler
CCU40.MCSS	I	0	Multi pattern sync with shadow transfer trigger
CCU40.SR0	O	NVIC; DMA; VADC.G0REQGTC; VADC.G1REQGTC; VADC.BGREQGTC;	Service request line

Table 21-13 CCU40 Pin Connections

Global Inputs/Outputs	I/O	Connected To	Description
CCU40.SR1	O	NVIC; DMA; DAC.TRIGGER[2]; U0C0.DX2E;	Service request line
CCU40.SR2	O	NVIC; VADC.G0REQTRA; VADC.G1REQTRA; VADC.BGREQTRA;	Service request line
CCU40.SR3	O	NVIC; CCU80.IGBTB; VADC.G0REQTRB; VADC.G1REQTRB; VADC.BGREQTRB; CCU80.IN0K;	Service request line

Table 21-14 CCU40 - CC40 Pin Connections

Input/Output	I/O	Connected To	Description
CCU40.IN0A	I (s)	PORTS	General purpose function
CCU40.IN0B	I (s)	PORTS	General purpose function
CCU40.IN0C	I (s)	PORTS	General purpose function
CCU40.IN0D	I (s)	ERU1.PDOUT1	General purpose function
CCU40.IN0E	I (s)	NOT CONNECTED	General purpose function
CCU40.IN0F	I (s)	NOT CONNECTED	General purpose function
CCU40.IN0G	I (s)	NOT CONNECTED	General purpose function
CCU40.IN0H	I (s)	CAN.SR7	General purpose function
CCU40.IN0I	I (s)	SCU.GSC40	General purpose function
CCU40.IN0J	I (s)	ERU1.PDOUT0	General purpose function
CCU40.IN0K	I (s)	ERU1.IOUT0	General purpose function
CCU40.IN0L	I (s)	U0C0.DX2INS	General purpose function
CCU40.IN0M	I (s)	CCU40.ST0	General purpose function
CCU40.IN0N	I (s)	CCU40.ST1	General purpose function
CCU40.IN0O	I (s)	CCU40.ST2	General purpose function

Capture/Compare Unit 4 (CCU4)

Table 21-14 CCU40 - CC40 Pin Connections

Input/Output	I/O	Connected To	Description
CCU40.IN0P	I (s)	CCU40.ST3	General purpose function
CCU40.MCI0	I	0	Multi Channel pattern input
CCU40.OUT0	O	PORTS	Slice compare output
CCU40.GP00	O	NOT CONNECTED	Selected signal for event 0
CCU40.GP01	O	NOT CONNECTED	Selected signal for event 1
CCU40.GP02	O	NOT CONNECTED	Selected signal for event 2
CCU40.ST0	O	ERU1.0A2; ERU1.0GU02;	Slice status bit
CCU40.PS0	O	NOT CONNECTED	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

Table 21-15 CCU40 - CC41 Pin Connections

Input/Output	I/O	Connected To	Description
CCU40.IN1A	I (s)	PORTS	General purpose function
CCU40.IN1B	I (s)	PORTS	General purpose function
CCU40.IN1C	I (s)	PORTS	General purpose function
CCU40.IN1D	I (s)	ERU1.PDOU0	General purpose function
CCU40.IN1E	I (s)	POSIF0.OUT0	General purpose function
CCU40.IN1F	I (s)	POSIF0.OUT1	General purpose function
CCU40.IN1G	I (s)	POSIF0.OUT3	General purpose function
CCU40.IN1H	I (s)	POSIF0.OUT4	General purpose function
CCU40.IN1I	I (s)	SCU.GSC40	General purpose function
CCU40.IN1J	I (s)	ERU1.PDOU1	General purpose function
CCU40.IN1K	I (s)	ERU1.IOU1	General purpose function
CCU40.IN1L	I (s)	NOT CONNECTED	General purpose function
CCU40.IN1M	I (s)	CCU40.ST0	General purpose function
CCU40.IN1N	I (s)	CCU40.ST1	General purpose function
CCU40.IN1O	I (s)	CCU40.ST2	General purpose function
CCU40.IN1P	I (s)	CCU40.ST3	General purpose function

Capture/Compare Unit 4 (CCU4)

Table 21-15 CCU40 - CC41 Pin Connections

Input/Output	I/O	Connected To	Description
CCU40.MCI1	I	0	Multi Channel pattern input
CCU40.OUT1	O	PORTS	Slice compare output
CCU40.GP10	O	NOT CONNECTED	Selected signal for event 0
CCU40.GP11	O	NOT CONNECTED	Selected signal for event 1
CCU40.GP12	O	NOT CONNECTED	Selected signal for event 2
CCU40.ST1	O	ERU1.1A2	Slice status bit
CCU40.PS1	O	POSIF0.SYNCC	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

Table 21-16 CCU40 - CC42 Pin Connections

Input/Output	I/O	Connected To	Description
CCU40.IN2A	I (s)	PORTS	General purpose function
CCU40.IN2B	I (s)	PORTS	General purpose function
CCU40.IN2C	I (s)	PORTS	General purpose function
CCU40.IN2D	I (s)	ERU1.PDOU0	General purpose function
CCU40.IN2E	I (s)	NOT CONNECTED	General purpose function
CCU40.IN2F	I (s)	NOT CONNECTED	General purpose function
CCU40.IN2G	I (s)	NOT CONNECTED	General purpose function
CCU40.IN2H	I (s)	NOT CONNECTED	General purpose function
CCU40.IN2I	I (s)	SCU.GSC40	General purpose function
CCU40.IN2J	I (s)	ERU1.PDOU2	General purpose function
CCU40.IN2K	I (s)	ERU1.IOU2	General purpose function
CCU40.IN2L	I (s)	U0C1.DX2INS	General purpose function
CCU40.IN2M	I (s)	CCU40.ST0	General purpose function
CCU40.IN2N	I (s)	CCU40.ST1	General purpose function
CCU40.IN2O	I (s)	CCU40.ST2	General purpose function
CCU40.IN2P	I (s)	CCU40.ST3	General purpose function
CCU40.MCI2	I	0	Multi Channel pattern input
CCU40.OUT2	O	PORTS	Slice compare output

Capture/Compare Unit 4 (CCU4)

Table 21-16 CCU40 - CC42 Pin Connections

Input/Output	I/O	Connected To	Description
CCU40.GP20	O	NOT CONNECTED	Selected signal for event 0
CCU40.GP21	O	NOT CONNECTED	Selected signal for event 1
CCU40.GP22	O	NOT CONNECTED	Selected signal for event 2
CCU40.ST2	O	ERU1.2A2	Slice status bit
CCU40.PS2	O	NOT CONNECTED	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

Table 21-17 CCU40 - CC43 Pin Connections

Input/Output	I/O	Connected To	Description
CCU40.IN3A	I (s)	PORTS	General purpose function
CCU40.IN3B	I (s)	PORTS	General purpose function
CCU40.IN3C	I (s)	PORTS	General purpose function
CCU40.IN3D	I (s)	ERU1.PDOU0	General purpose function
CCU40.IN3E	I (s)	NOT CONNECTED	General purpose function
CCU40.IN3F	I (s)	NOT CONNECTED	General purpose function
CCU40.IN3G	I (s)	VADC.G0ARBCNT	General purpose function
CCU40.IN3H	I (s)	CCU80.IGBTO	General purpose function
CCU40.IN3I	I (s)	SCU.GSC40	General purpose function
CCU40.IN3J	I (s)	ERU1.PDOU3	General purpose function
CCU40.IN3K	I (s)	ERU1.IOU3	General purpose function
CCU40.IN3L	I (s)	U1C0.DX2INS	General purpose function
CCU40.IN3M	I (s)	CCU40.ST0	General purpose function
CCU40.IN3N	I (s)	CCU40.ST1	General purpose function
CCU40.IN3O	I (s)	CCU40.ST2	General purpose function
CCU40.IN3P	I (s)	CCU40.ST3	General purpose function
CCU40.MCI3	I	0	Multi Channel pattern input
CCU40.OUT3	O	PORTS	Slice compare output
CCU40.GP30	O	NOT CONNECTED	Selected signal for event 0
CCU40.GP31	O	NOT CONNECTED	Selected signal for event 1

Capture/Compare Unit 4 (CCU4)

Table 21-17 CCU40 - CC43 Pin Connections

Input/Output	I/O	Connected To	Description
CCU40.GP32	O	NOT CONNECTED	Selected signal for event 2
CCU40.ST3	O	VADC.G0REQGTA; VADC.G1REQGTA; VADC.BGREQGTA; ERU1.3A2; CCU80.IGBTA;	Slice status bit
CCU40.PS3	O	NOT CONNECTED	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

21.8.2 CCU41 pins

Table 21-18 CCU41 Pin Connections

Global Inputs/Outputs	I/O	Connected To	Description
CCU41.MCLK	I	SCU.CCUCLK	Kernel clock
CCU41.CLKA	I (s)	ERU1.IOUT0	another count source for the prescaler
CCU41.CLKB	I (s)	ERU1.IOUT1	another count source for the prescaler
CCU41.CLKC	I (s)	0	another count source for the prescaler
CCU41.MCSS	I	0	Multi pattern sync with shadow transfer trigger
CCU41.SR0	O	NVIC; DMA;	Service request line
CCU41.SR1	O	NVIC; DMA; DAC.TRIGGER[3]; VADC.G0REQGTD; VADC.G1REQGTD; VADC.BGREQGTD; U1C0.DX2E;	Service request line

Table 21-18 CCU41 Pin Connections

Global Inputs/Outputs	I/O	Connected To	Description
CCU41.SR2	O	NVIC; VADC.G0REQTRC; VADC.G1REQTRC; VADC.BGREQTRC;	Service request line
CCU41.SR3	O	NVIC; VADC.G0REQTRD; VADC.G1REQTRD; VADC.BGREQTRD; CCU80.IN1K;	Service request line

Table 21-19 CCU41 - CC40 Pin Connections

Input/Output	I/O	Connected To	Description
CCU41.IN0A	I (s)	PORTS	General purpose function
CCU41.IN0B	I (s)	PORTS	General purpose function
CCU41.IN0C	I (s)	PORTS	General purpose function
CCU41.IN0D	I (s)	ERU1.PDOU1	General purpose function
CCU41.IN0E	I (s)	NOT CONNECTED	General purpose function
CCU41.IN0F	I (s)	NOT CONNECTED	General purpose function
CCU41.IN0G	I (s)	NOT CONNECTED	General purpose function
CCU41.IN0H	I (s)	CAN.SR7	General purpose function
CCU41.IN0I	I (s)	SCU.GSC41	General purpose function
CCU41.IN0J	I (s)	ERU1.PDOU0	General purpose function
CCU41.IN0K	I (s)	ERU1.IOU0	General purpose function
CCU41.IN0L	I (s)	VADC.G0BFL0	General purpose function
CCU41.IN0M	I (s)	CCU41.ST0	General purpose function
CCU41.IN0N	I (s)	CCU41.ST1	General purpose function
CCU41.IN0O	I (s)	CCU41.ST2	General purpose function
CCU41.IN0P	I (s)	CCU41.ST3	General purpose function
CCU41.MCI0	I	0	Multi Channel pattern input
CCU41.OUT0	O	PORTS	Slice compare output
CCU41.GP00	O	NOT CONNECTED	Selected signal for event 0

Capture/Compare Unit 4 (CCU4)

Table 21-19 CCU41 - CC40 Pin Connections

Input/Output	I/O	Connected To	Description
CCU41.GP01	O	NOT CONNECTED	Selected signal for event 1
CCU41.GP02	O	NOT CONNECTED	Selected signal for event 2
CCU41.ST0	O	ERU1.OGU12	Slice status bit
CCU41.PS0	O	NOT CONNECTED	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

Table 21-20 CCU41 - CC41 Pin Connections

Input/Output	I/O	Connected To	Description
CCU41.IN1A	I (s)	PORTS	General purpose function
CCU41.IN1B	I (s)	PORTS	General purpose function
CCU41.IN1C	I (s)	PORTS	General purpose function
CCU41.IN1D	I (s)	ERU1.PDOU0	General purpose function
CCU41.IN1E	I (s)	NOT CONNECTED	General purpose function
CCU41.IN1F	I (s)	NOT CONNECTED	General purpose function
CCU41.IN1G	I (s)	NOT CONNECTED	General purpose function
CCU41.IN1H	I (s)	NOT CONNECTED	General purpose function
CCU41.IN1I	I (s)	SCU.GSC41	General purpose function
CCU41.IN1J	I (s)	ERU1.PDOU1	General purpose function
CCU41.IN1K	I (s)	ERU1.IOU1	General purpose function
CCU41.IN1L	I (s)	NOT CONNECTED	General purpose function
CCU41.IN1M	I (s)	CCU41.ST0	General purpose function
CCU41.IN1N	I (s)	CCU41.ST1	General purpose function
CCU41.IN1O	I (s)	CCU41.ST2	General purpose function
CCU41.IN1P	I (s)	CCU41.ST3	General purpose function
CCU41.MC11	I	0	Multi Channel pattern input
CCU41.OUT1	O	PORTS	Slice compare output
CCU41.GP10	O	NOT CONNECTED	Selected signal for event 0
CCU41.GP11	O	NOT CONNECTED	Selected signal for event 1
CCU41.GP12	O	NOT CONNECTED	Selected signal for event 2

Capture/Compare Unit 4 (CCU4)

Table 21-20 CCU41 - CC41 Pin Connections

Input/Output	I/O	Connected To	Description
CCU41.ST1	O	NOT CONNECTED	Slice status bit
CCU41.PS1	O	POSIF1.MSYNCC	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

Table 21-21 CCU41 - CC42 Pin Connections

Input/Output	I/O	Connected To	Description
CCU41.IN2A	I (s)	PORTS	General purpose function
CCU41.IN2B	I (s)	PORTS	General purpose function
CCU41.IN2C	I (s)	PORTS	General purpose function
CCU41.IN2D	I (s)	ERU1.PDOU0	General purpose function
CCU41.IN2E	I (s)	NOT CONNECTED	General purpose function
CCU41.IN2F	I (s)	NOT CONNECTED	General purpose function
CCU41.IN2G	I (s)	NOT CONNECTED	General purpose function
CCU41.IN2H	I (s)	NOT CONNECTED	General purpose function
CCU41.IN2I	I (s)	SCU.GSC41	General purpose function
CCU41.IN2J	I (s)	ERU1.PDOU2	General purpose function
CCU41.IN2K	I (s)	ERU1.IOU2	General purpose function
CCU41.IN2L	I (s)	VADC.G0BFL1	General purpose function
CCU41.IN2M	I (s)	CCU41.ST0	General purpose function
CCU41.IN2N	I (s)	CCU41.ST1	General purpose function
CCU41.IN2O	I (s)	CCU41.ST2	General purpose function
CCU41.IN2P	I (s)	CCU41.ST3	General purpose function
CCU41.MCI2	I	0	Multi Channel pattern input
CCU41.OUT2	O	PORTS	Slice compare output
CCU41.GP20	O	NOT CONNECTED	Selected signal for event 0
CCU41.GP21	O	NOT CONNECTED	Selected signal for event 1
CCU41.GP22	O	NOT CONNECTED	Selected signal for event 2

Capture/Compare Unit 4 (CCU4)

Table 21-21 CCU41 - CC42 Pin Connections

Input/Output	I/O	Connected To	Description
CCU41.ST2	O	NOT CONNECTED	Slice status bit
CCU41.PS2	O	NOT CONNECTED	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

Table 21-22 CCU41 - CC43 Pin Connections

Input/Output	I/O	Connected To	Description
CCU41.IN3A	I (s)	PORTS	General purpose function
CCU41.IN3B	I (s)	PORTS	General purpose function
CCU41.IN3C	I (s)	PORTS	General purpose function
CCU41.IN3D	I (s)	ERU1.PDOU0	General purpose function
CCU41.IN3E	I (s)	NOT CONNECTED	General purpose function
CCU41.IN3F	I (s)	NOT CONNECTED	General purpose function
CCU41.IN3G	I (s)	VADC.G1ARBCNT	General purpose function
CCU41.IN3H	I (s)	NOT CONNECTED	General purpose function
CCU41.IN3I	I (s)	SCU.GSC41	General purpose function
CCU41.IN3J	I (s)	ERU1.PDOU3	General purpose function
CCU41.IN3K	I (s)	ERU1.IOU3	General purpose function
CCU41.IN3L	I (s)	VADC.G0BFL2	General purpose function
CCU41.IN3M	I (s)	CCU41.ST0	General purpose function
CCU41.IN3N	I (s)	CCU41.ST1	General purpose function
CCU41.IN3O	I (s)	CCU41.ST2	General purpose function
CCU41.IN3P	I (s)	CCU41.ST3	General purpose function
CCU41.MCI3	I	0	Multi Channel pattern input
CCU41.OUT3	O	PORTS	Slice compare output
CCU41.GP30	O	NOT CONNECTED	Selected signal for event 0
CCU41.GP31	O	NOT CONNECTED	Selected signal for event 1
CCU41.GP32	O	NOT CONNECTED	Selected signal for event 2

Table 21-22 CCU41 - CC43 Pin Connections

Input/Output	I/O	Connected To	Description
CCU41.ST3	O	VADC.G0REQGTB; VADC.G1REQGTB; VADC.BGREQGTB;	Slice status bit
CCU41.PS3	O	NOT CONNECTED	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

22 Capture/Compare Unit 8 (CCU8)

The CCU8 peripheral functions play a major role in applications that need complex Pulse Width Modulation (PWM) signal generation, with complementary high side and low side switches, multi phase control or output parity checking. These functions in conjunction with a very flexible and programmable signal conditioning scheme, make the CCU8 the must have peripheral for state of the art motor control, multi phase and multi level power electronics systems.

The internal modularity of CCU8, translates into a software friendly system for fast code development and portability between applications.

Table 22-1 Abbreviations table

PWM	Pulse Width Modulation
CCU8x	Capture/Compare Unit 8 module instance x
CC8y	Capture/Compare Unit 8 Timer Slice instance y
ADC	Analog to Digital Converter
POSIF	Position Interface peripheral
SCU	System Control Unit
f_{ccu8}	CCU8 module clock frequency
f_{tclk}	CC8y timer clock frequency

Note: A small “y” or “x” letter in a register indicates an index

22.1 Overview

The CCU8 unit is comprised of four identical 16 bit Capture/Compare Timer slices, CC8y. Each Timer Slice can work in Compare or in Capture Mode. In Compare Mode, one has two dedicated compare channels that enable the generation of up to 4 PWM signals per Timer Slice (up to 16 PWM outputs per CCU8 unit), with dead time insertion to prevent short circuits in the switches. In Capture Mode a set of up to four capture registers is available.

Each CCU8 module has four service request lines that can be easily programmed to act as synchronized triggers between the PWM signal generation and an ADC conversion.

Straightforward timer slice concatenation is also possible, enabling up to 64 bit timing operations. This offers a flexible frequency measurement, frequency multiplication and pulse width modulation scheme.

A programmable function input selector for each timer slice, that offers up to nine functions, discards the need of complete resource mapping due to input ports availability.

Capture/Compare Unit 8 (CCU8)

All built-in link between the CCU8 and POSIF modules also enable a flexible digital motor control loop implementation, with direct coupling with Hall Sensors for Brushless DC Motor Control.

22.1.1 Features

CCU8 Module Features

Each CCU8 represents a combination of four Timer Slices, that can work independently in compare or capture mode. Each timer slice has 4 dedicated outputs for PWM signal generation.

All four CCU8 timer slices, CC8y, are identical in terms of available functions and operating modes. Avoiding this way the need of implementing different software routines, depending on which resource of CCU8 is used.

A built-in link between the four timer slices is also available, enabling this way a simplified timer concatenation and sequential operations.

General Features

- 16 bit timer cells
- programmable low pass filter for the inputs
- built-in timer concatenation
 - 32, 48 or 64 bit width
- shadow transfer for the period and compare channels
- four capture registers in capture mode
- programmable clock prescaler
- normal timer mode
- gated timer mode
- three counting schemes
 - center aligned
 - edge aligned
 - single shot
- PWM generation
- asymmetric PWM generation
- TRAP function
- dead time generation
- start/stop can be controlled by external events
- counting external events
- four dedicated service request lines per CCU8

Additional features

- external modulation function
- load controlled by external events

Capture/Compare Unit 8 (CCU8)

- dithering PWM
- floating point pre scaler
- output state override by an external event
- programmable output parity checker
- easy connection with POSIF unit for
 - hall sensor mode
 - rotary encoder mode
 - multi channel/multi phase control

CCU8 features vs. applications

On [Table 22-2](#) a summary of the major features of the CCU8 unit mapped with the most common applications.

Table 22-2 Applications summary

Feature	Applications
Four independent timer cells	Independent PWM generation: <ul style="list-style-type: none"> • Multiple buck/boost converter control (with independent frequencies) • Different mode of operation for each timer, increasing the resources optimization • Up to 2 H-Bridge control • multiple Zero Voltage Switch (ZVS) converter control with easy link to the ADC channels. • Multi Level Inverters
Two compare channels per Timer Slice	Linking between the two compare channels or linking between two Timer Slices: <ul style="list-style-type: none"> • Asymmetric PWM signal generation possibility decreases the number of current sensors • Linking between timer slices enable Phase Shift Full Bridge topologies control • Linking between slices enable N-Phase DC/DC converter control

Table 22-2 Applications summary (cont'd)

Feature	Applications
Two Dead Time Generators	Independent dead time values for rising and falling transitions and independent channel dead time counter: <ul style="list-style-type: none"> • Each channel can work stand alone with different dead time values. This enables the control of up to 2 Half-Bridges with different dead time values and the same frequency • Different dead time values for rising and falling transitions can be used to optimize the switching activity of the MOSFETs
Concatenated timer cells	Easy to configure timer extension up to 64 bit: <ul style="list-style-type: none"> • High dynamic trigger capturing • High dynamic signal measurement
Dithering PWM	Generating a fractional PWM frequency or duty cycle: <ul style="list-style-type: none"> • To avoid big steps on frequency or duty cycle adjustment in slow control loop applications • Increase the PWM signal resolution over time
Floating prescaler	Automated control signal measurement: <ul style="list-style-type: none"> • decrease SW activity for monitoring signals with high or unknown dynamics • generating a more than 16 bit timer for system control
Up to 9 functions via external signals for each timer	Flexible resource optimization: <ul style="list-style-type: none"> • The complete set of external functions is always available • Several arrangements can be done inside a CCU8, e.g., one timer working in capture mode and one working in compare
Output Parity Checker	Automated Mosfet signal monitoring: <ul style="list-style-type: none"> • parity checker can be used to monitor the output of the IGBTs and comparing them against the complete set of PWM outputs of CCU8. • Avoiding short circuits in a multi Mosfet system.

Table 22-2 Applications summary (cont'd)

Feature	Applications
4 dedicated service request lines	Specially developed for: <ul style="list-style-type: none"> • generating interrupts for the microprocessor • flexible connectivity between peripherals, e.g., ADC triggering.
Linking with POSIF	Flexible profiles for: <ul style="list-style-type: none"> • Rotary Encoder connection • Hall Sensor • Modulating the 4 timer outputs via SW

22.1.2 Block Diagram

Each CCU8 timer slice can operate independently from the other slices for all the available modes. Each timer slice contains a dedicated input selector for functions linked with external events and has 4 dedicated compare output signals, for PWM signal generation.

The built-in timer concatenation is only possible with adjacent slices, e.g. CC80/CC81. Combinations for slice concatenations like, CC80/CC82 or CC80/CC83 are not possible.

The individual service requests for each timer slice (four per slice) are multiplexed into four module service requests lines, [Figure 22-1](#).

Capture/Compare Unit 8 (CCU8)

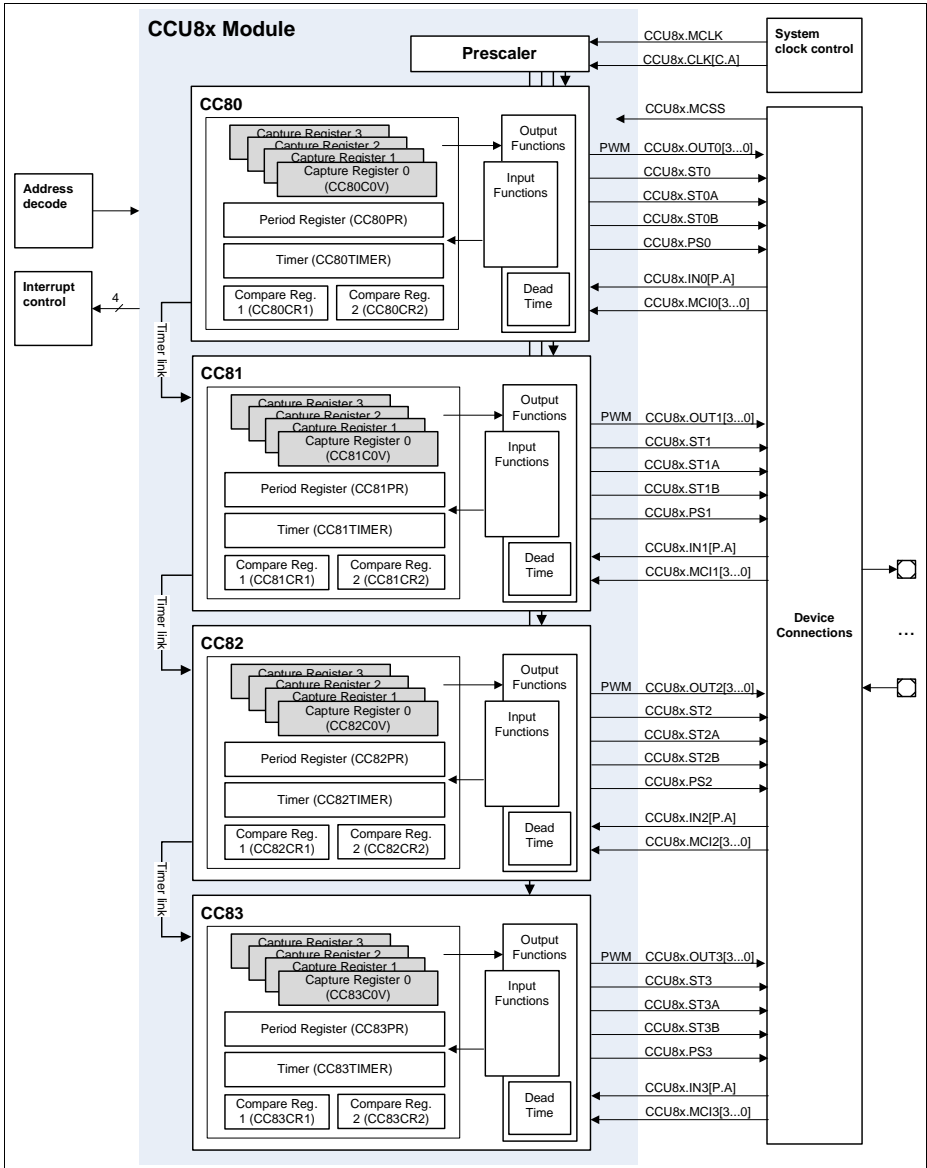


Figure 22-1 CCU8 block diagram

22.2 Functional Description

In each figure several registers may be depicted to indicate controlability or configurability. These registers follow the description given in [Figure 22-2](#). One should also note that indexing in a register can be done via the non capital y, x or n.

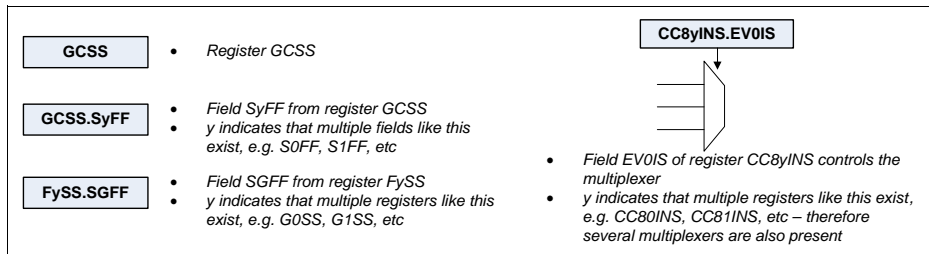


Figure 22-2 Register description in figures (example)

22.2.1 Overview

The input path of a CCU8 slice is comprised of a selector ([Section 22.2.2](#)) and a connection matrix unit ([Section 22.2.3](#)). The output path contains a service request control unit, a timer concatenation unit and two units that control directly the state of the output signal for each specific slice (for TRAP, dead time generation and modulation handling), see [Figure 22-3](#).

Capture/Compare Unit 8 (CCU8)

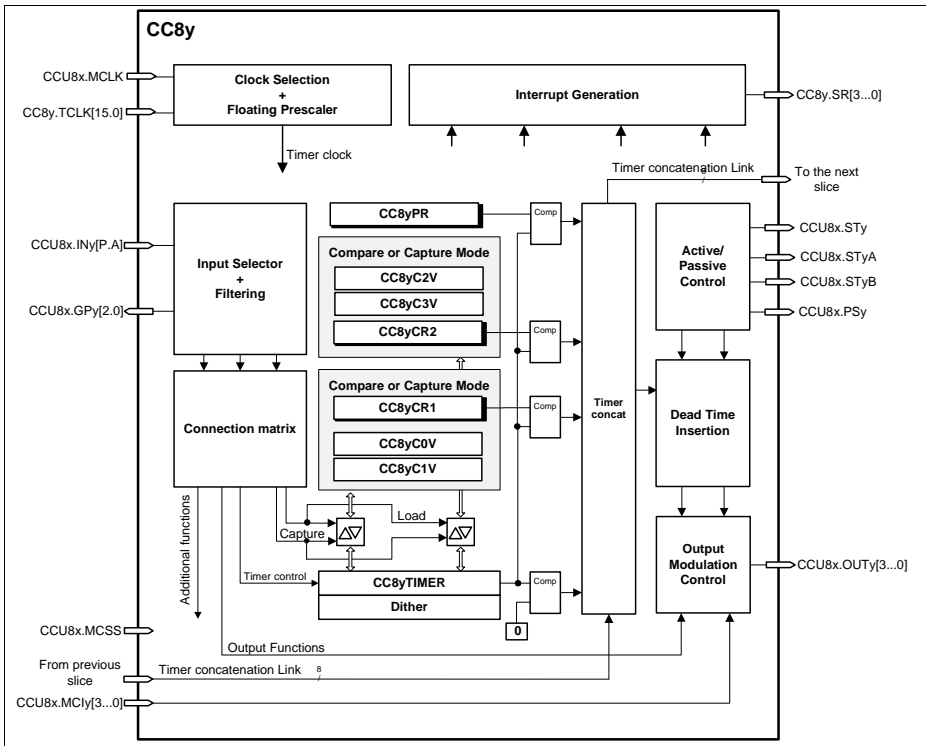


Figure 22-3 CCU8 slice block diagram

The timer core is built of 16 bit counter and one period register and two compare channels in compare mode, or four capture channels plus the period register in capture mode.

Individual timer clocks can be selected for each and every Timer Slice, enabling a very flexible resource organization inside each CCU8 module.

In compare mode the period sets the maximum counting value while the two compare registers are used to control the ACTIVE/PASSIVE state of the four dedicated comparison slice outputs.

Each CCU8 slice contains a dedicated timer link interface that is used to perform timer concatenation, up to 64 bits. This timer concatenation is controlled via a single bit field configuration.

Table 22-3 describes the inputs and outputs for each CCU8 Timer Slice.

Capture/Compare Unit 8 (CCU8)

Inputs and outputs that are not seen at the CCU8 module boundaries have a nomenclature of CC8y.<name>, whilst CCU8 module inputs and outputs are described as CCU8x.<signal_name>y (indicating the variable y the object slice).

Table 22-3 CCU8 slice pin description

Pin	I/O	Description
CCU8x.MCLK	I	Module clock
CC8y.TCLK		Clock from the pre scaler
CCU8x.INy[P:A]	I	Slice functional inputs (used to control the functionality throughout slice external events)
CCU8x.MCIy[3...0]	I	Multi Channel mode inputs
CCU8x.MCSS	I	Multi Channel shadow transfer trigger
CC8y.SR[3...0]	O	Slice service request lines
CCU8x.GPy[2...0]	O	Signals decoded from the input selector (used for the parity checker function)
CCU8x.STy	O	This signal can be the slice comparison status value of channel 1, channel 2 or a AND between both
CCU8x.STyA	O	Slice comparison status value of channel 1
CCU8x.STyB	O	Slice comparison status value of channel 2
CCU8x.PSy	O	Period match
CCU8x.OUTy[3...0]	O	Slice dedicated output pins

Note:

6. *The status bit outputs of the Kernel, CCU8x.STy, CCU8x.STyA and CCU8x.STyB are extended for one more kernel clock cycle.*
7. *The Service Request signals at the output of the kernel are extended for one more kernel clock cycle.*
8. *The maximum output signal frequency of the CCU8x.STy, CCU8x.STyA and CCU8x.STyA is module clock divided by 4.*

The slice timer, can count up or down depending on the selected operating mode. A direction flag contains the actual counting direction.

The timer is connected to three stand alone comparators, one for the period match and two for the compare match of each compare channel. The registers used for comparison match of both compare channels, can be programmed to serve as capture registers, enabling sequential capture capabilities on external events.

Capture/Compare Unit 8 (CCU8)

In normal edge aligned counting scheme, the counter is cleared to 0000_H each time it matches the period value defined in the period register. In center aligned mode, the counter direction changes from 'up counting' to 'down counting' after reaching the period value. Both period and compare registers have an aggregated shadow register, which enables the update of the PWM period and duty cycle on the fly.

A single shot mode is also available, where the counter stops after it reaches the value set in the period register.

The start and stop of the counter can be set/clear by software access or by a programmable input pin.

The dead time generator can be programmed with different values for the rising and falling edge of the output.

Functions like, load, counting direction (up/down), TRAP, output modulation can also be controlled with external events, see [Section 22.2.3](#).

22.2.2 Input Selector

The first unit of the slice input path, is used to select from which are used to control the available external functions.

Inside this block the user also has the possibility to perform a low pass filtering of the signals and selecting the active edge(s) or level of the external event, see [Figure 22-4](#).

The user has the possibility of selecting any of the CCU8x.INy[P:A] inputs has the source of an event.

At the output of this unit we have a user selection of three events, that were configured to be active at rising, falling or both edges, or level active. These selected events can then be mapped to several functions.

Notice that each decoded event contains two outputs, one edge active and one level active, due to the fact that some functions like counting, capture or load are edge sensitive events while, timer gating or up down counting selection are level active.

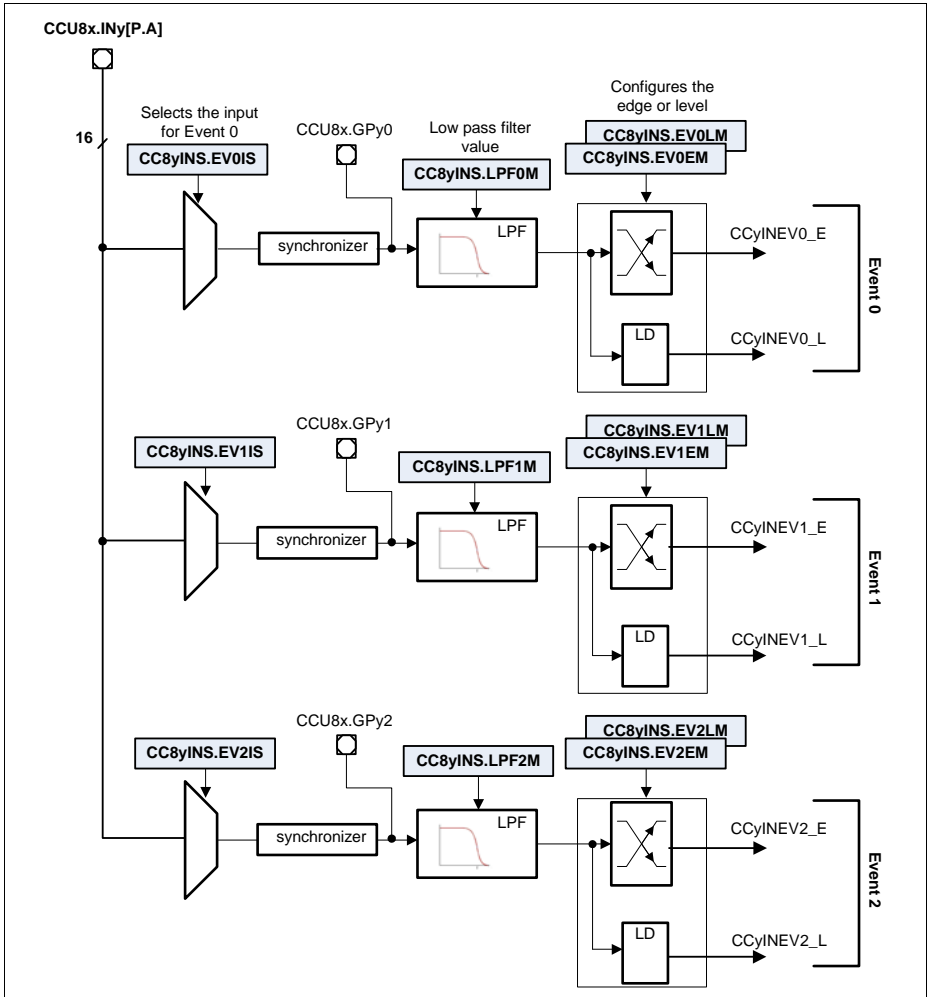


Figure 22-4 Slice input selector diagram

22.2.3 Connection Matrix

The connection matrix maps the events coming from the input selector to several user configured functions, [Figure 22-5](#). The following functions can be enabled on the connection matrix:

Table 22-4 Connection matrix available functions

Function	Brief description	Map to figure Figure 22-5
Start	Edge signal to start the timer	CCystrt
Stop	Edge signal to stop the timer	CCystp
Count	Edge signal used for counting events	CCycnt
Up/down	Level signal used to select up or down counting direction	CCyupd
Capture 0	Edge signal that triggers a capture into the capture registers 0 and 1	CCycapt0
Capture 1	Edge signal that triggers a capture into the capture registers 2 and 3	CCycapt1
Gate	Level signal used to gate the timer clock	CCygate
Load	Edge signal that loads the timer with the value present at the compare register	CCyload
TRAP	Level signal used for fail-safe operation	CCytrap
Modulation	Level signal used to modulate/clear the output	CCymod
Status bit override	Status bit is going to be overridden with an input value	CCyoval for the value CCyoset for the trigger

Inside the connection matrix we also have a unit that performs the built-in timer concatenation. This concatenation enables a completely synchronized operation between the concatenated slices for timing operations and also for capture and load actions. The timer slice concatenation is done via the **CC8yCMC.TCE** bitfield. For a complete description of the concatenation function, please address [Section 22.2.10](#).

Capture/Compare Unit 8 (CCU8)

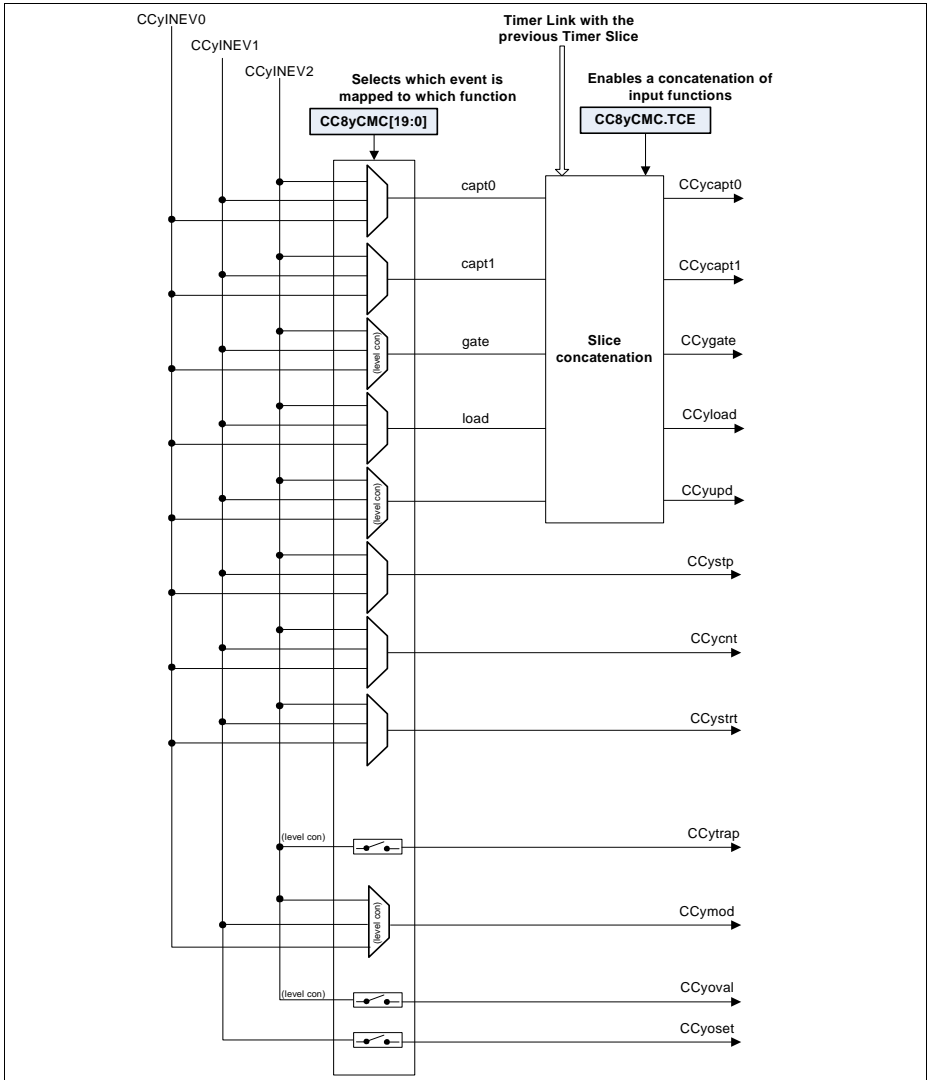


Figure 22-5 Slice connection matrix diagram

22.2.4 Start/Stop Control

Each slice contains a run bit register, that indicates the actual status of the timer, **CC8yTCST.TRB**. The start and stop of the timer can be done by software access or can be controlled directly by external events, see [Figure 22-6](#).

Selecting an external signal that acts as a start trigger does not force the user to use an external stop trigger and vice versa.

Selecting the single shot mode, imposes that after the counter reaches the period value the run bit, **CC8yTCST.TRB**, is going to be cleared and therefore the timer is stopped.

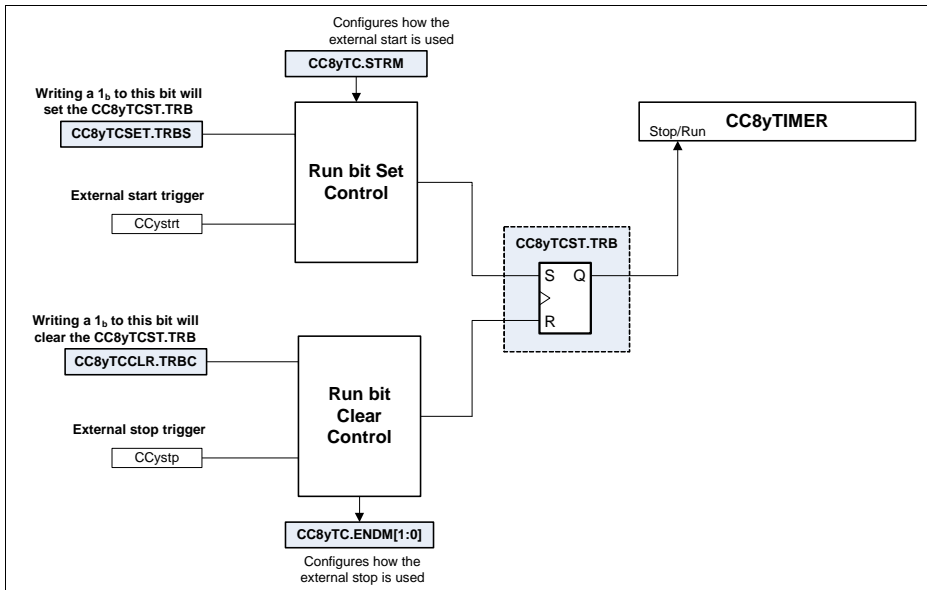


Figure 22-6 Timer start/stop control diagram

One can use the external stop signal to perform the following functions (configuration via **CC8yTC.ENDM**):

- Clear the run bit (stops the timer) - default
- Clear the timer (to 0000_H) but it does not clear the run bit (timer still running)
- Clear the timer and the run bit

One can use the external start to perform the following functions (configuration via **CC8yTC.STRM**):

- Start the timer (resume operation)
- Clear and starts the timer

Capture/Compare Unit 8 (CCU8)

The set (start the timer) of the timer run bit, always has priority over a clear (stop the timer).

To start multiple CCU8 timers at the same time/synchronously one should use a dedicated input as external start (see [Section 22.2.8.1](#) for a description how to configure an input as start function). This input should be connected to all the Timers that need to be started synchronously (see [Section 22.8](#) for a complete list of module connections), [Figure 22-7](#).

For starting the timers synchronously via software there is a dedicated input signal, controlled by the SCU (System Control Unit), that is connected to all the CCU8 timers. This signal should then be configured as an external start signal (see [Section 22.2.8.1](#)) and then the software must write $1_B/0_B$ (depending on the external start function configuration) to the specific bitfield of the CCUCON register (this register is described on the SCU chapter).

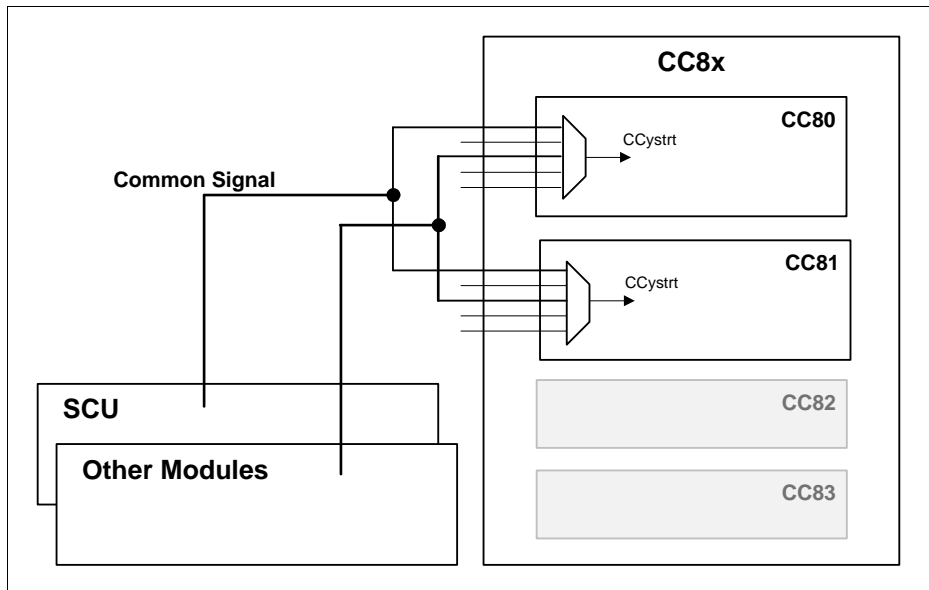


Figure 22-7 Start multiple timers synchronously

22.2.5 Counting Modes

Each CC8y timer can be programmed into three different counting schemes:

- Edge aligned (default)
- Center aligned
- Single shot (edge or center aligned)

Capture/Compare Unit 8 (CCU8)

These three counting schemes can be used as stand alone without the need of selecting any inputs as external event sources. Nevertheless it is also possible to control the counting operation via external events like, timer gating, counting trigger, external stop, external start, etc.

For all the counting modes, it is possible to update on the fly the values for the timer period and compare channel. This enables a cycle by cycle update of the PWM frequency and duty cycle.

Each compare channel of the CC8y Timer Slice has an associated Status Bit (**GCST.CC8yST1** for compare channel 1 and **GCST.CC8yST2** for compare channel 2), that indicates the active or passive state of the channel, **Figure 22-8**. The set and clear of the status bits and the respective PWM signal generation is dictated by the timer period, compare value and the current counting mode. See the different counting mode descriptions, **Section 22.2.5.3** to **Section 22.2.5.5** to understand how these bits are set and cleared.

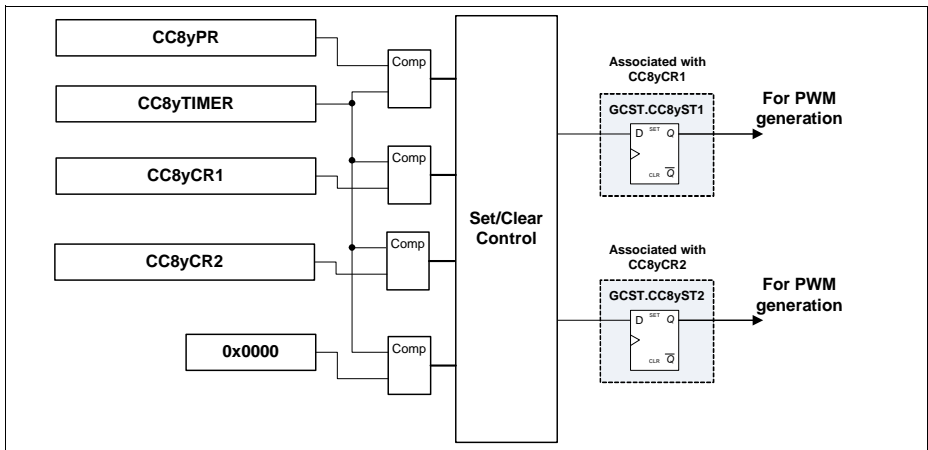


Figure 22-8 CC8y Status Bits

22.2.5.1 Calculating the PWM Period and Duty Cycle

The period of the timer is determined by the value in the period register, **CC8yPR** and by the timer mode.

The base for the PWM signal frequency and duty cycle, is always related to the clock frequency of the timer itself and not to the frequency of the module clock (due to the fact that the timer clock can be a scaled version of the module clock).

In Edge Aligned Mode, the timer period is:

$$T_{per} = \langle \text{Period-Value} \rangle + 1; \text{ in } f_{tclk} \tag{22.1}$$

In Center Aligned Mode, the timer period is:

$$T_{per} = (<Period-Value> + 1) \times 2; \text{ in } f_{clk} \quad (22.2)$$

For each of these counting schemes, the duty cycle of generated PWM signal is dictated by the value programmed into the compare channel registers, **CC8yCR1** and **CC8yCR2**. Notice that one can have different duty cycle values for each of the compare channels.

In Edge Aligned and Center Aligned Mode, the PWM duty cycle is:

$$DC = 1 - <Compare-Value>/(<Period-Value> + 1) \quad (22.3)$$

Both the period and compare registers, **CC8yPR**, **CC8yCR1** and **CC8yCR2** respectively, can be updated on the fly via software enabling a glitch free transition between different period and duty cycle values for the generated PWM signal, [Section 22.2.5.2](#)

22.2.5.2 Updating the Period and Duty Cycle

Standard Shadow Transfer Control

Each CCU8 timer slice provides an associated shadow register for the period and the two compare values. This facilitates a concurrent update by software for these three parameters, with the objective of modifying during run time the PWM signal period and duty cycle.

In addition to the shadow registers for the period and compare values, one also has available shadow registers for the floating prescaler, dither and passive level, **CC8yFPCS**, **CC8yDITS** and **CC8yPSL** respectively (please address [Section 22.2.13](#) and [Section 22.2.12](#) for a complete description of these functions).

The structure of the shadow registers can be seen in [Figure 22-9](#).

Capture/Compare Unit 8 (CCU8)

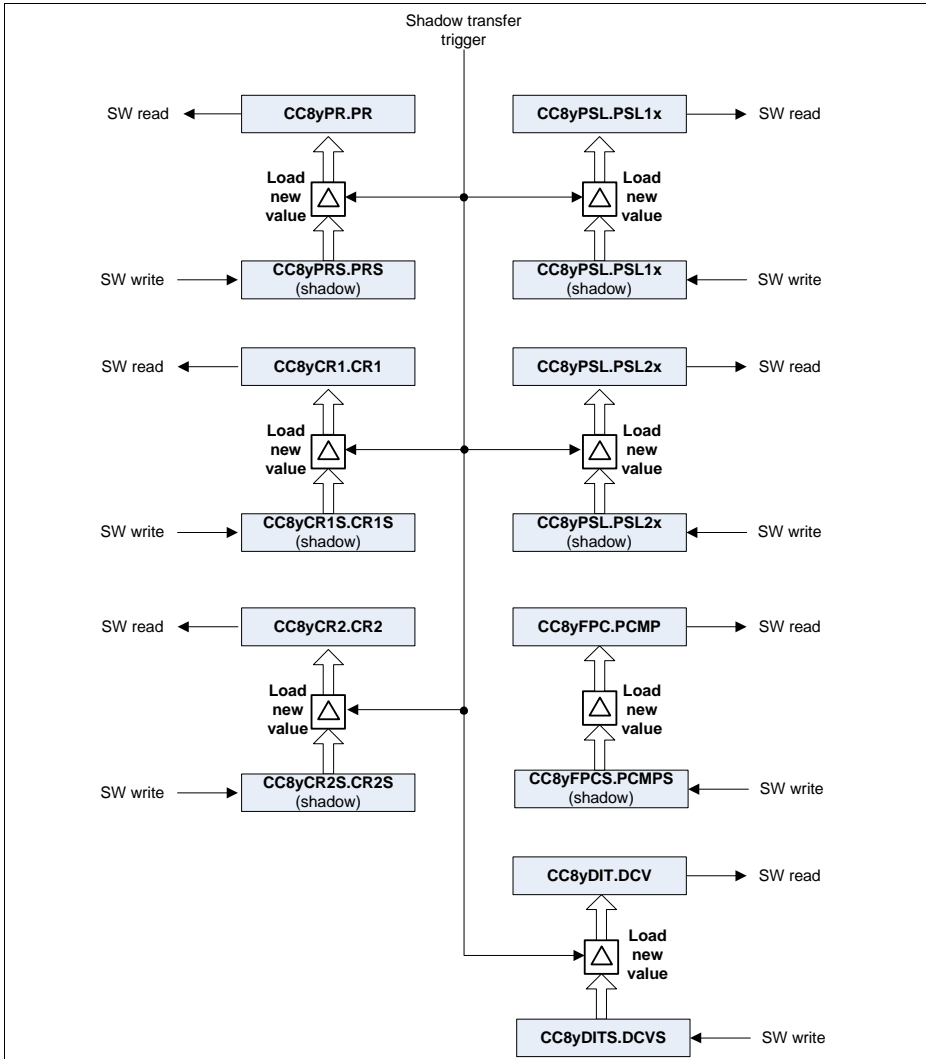


Figure 22-9 Shadow registers overview

The update of these registers can only be done by writing a new value into the associated shadow register and wait for a shadow transfer to occur.

Each group of shadow registers have an individual shadow transfer enable bit, [Figure 22-10](#). The software must set this enable bit to 1_B, whenever an update of the

Capture/Compare Unit 8 (CCU8)

values is needed. These bits are automatically cleared by the hardware, whenever an update of the values is finished. Therefore every time that an update of the registers is needed the software must set again the specific bit(s).

Nevertheless it is also possible to clear the enable bit via software. This can be used in the case that an update of the values needs to be cancelled (after the enable bit has already been set).

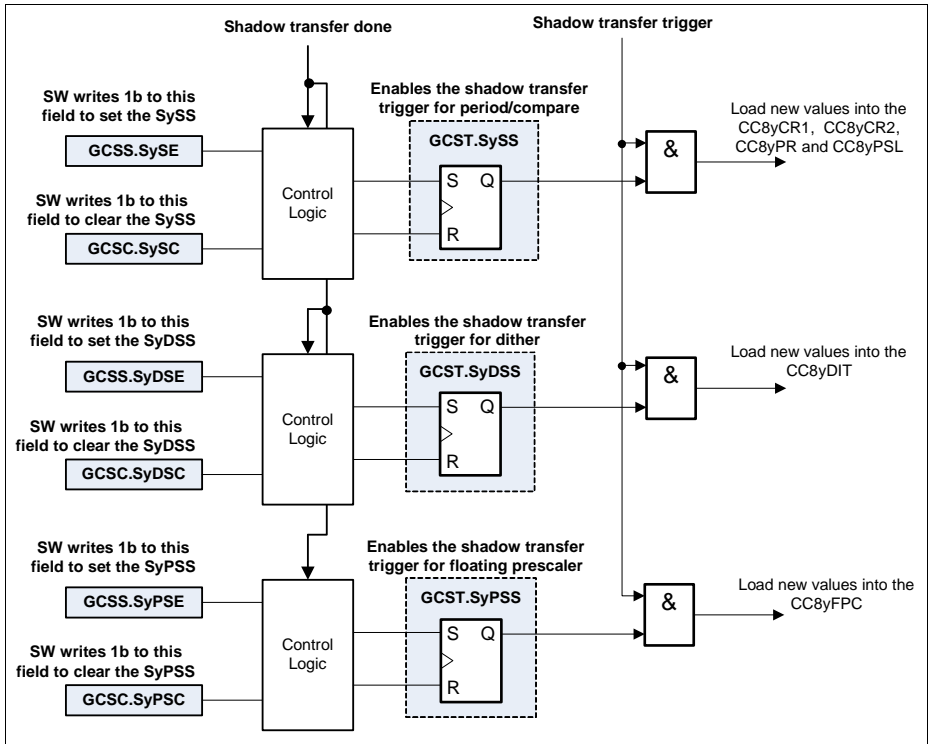


Figure 22-10 Shadow transfer enable logic

The shadow transfer operation is going to be done in the immediately next occurrence of a shadow transfer trigger, after the shadow transfer enable is set (**GCST.SySS**, **GCST.SyDSS**, **GCST.SyPSS** set to 1_B).

The occurrence of the shadow transfer trigger is imposed by the timer counting scheme (edge aligned or center aligned). Therefore the slots when the values are updated can be:

- in the next clock cycle after a Period Match while counting up
- in the next clock cycle after an One Match while counting down

Capture/Compare Unit 8 (CCU8)

- immediately, if the timer is stopped and the shadow transfer enable bit(s) is set

Figure 22-11 shows an example of the shadow transfer control when the timer slice has been configured into center aligned mode. For a complete description of all the timer slice counting modes, please address [Section 22.2.5.3](#), [Section 22.2.5.4](#) and [Section 22.2.5.5](#).

It is also possible to control in which slot the shadow transfer is done, via the STM field. This is only valid in Center Aligned Mode:

- **CC8ySTC.STM** = 00_B (default) - Shadow transfer is done at the Period Match and One match slot
- **CC8ySTC.STM** = 01_B - Shadow transfer is done only at the Period Match slot
- **CC8ySTC.STM** = 10_B - Shadow transfer is done only at the One Match slot

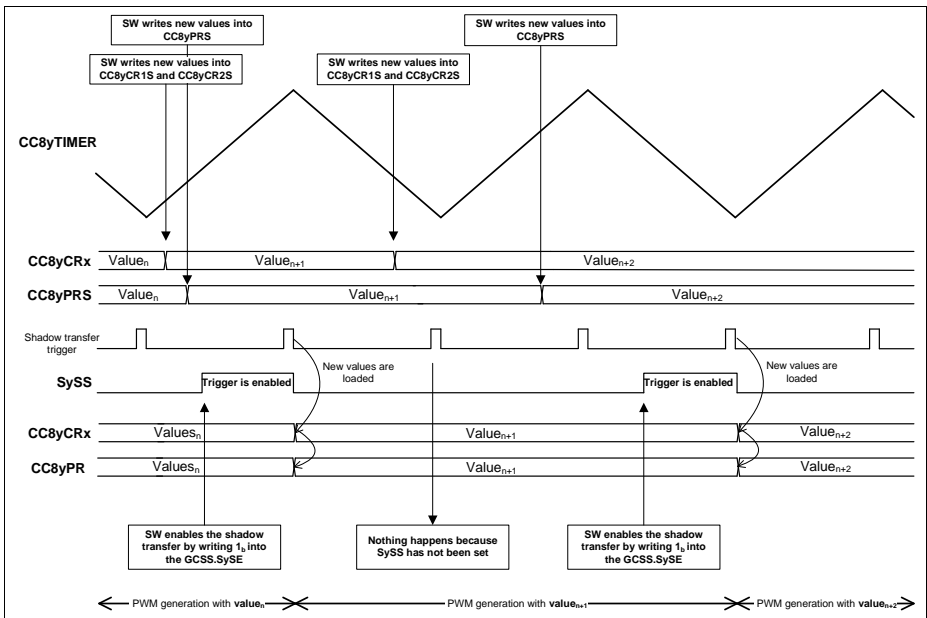


Figure 22-11 Shadow transfer timing example - center aligned mode

When using the CCU8 in conjunction with the POSIF to control the multi channel mode, it can be necessary in some cases, to perform the shadow transfers synchronously with the update of the multi channel pattern. To perform this action, each CCU8 contains a dedicated input that can be used to synchronize the two events, the CCU8x.MCSS.

This input, when enabled, is used to set the shadow transfer enable bitfields (**GCST.SySS**, **GCST.SyDSS** and **GCST.SyPSS**) of the specific slice. It is possible to select which slice is using this input to perform the synchronization via the **GCTRL.MSEy**

Capture/Compare Unit 8 (CCU8)

bit field. It is also possible to enable the usage of this signal for the three different shadow transfer signals: compare and period values, dither compare value and prescaler compare value. This can be configured on the **GCTRL.MSDE** field.

The structure for using the CCU8x.MCSSS input signal can be seen in **Figure 22-12**. The usage of this signal is just an add on to the shadow transfer control and therefore all the previous described functions are still available.

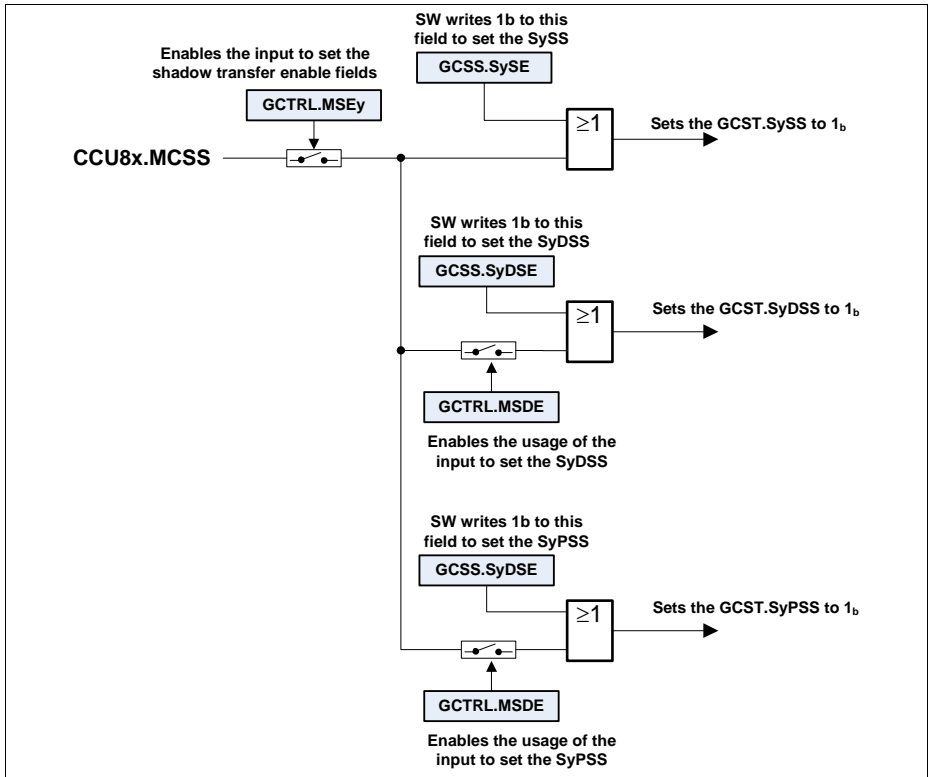


Figure 22-12 Usage of the CCU8x.MCSSS input

Cascaded Shadow Transfer

It is possible to cascade the shadow transfer operation throughout the CCU8 timer slices. The specific shadow transfer of a timer slice is cascaded with the adjacent timer slices, **Figure 22-13**.

To enable the cascaded shadow transfer function, the bitfield **CC8ySTC.CSE** of the specific timer slice needs to be set to 1_B.

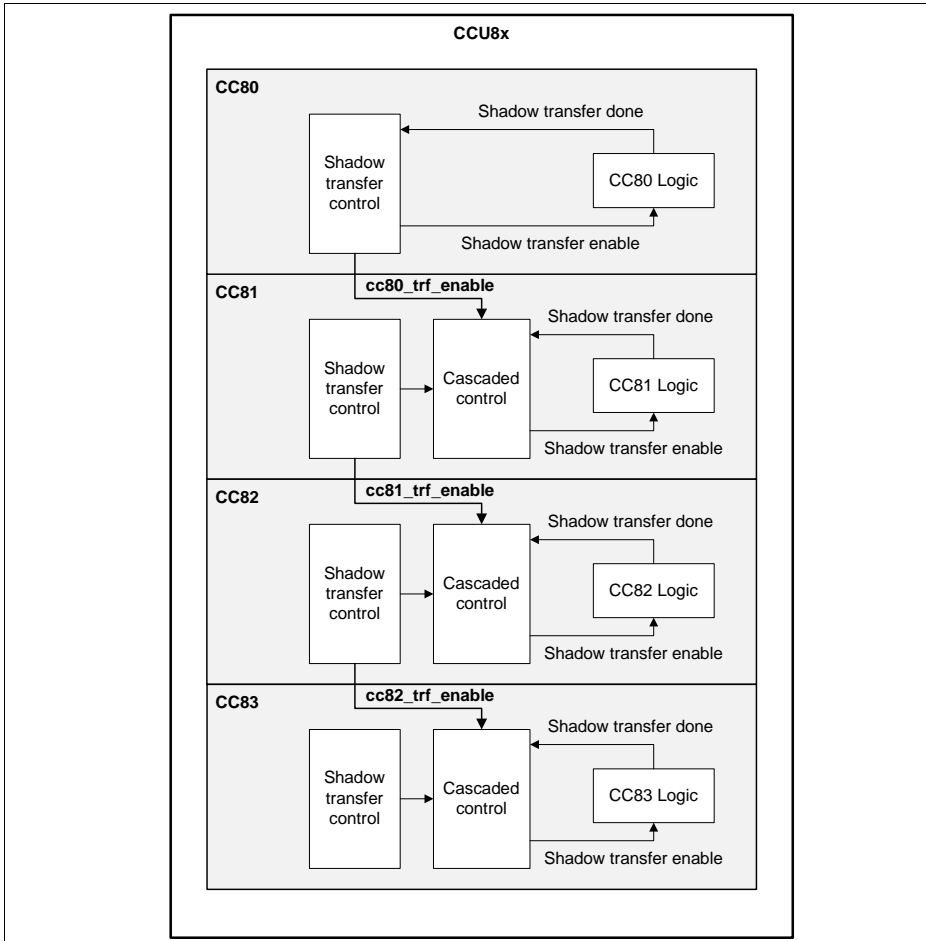


Figure 22-13 Cascaded shadow transfer linking

The shadow transfer enable bits, still need to be set via SW for each of the individual slices, [Figure 22-14](#).

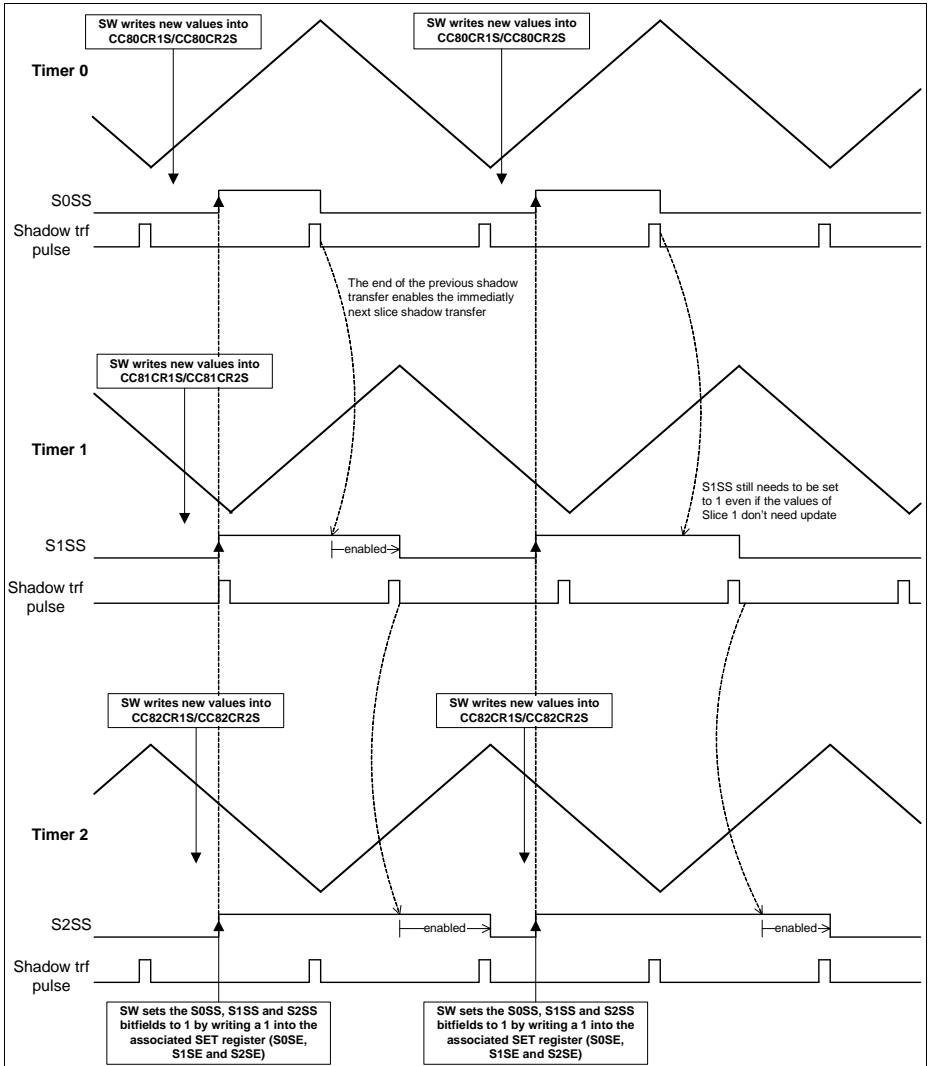


Figure 22-14 Cascade shadow transfer timing

Note: The shadow transfer enable bits, need to be set in all timer slices that are being used in the cascaded architecture, at the same time. The shadow transfer enable bits, also need to be set for all slices even if the shadow values of some slices were not updated.

22.2.5.3 Edge Aligned Mode

Edge aligned mode is the default counting scheme. In this mode, the timer is incremented until it matches the value programmed in the period register, **CC8yPR**. When period match is detected the timer is cleared to 0000_H and continues to be incremented.

In this mode, the value of the period register and compare registers are updated with the values written by software into the correspondent shadow register, every time that an overflow occurs (period match), see **Figure 22-15**.

In edge aligned mode, the status bit of the comparison (**CC8ySTx**) is set one clock cycle after the timer hits the value programmed into the compare register. The clear of the status bit is done one clock cycle after the timer reaches 0000_H .

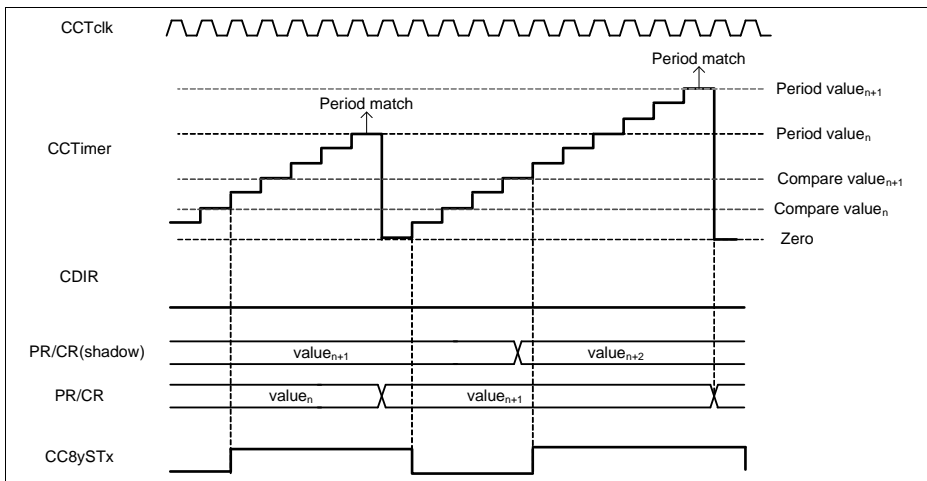


Figure 22-15 Edge aligned mode, **CC8yTC.TCM = 0_B**

22.2.5.4 Center Aligned Mode

In center aligned mode, the timer is counting up or down with respect to the following rules:

- The counter counts up while **CC8yTCST.CDIR = 0_B** and it counts down while **CC8yTCST.CDIR = 1_B**.
- Within the next clock cycle, the count direction is set to counting up (**CC8yTCST.CDIR = 0_B**) when the counter reaches 0001_H while counting down.
- Within the next clock cycle, the count direction is set to counting down (**CC8yTCST.CDIR = 1_B**), when the period match is detected while counting up.

Capture/Compare Unit 8 (CCU8)

The status bit (CC8ySTx) is always 1_B when the counter value is equal or greater than the compare value and 0_B otherwise.

While in edge aligned mode, the shadow transfer for compare and period registers is executed once per period. It is executed twice in center aligned mode as follows

- Within the next clock cycle after the counter reaches the period value, while counting up (**CC8yTCST.CDIR = 0_B**).
- Within the next clock cycle after the counter reaches 0001_H, while counting down (**CC8yTCST.CDIR = 1_B**).

*Note: Bit **CC8yTCST.CDIR** changes within the next timer clock after the one-match or the period-match, which means that the timer continues counting in the previous direction for one more cycle before changing the direction.*

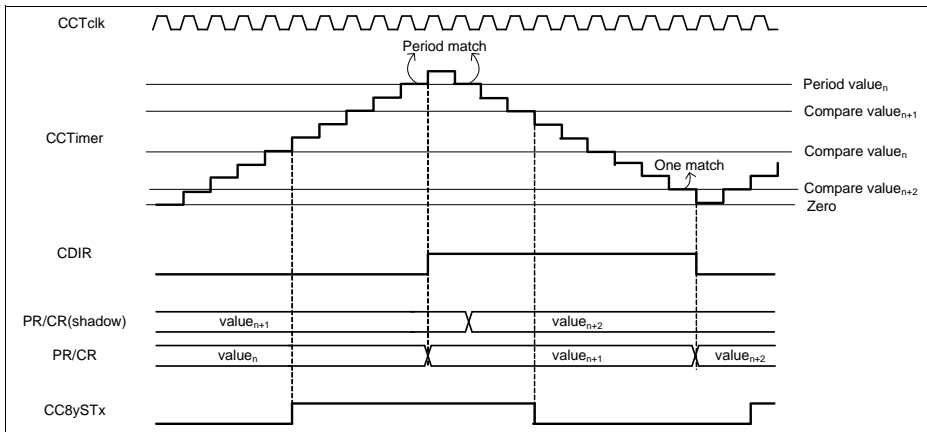


Figure 22-16 Center aligned mode, **CC8yTC.TCM = 1_B**

22.2.5.5 Single Shot Mode

In single shot mode, the timer is stopped after the current timer period is finished. This mode can be used with a center or edge aligned scheme.

In edge aligned mode, **Figure 22-17**, the timer is stopped when it is cleared to 0000_H after having reached the period value. In center aligned mode, **Figure 22-18**, the period is finished when the timer has counted down to 0000_H.

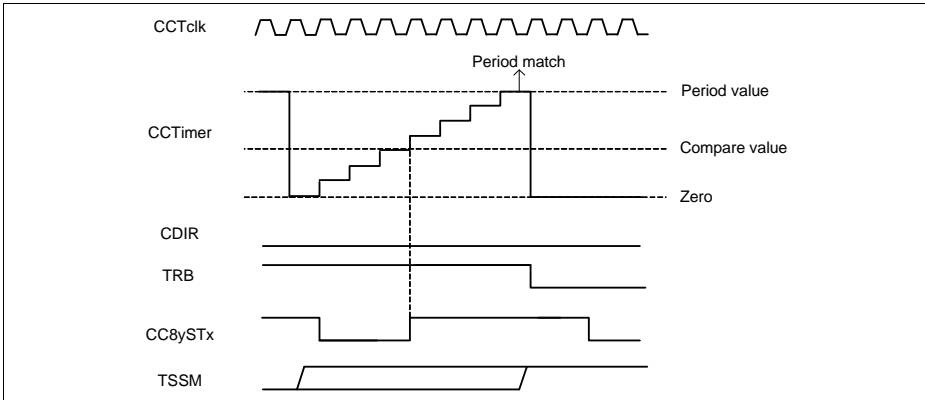


Figure 22-17 Single shot edge aligned - $CC8yTC.TSSM = 1_B$, $CC8yTC.TCM = 0_B$

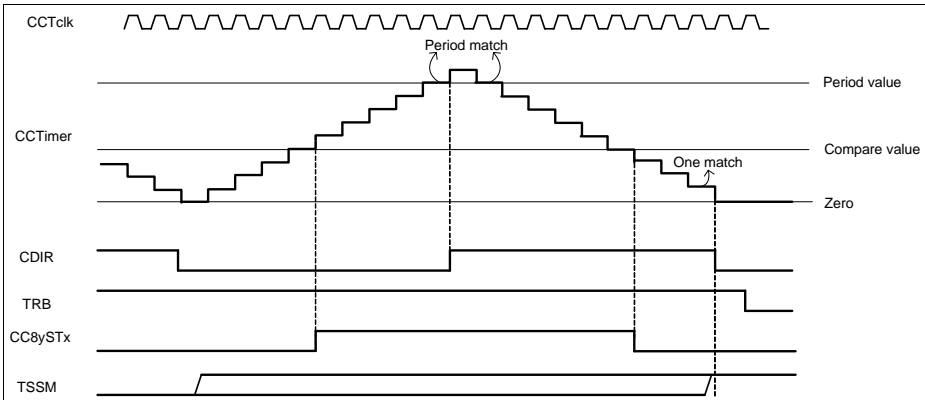


Figure 22-18 Single shot center aligned - $CC8yTC.TSSM = 1_B$, $CC8yTC.TCM = 1_B$

22.2.6 Active/Passive Rules

The general rules that set or clear the associated timer slice status bit, can be generalized independently of the timer counting mode.

The following events set the Status bit to Active:

- in the next f_{tclk} cycle after a compare match while counting up
- in the next f_{tclk} cycle after a zero match while counting down

The following events set the Status bit to Inactive:

- in the next f_{tclk} cycle after a zero match (and not compare match) while counting up
- in the next f_{tclk} cycle after a compare match while counting down

Capture/Compare Unit 8 (CCU8)

If external events are being used to control the timer operation, these rules are still applicable.

The status bit state can only be 'override' via software or by the external status bit override function, [Section 22.2.8.10](#).

The software at any time can write a 1_B into the **GCSS.SySTS** bitfield, which will set the status bit **GCST.CC8yST** of the specific timer slice. By writing a 1_B into the **GCSC.SySTC** bitfield, the software imposes a clear of the specific status bit.

22.2.7 Compare Modes

Compare Channel Scheme

Each CCU8 slice has two compare channels and two dead time generators, one for each channel, see [Figure 22-19](#). Each compare uses the information of the status bit, CC8ySTx, to generate two complementary outputs. All the outputs, CCU8x.OUTy0, CCU8x.OUTy1, CCU8x.OUTy2 and CCU8x.OUTy3, have a dedicated passive level control bit.

Each compare channel can work in an individual manner for both edge and center aligned modes. This means that two different complementary PWM signals can be generated by using the available compare channels. The PWM frequency is the same for both channels, but the duty cycle can be programmed independently for each channel.

It is also possible to select an asymmetric output scheme, by setting the field **CC8yCHC.ASE** = 1_B. In the asymmetric mode, the compare channels are grouped together to generate a single complementary PWM signal at the CCU8x.OUTy0 and CCU8x.OUTy1 pins.

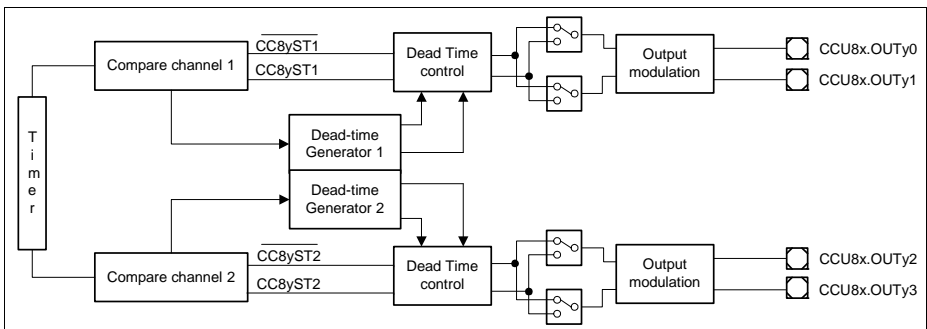


Figure 22-19 Compare channels diagram

Dead Time Generator

In most cases the switching behavior regarding the switch-on and switch-off times is not symmetrical, which can lead to a short circuit if the switch-on time is smaller than the switch-off time. To overcome this problem, each Timer Slice channel contains a dead time generator, which is able to delay the switching edges of the output signals, **Figure 22-20**.

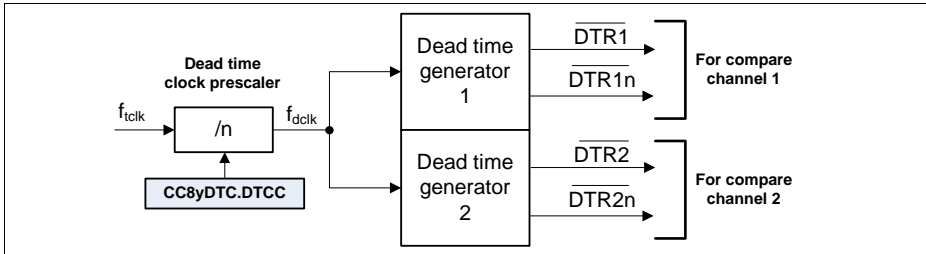


Figure 22-20 Dead Time scheme

Each dead time generator contains an eight bit counter with a different programmable reload value for rise and fall times. The dead time generators contain a programmable prescaler for the dead time counter clock, to enable large dead time insertion values, **Table 22-5**.

Table 22-5 Dead time prescaler values

CC8yDTC.DTCC[1:0]	Frequency
00 _B	f_{tclk}
01 _B	$f_{tclk}/2$
10 _B	$f_{tclk}/4$
11 _B	$f_{tclk}/8$

Any transition on the associated status bits, CC8ySTx, will trigger the start of the specific dead time generator, **Figure 22-21**.

When a SET (CC8ySTx passes from 0_B to 1_B) action for the CC8ySTx bit is detected, the dead time counter is reloaded with the value present on the **CC8yDC1R.DT1R** or **CC8yDC2R.DT2R** (depending on which channel we are addressing).

When a CLEAR action for the CC8ySTx bit is detected (CC8ySTx passes from 0_B to 1_B), the dead time counter is reloaded with the value present on the **CC8yDC1R.DT1F** or **CC8yDC2R.DT2F** (depending on which channel we are addressing).

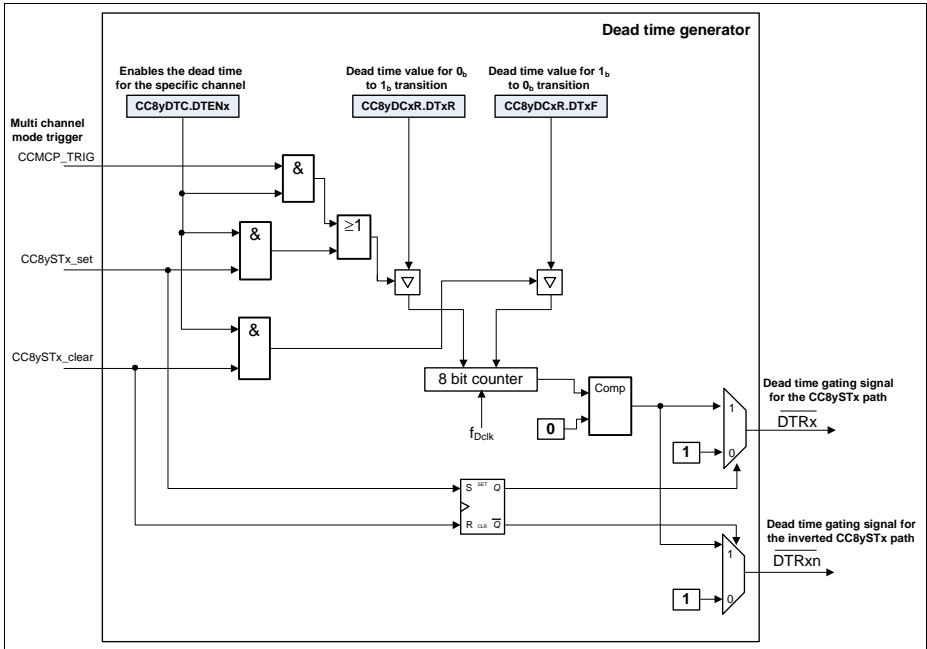


Figure 22-21 Dead Time generator scheme

Each dead time generator outputs two signals that are used to control the two complementary outputs (the CC8ySTx and the inverted CC8ySTx). The separation of the control signals enable a flexible enable/disable scheme inside of each compare channel, **Figure 22-22**. This means that the dead time generator can be enabled for one compare channel, but the dead time insertion can be discarded for one of the outputs.

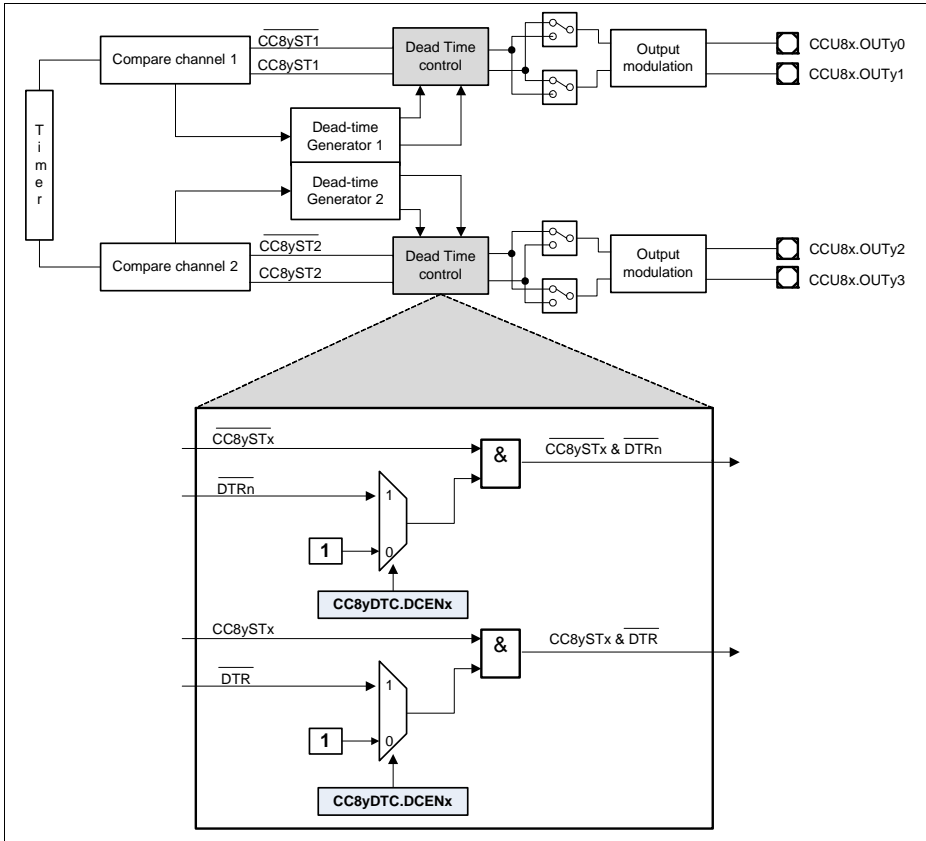


Figure 22-22 Dead Time control cell

When using the Multi Channel mode, $CC8yTC.MCMEy = 1_B$, there can be the scenario where the generated PWM signal has 100% duty cycle. This means that the respective status bit is always set and it is the Multi Channel pattern that is controlling the output modulation. In this case, we can have a transition from Inactive to Active state at the output, without having a transition on the specific status bit, creating a short on the switches due to the non existence of dead time insertion.

To overcome this possible short on the switches, a trigger from the multi channel control, CCMP_TRIG on [Figure 22-21](#), is fed to the dead time generators. [Figure 22-23](#) shows the scheme for the generation of the CCMP_TRIG, where the signals, CCMCMx0 and CCMCMx1 represent the sampled multi channel pattern for a specific channel.

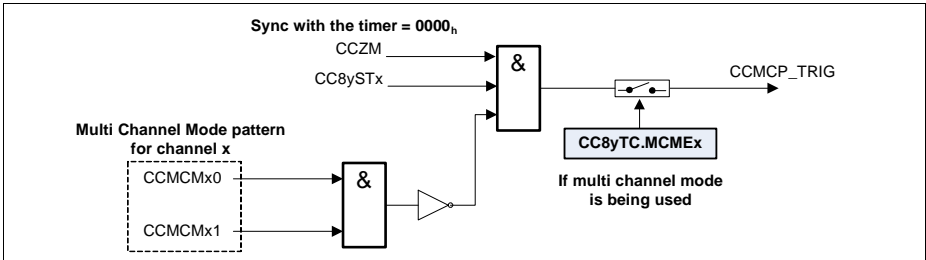


Figure 22-23 Dead Time trigger with the Multi Channel pattern

22.2.7.1 Edge Aligned Compare Modes

Standard Edge Aligned Mode

When the Timer Slice is programmed in edge aligned mode, the two channels can work independently, which means that the compare values can be programmed with different values (originating different duty cycles). In this scenario, each channel can output a pair of PWM signals used to control a high and low side switches, see [Figure 22-24](#).

In this mode, for each channel the dead time for rise and fall transitions are controlled by the values programmed in the [CC8yDC1R.DT1R](#) and [CC8yDC2R.DT2R](#), and [CC8yDC1R.DT1F](#) and [CC8yDC2R.DT2F](#) fields, respectively.

[Figure 22-25](#) shows the timing diagrams for a specific slice when the compare values of each channel are different.

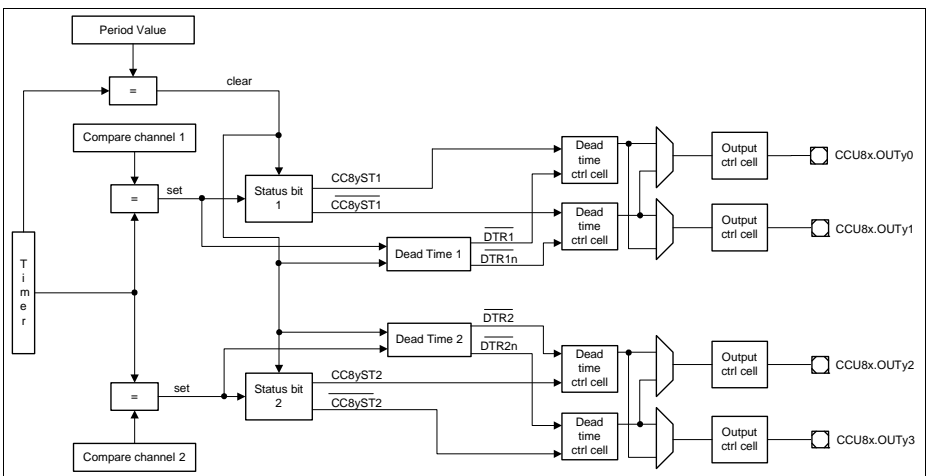


Figure 22-24 Edge Aligned with two independent channels scheme

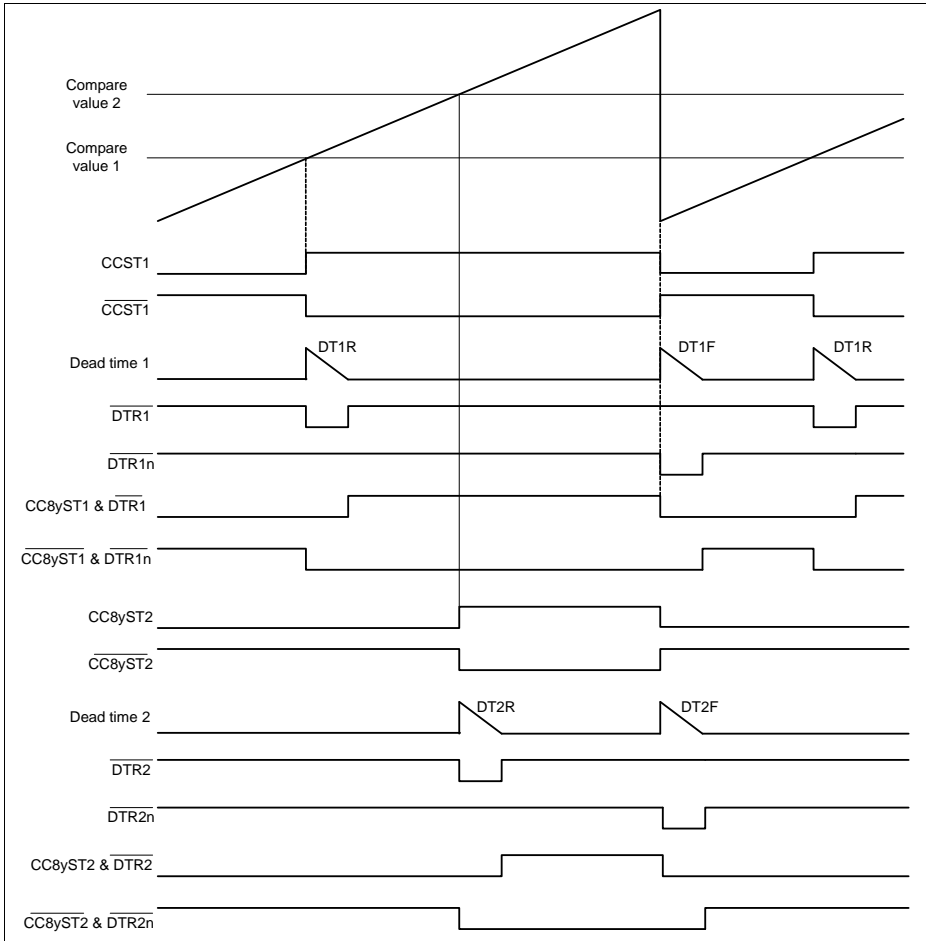


Figure 22-25 Edge Aligned - four outputs with dead time

Asymmetrical Edge Aligned Mode

There is also the possibility of using the two channels combined to generate an asymmetric PWM output. This mode is selected by setting the field **CC8yCHC.ASE** = 1_B.

In this mode, the compare channel 2 is disabled and therefore the outputs linked with this path are always in the passive state.

The status bit of the compare channel 1 is set when a compare match with the compare value 1 (field **CC8yCR1.CR1**) occurs and is cleared when a compare match with the compare value 2 (field **CC8yCR2.CR2**) occurs, see [Figure 22-26](#).

Capture/Compare Unit 8 (CCU8)

When the **CC8yCR2.CR2** is programmed with a value smaller than the one present in **CC8yCR1.CR1**, the **CCST1** bit is always 0_B.

The dead time values for the rising and falling transitions are controlled by the fields **CC8yDC1R.DT1R** and **CC8yDC1R.DT1F**, respectively.

Figure 22-27 and **Figure 22-28** show the timing diagram for the Edge Aligned mode when the asymmetric scheme is active.

Note: When an external signal is used to control the counting direction, the asymmetric mode cannot be enabled.

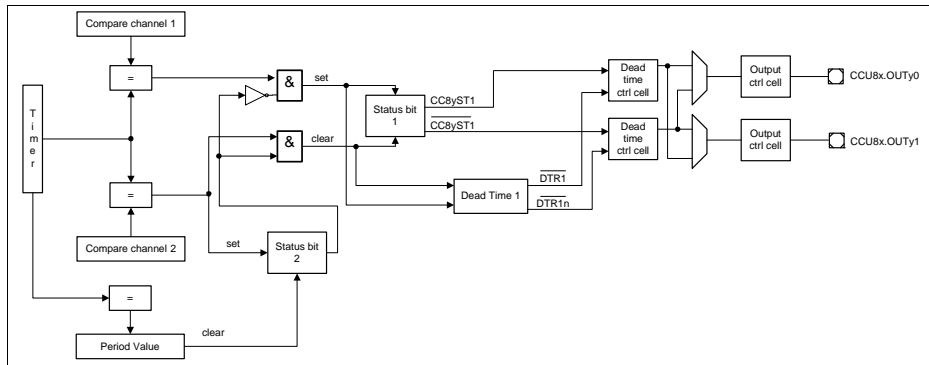


Figure 22-26 Edge Aligned with combined channels scheme

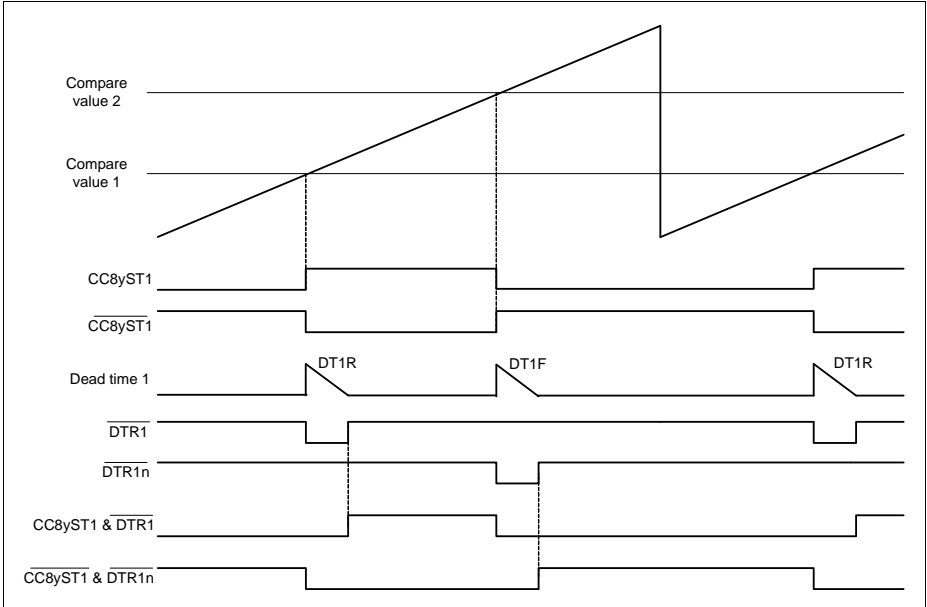


Figure 22-27 Edge Aligned - Asymmetric PWM timing, $CC8yCR1.CR1 < CC8yCR2.CR2$

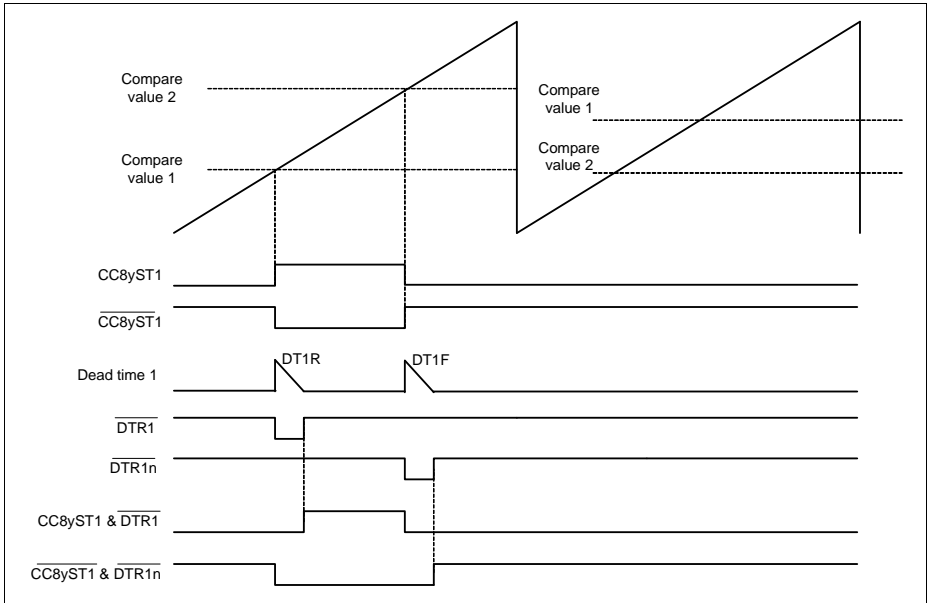


Figure 22-28 Edge Aligned - Asymmetric PWM timing, $CC8yCR1.CR1 > CC8yCR2.CR2$

22.2.7.2 Center Aligned Compare Modes

Standard Center Aligned Mode

In center aligned mode, like in edge aligned, it is possible to use the two compare channels independently. In this mode, each channel can generate a pair of PWM complementary signals with different duty cycle values, controlled via the **CC8yCR1** for channel 1 and **CC8yCR2** for channel 2.

For the dead time insertion, each channel as a pair of programmable values for the rise and fall transitions: **CC8yDC1R.DT1R** and **CC8yDC1R.DT1F** for channel 1; **CC8yDC2R.DT2R** and **CC8yDC2R.DT2F** for channel 2.

The major difference between the center and the edge aligned mode is directly linked to the set/clear logic of the status bit, see **Section 22.2.5**.

Figure 22-29 shows the scheme for both channels for this operating mode and **Figure 22-30** shows the timing diagrams for a specific channel.

Note: When an external signal is used to control the counting direction, the counting scheme is always edge aligned.

Capture/Compare Unit 8 (CCU8)

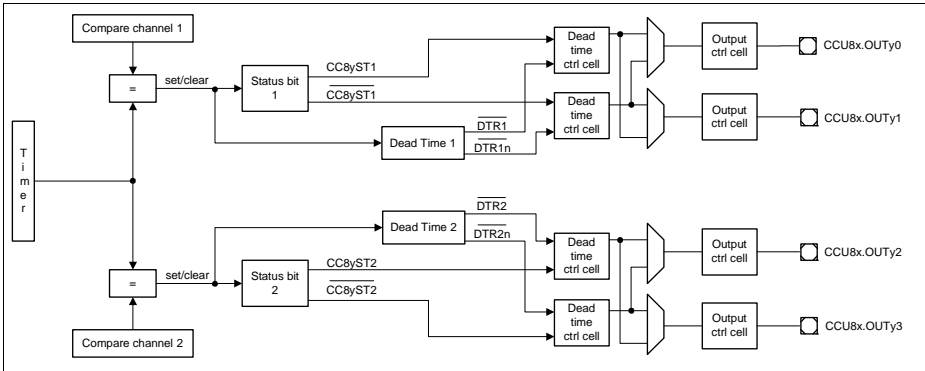


Figure 22-29 Center Aligned with two independent channels scheme

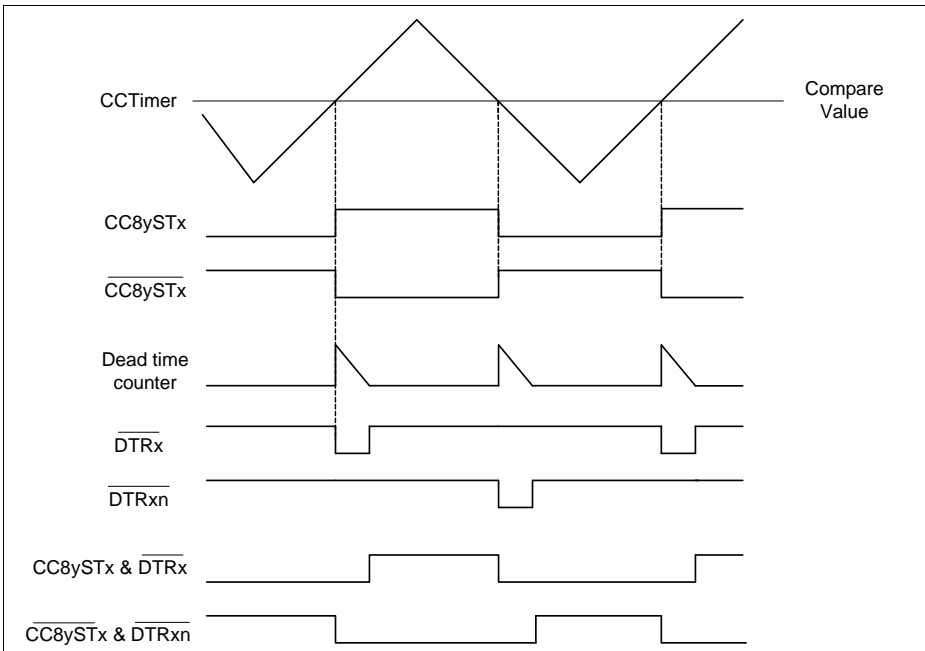


Figure 22-30 Center aligned - Independent channel with dead time

Asymmetrical Center Aligned Mode

The asymmetric mode is enabled in center aligned by setting the field **CC8yCHC.ASE** to 1_B.

Capture/Compare Unit 8 (CCU8)

In this mode, like in Edge Aligned, the outputs linked with the compare channel 2 are set to their passive levels.

The status bit, CC8yST1, is set when a compare match of channel 1 occurs while counting up, and is cleared when a compare match of channel 2 occurs while counting down, see **Figure 22-31**.

The dead time rise and fall times are controlled by the values programmed into the fields, **CC8yDC1R.DT1R** and **CC8yDC1R.DT1F**, respectively.

Figure 22-32 shows the timing diagram for the asymmetric mode. Notice that even in asymmetric mode the dead time can be disabled in each of the outputs independently.

Note: When an external signal is used to control the counting direction, the asymmetric mode cannot be enabled.

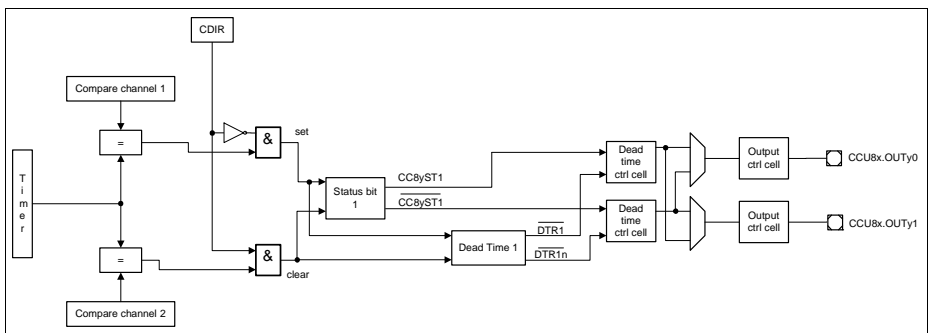


Figure 22-31 Center Aligned Asymmetric mode scheme

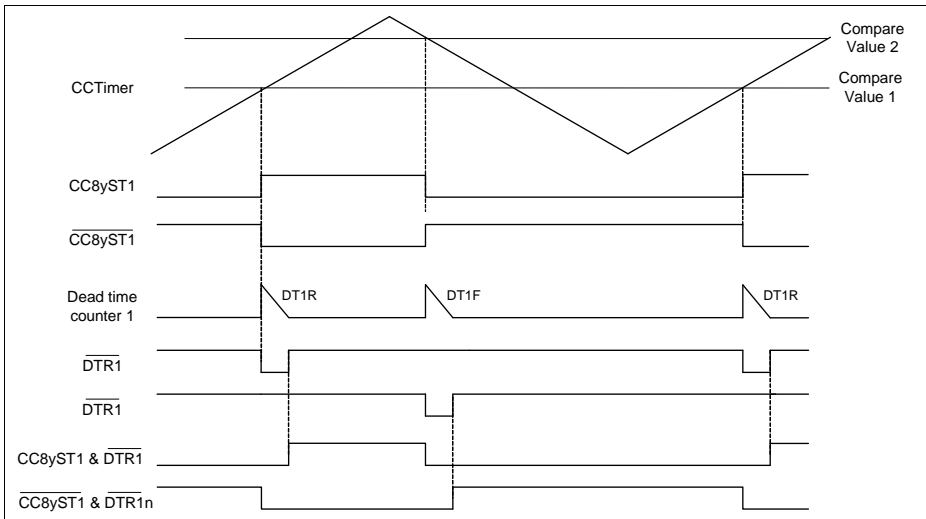


Figure 22-32 Asymmetric Center aligned mode with dead time

22.2.8 External Events Control

Each CCU8 slice has the possibility of using up to three different input events, see [Section 22.2.2](#). These three events can then be mapped to Timer Slice functions (the full set of available functions is described at [Section 22.2.3](#)).

These events can be mapped to any of the CCU8x.INy[P...A] inputs and there isn't any imposition that an event cannot be used to perform several functions or, that an input cannot be mapped to several events (e.g. input X triggers event 0 with rising edge and triggers event 1 with the falling edge).

22.2.8.1 External Start/Stop

To select an external start function, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the [CC8yINS.EVxIS](#) register and indicating the active edge of the signal on the [CC8yINS.EVxEM](#) register.

This event should be then mapped to the start or stop functionality by setting the [CC8yCMC.STRTS](#) (for the start) or the [CC8yTC.ENDM](#) (for the stop) with the proper value.

The same procedure is applicable to the stop functionality.

Notice that both start and stop functions are edge and not level active and therefore the active/passive configuration is set only by the [CC8yINS.EVxEM](#).

Capture/Compare Unit 8 (CCU8)

The external stop by default just clears the run bit (**CC8yTCST.TRB**), while the start functions does the opposite. Nevertheless one can select an extended subset of functions for the external start and stop. This subset is controlled by the registers **CC8yTC.ENDM** (for the stop) and **CC8yTC.STRM** (for the start).

For the start subset (**CC8yTC.STRM**):

- sets the run bit/starts the timer (resume operation)
- clears the timer, sets the run bit/starts the timer (flush and start)

For the stop subset (**CC8yTC.ENDM**):

- clears the run/stops the timer (stop)
- clears the timer (flush)
- clears the timer, clears the run bit/stops the timer (flush and stop)

If in conjunction with an external start/stop (configured also/only as flush) and external up/down signal is used, during the flush operation the timer is going to be set to 0000_H if the actual counting direction is up or set with the value of the period register if the counting direction is down.

Figure 22-33 to **Figure 22-36** shows the usage of two signals to perform the start/stop functions in all the previously mentioned subsets. External Signal(1) acts as an active HIGH start signal, while External Signal(2) is used as an active HIGH stop function.

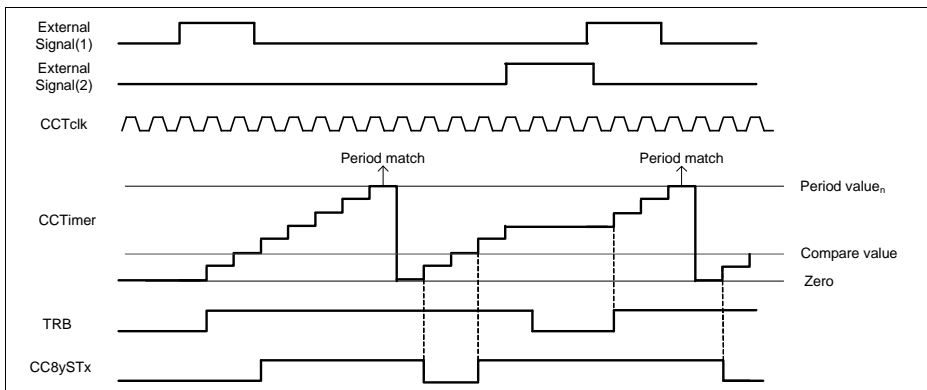


Figure 22-33 Start (as start)/ stop (as stop) - **CC8yTC.STRM = 0_B , **CC8yTC.ENDM** = 00_B**

Capture/Compare Unit 8 (CCU8)

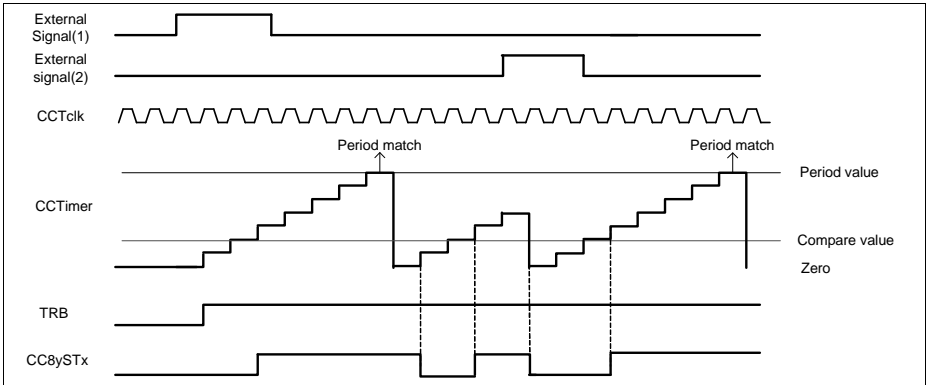


Figure 22-34 Start (as start)/ stop (as flush) - $CC8yTC.STRM = 0_B$, $CC8yTC.ENDM = 01_B$

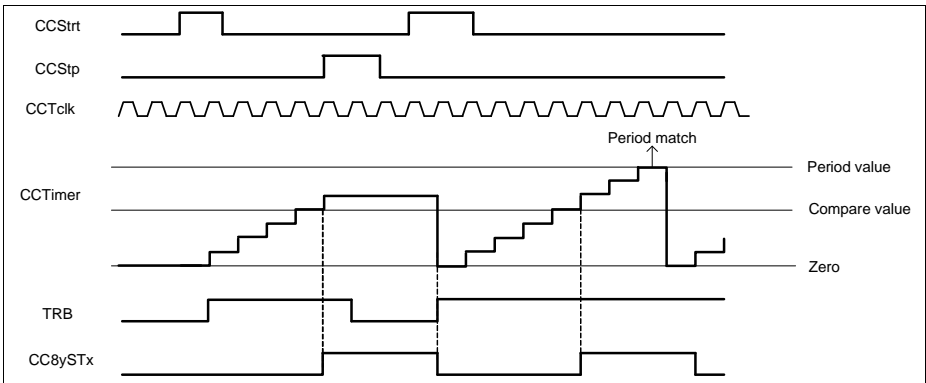
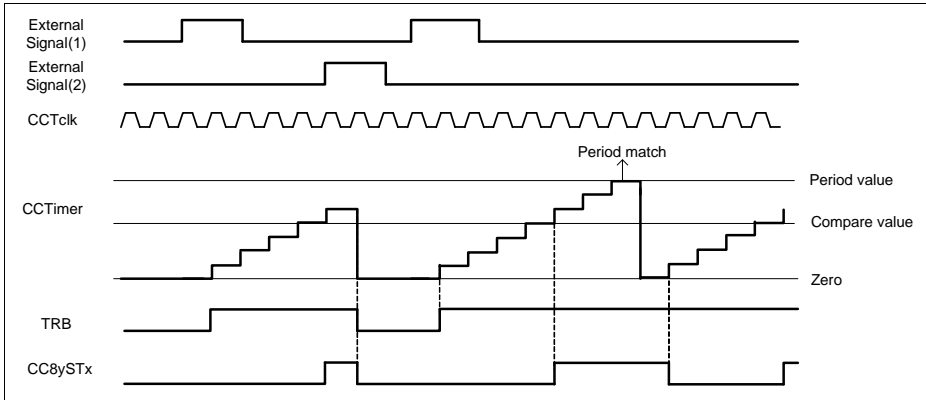


Figure 22-35 Start (as flush and start)/ stop (as stop) - $CC8yTC.STRM = 1_B$, $CC8yTC.ENDM = 00_B$



**Figure 22-36 Start (as start)/ stop (as flush and stop) - $CC8yTC.STRM = 0_B$,
 $CC8yTC.ENDM = 10_B$**

22.2.8.2 External Counting Direction

There is the possibility of selecting an input signal to act as counting up/counting down control.

To select an external up/down control, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC8yINS.EVxIS** register and indicating the active level of the signal on the **CC8yINS.EVxLM** register. This event should be then mapped to the up/down functionality by setting the **CC8yCMC.UDS** with the proper value.

Notice that the up/down function is level active and therefore the active/passive configuration is set only by the **CC8yINS.EVxLM**.

The status bit of the slice (CCSTx) is always set when the timer value is equal or greater than the value stored in the compare register, see [Section 22.2.6](#).

The update of the period and compare register values is done when:

- with the next clock after a period match, while counting up (**CC8yTCST.CDIR** = 0_B)
- with the next clock after a one match, while counting down (**CC8yTCST.CDIR** = 1_B)

The value of the **CC8yTCST.CDIR** register is updated accordingly with the changes on the decoded event. The Up/Down direction is always understood as **CC8yTCST.CDIR** = 1 when counting down and **CC8yTCST.CDIR** = 0_B when counting up. Using an external signal to perform the up/down counting function and configuring the event as active HIGH means that the timer is counting up when the signal is HIGH and counting down when LOW.

Capture/Compare Unit 8 (CCU8)

Figure 22-37 shows an external signal being used to control the counting direction of the time. This signal was selected as active HIGH, which means that the timer is counting down while the signal is HIGH and counting up when the signal is LOW.

*Note: For a signal that should impose an increment when LOW and a decrement when HIGH, the user needs to set the **CC8yINS.EVxLM** = 0_B. When the operation is switched, then the user should set **CC8yINS.EVxLM** = 1_B.*

Note: Using an external counting direction control, sets the slice in edge aligned mode.

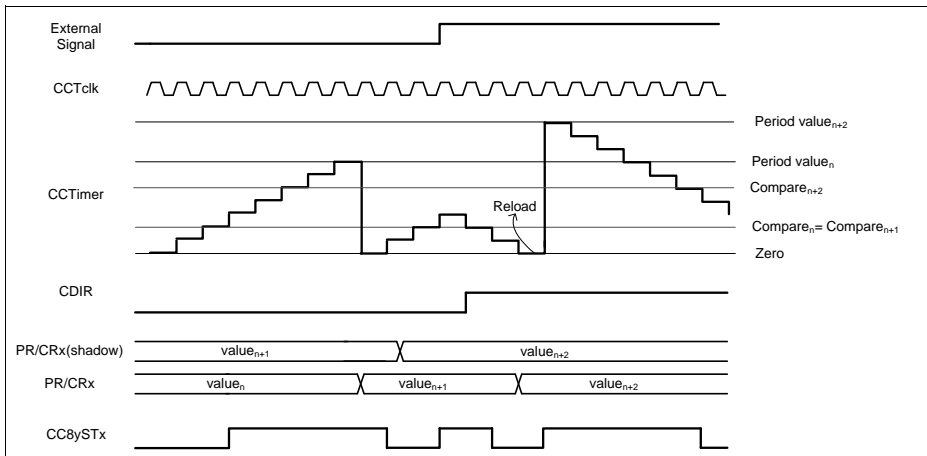


Figure 22-37 External counting direction

22.2.8.3 External Gating Signal

For pulse measurement, the user has the possibility of selecting an input signal that operates as counting gating.

To select an external gating control, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC8yINS.EVxIS** register and indicating the active level of the signal on the **CC8yINS.EVxLM** register. This event should be then mapped to the gating functionality by setting the **CC8yCMC.GATES** with the proper value.

Notice that the gating function is level active and therefore the active/passive configuration is set only by the **CC8yINS.EVxLM**.

The status bit during an external gating signal continues to be asserted when the compare value is reached and deasserted when the counter reaches 0000_H. One should note that the counter continues to use the period register to identify a wrap around condition. **Figure 22-38** shows the usage of an external signal for gating the slice

counter. The signal was set as active LOW, which means the counter gating functionality is active when the external value is zero.

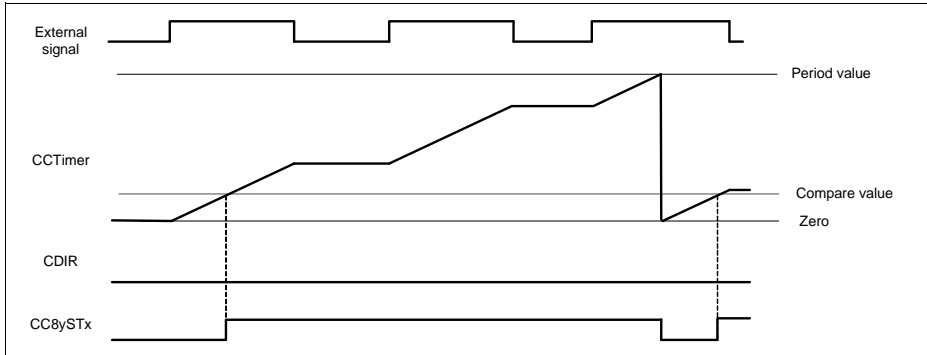


Figure 22-38 External gating

22.2.8.4 External Count Signal

There is also the possibility of selecting an external signal to act as the counting event. To select an external counting, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC8yINS.EVxIS** register and indicating the active edge of the signal on the **CC8yINS.EVxEM** register. This event should be then mapped to the counting functionality by setting the **CC8yCMC.CNTS** with the proper value.

Notice that the counting function is edge active and therefore the active/passive configuration is set only by the **CC8yINS.EVxEM**.

One can select just a the rising, falling or both edges to perform a count. On **Figure 22-39**, the external signal was selected as a counter event for both falling and rising edges. Wrap around condition is still applied with a comparison with the period register.

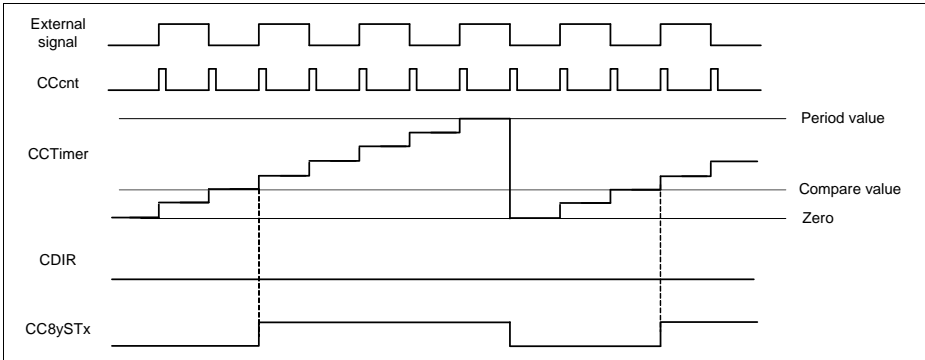


Figure 22-39 External count

22.2.8.5 External Load

Each slice of the CCU8 also has a functionality that enables the user to select an external signal as trigger for reloading the value of the timer with the current value of one compare register (if **CC8yTCST.CDIR** = 0_B) or with the value of the period register (if **CC8yTCST.CDIR** = 1_B).

The timer can be reloaded with the value from the compare channel 1 or compare channel 2 depending on the value set in the **CC8yTC.TLS** field, see [Figure 22-40](#).

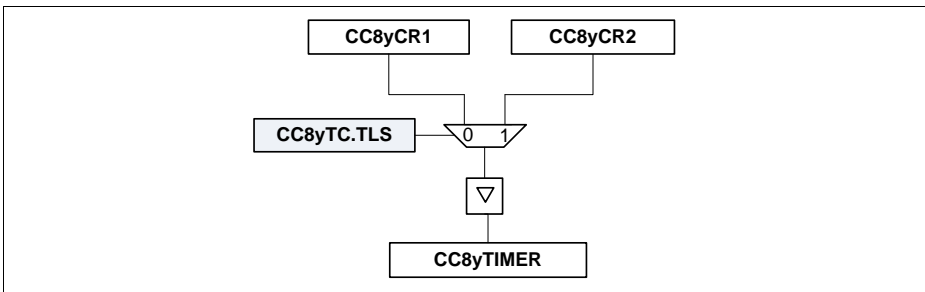


Figure 22-40 Timer load selection

To select an external load signal, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC8yINS.EVxIS** register and indicating the active edge of the signal on the **CC8yINS.EVxEM** register. This event should be then mapped to the load functionality by setting the **CC8yCMC.LDS** with the proper value.

Notice that the load function is edge active and therefore the active/passive configuration is set only by the **CC8yINS.EVxEM**.

Capture/Compare Unit 8 (CCU8)

On figure **Figure 22-41**, the external signal (1) was used to act as a load trigger, active on the rising edge. Every time that a rising edge on external signal (1) is detected, the timer value is loaded with the value present on the compare register. If an external signal is being used to control the counting direction, up or down, the timer value can be loaded also with the value set in the period register. The External signal (2) represents the counting direction control (active HIGH). If at the moment that a load trigger is detected, the signal controlling the counting direction is imposing a decrement, then the value set in the timer is the period value.

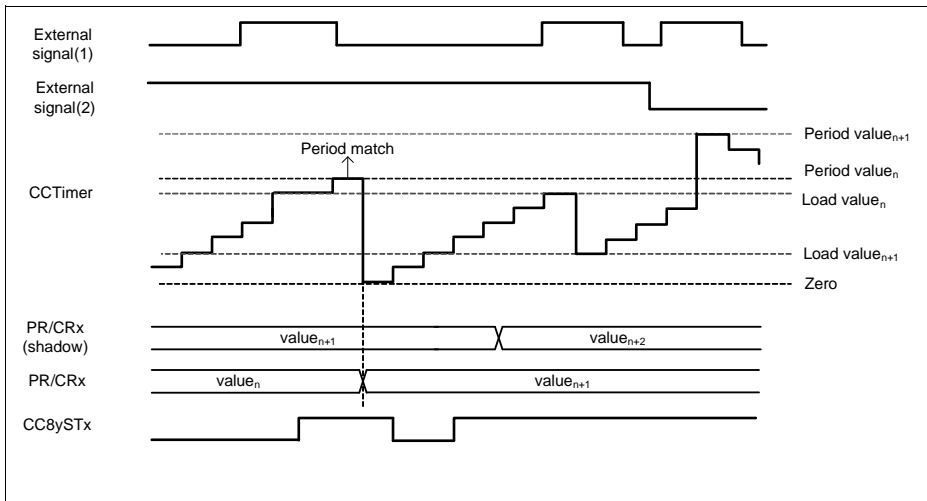


Figure 22-41 External load

22.2.8.6 External Capture

When selecting an external signal to be used as a capture trigger (if **CC8yCMC.CAP0S** or **CC8yCMC.CAP1S** are different from 0_H), the user is automatically setting the specific slice into capture mode.

In capture mode the user can have up to four capture registers, see **Figure 22-44**: capture register 0 (**CC8yC0V**), capture register 1 (**CC8yC1V**), capture register 2 (**CC8yC2V**) and capture register 3 (**CC8yC3V**).

These registers are shared between compare and capture modes, which imposes:

- if **CC8yC0V** and **CC8yC1V** are used for capturing, the compare registers **CC8yCR1** and **CC8yCR1S** are not available (compare channel 1 is not available)
- if **CC8yC2V** and **CC8yC3V** are used for capturing, the compare registers **CC8yCR2** and **CC8yCR2S** are not available (compare channel 2 is not available)

Capture/Compare Unit 8 (CCU8)

To select an external capture signal, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC8yINS.EVxIS** register and indicating the active edge of the signal on the **CC8yINS.EVxEM** register.

This event should be then mapped to the capture functionality by setting the **CC8yCMC.CAP0S/CC8yCMC.CAP1S** with the proper value.

Notice that the capture function is edge active and therefore the active/passive configuration is set only by the **CC8yINS.EVxEM**.

The user has the possibility of selecting the following capture schemes:

- Different capture events for **CC8yC0V/CC8yC1V** and **CC8yC2V/CC8yC3V**
- The same capture event for **CC8yC0V/CC8yC1V** and **CC8yC2V/CC8yC3V** with the same capture edge. For this capture scheme, only the CCcapt1 functionality needs to be programmed. To enable this scheme, the field **CC8yTC.SCE** needs to be set to 1_B.

Different Capture Events (SCE = 0_B)

Every time that a capture trigger 1 occurs, CCcapt1, the actual value of the timer is captured into the capture register 3 and the previous value stored in this register is transferred into capture register 2.

Every time that a capture trigger 0 occurs, CCcapt0, the actual value of the timer is captured into the capture register 1 and the previous value stored in this register is transferred into capture register 0.

Every time that a capture procedure into one of the registers occurs, the respective full flag is set. This flag is cleared automatically by HW when the SW reads back the value of the capture register (by reading the specific capture register or by reading the extended capture read value, see [Section 22.2.8.7](#)).

The capture of a new value into a specific capture registers is dictated by the status of the full flag as follows:

$$CC8yCIV_{capt} = \text{NOT}(CC8yCIV_{full_flag} \text{ AND } CC8yC0V_{full_flag}) \quad (22.4)$$

$$CC8yC0V_{capt} = CC8yCIV_{full_flag} \text{ AND NOT}(CC8yC0V_{full_flag}) \quad (22.5)$$

It is also possible to disable the effect of the full flags by setting the **CC8yTC.CCS = 1_B**. This enables a continuous capturing independent if the values captured have been read or not.

On [Figure 22-42](#), an external signal was selected as an event for capturing the timer value into the **CC8yC0V/CC8yC1V** registers. The status bit, CC8ySTx, during capture mode is asserted whenever a capture trigger is detected and de asserted when the counter matches 0000_H.

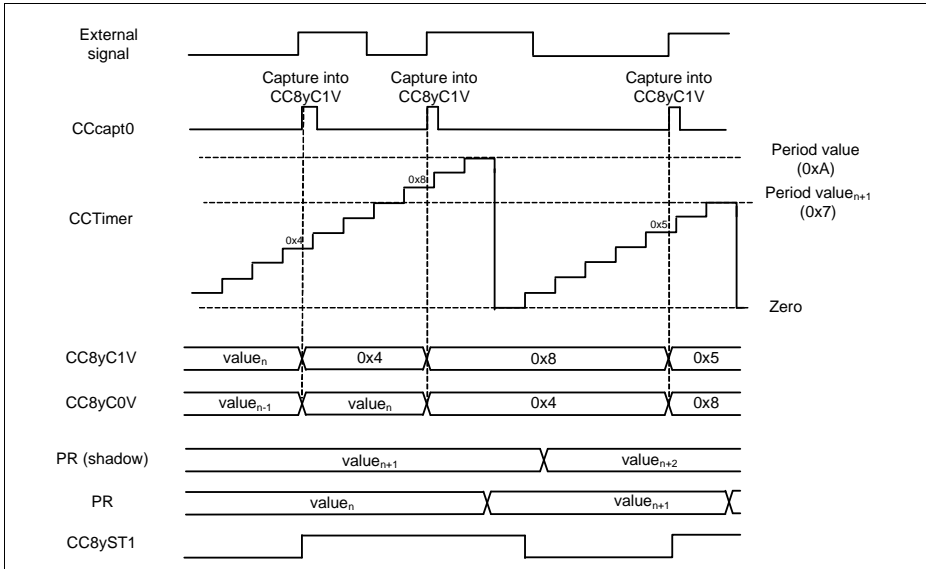


Figure 22-42 External capture - $CC8yCMC.CAP0S \neq 00_B$, $CC8yCMC.CAP1S = 00_B$

On [Figure 22-43](#), two different signals were used as trigger sources for capturing the timer value into the [CC8yC0V/CC8yC1V](#) and [CC8yC2V/CC8yC3V](#) registers. External signal(1) was selected as an rising edge active source for the channel 1 capture trigger. External signal(2) was selected has the source for the channel 2 capture trigger, but as opposite to the external signal(1), the active edge was set as falling.

See [Section 22.2.14.4](#) for the complete capture mode usage description.

Capture/Compare Unit 8 (CCU8)

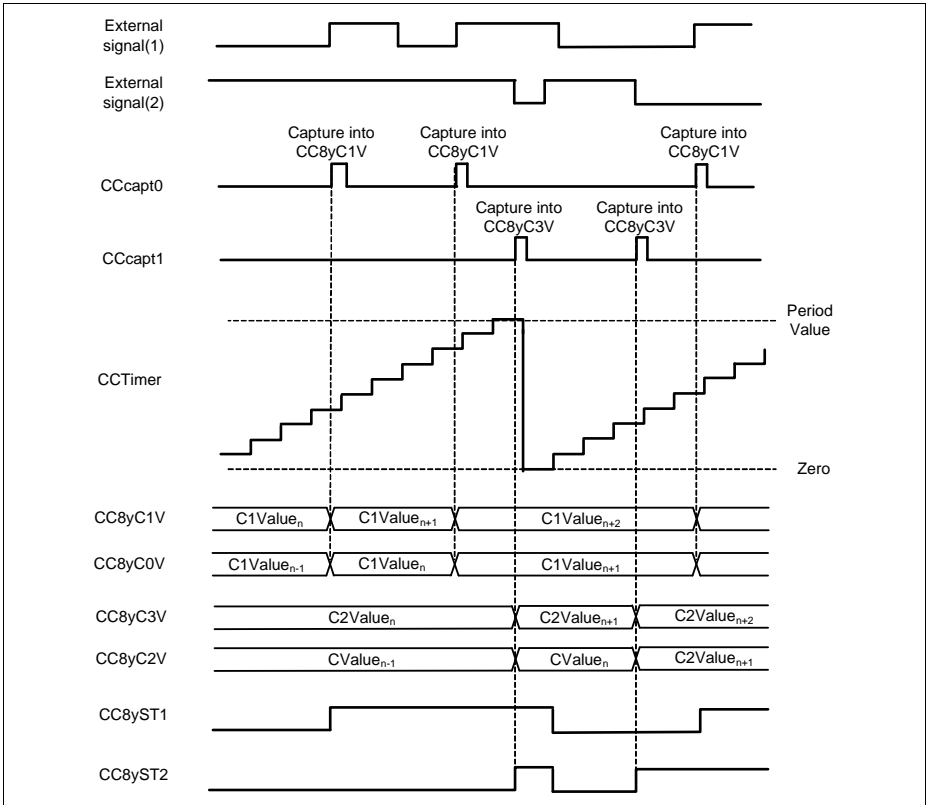


Figure 22-43 External capture - $CC8yCMC.CAP0S \neq 00_B$, $CC8yCMC.CAP1S \neq 00_B$

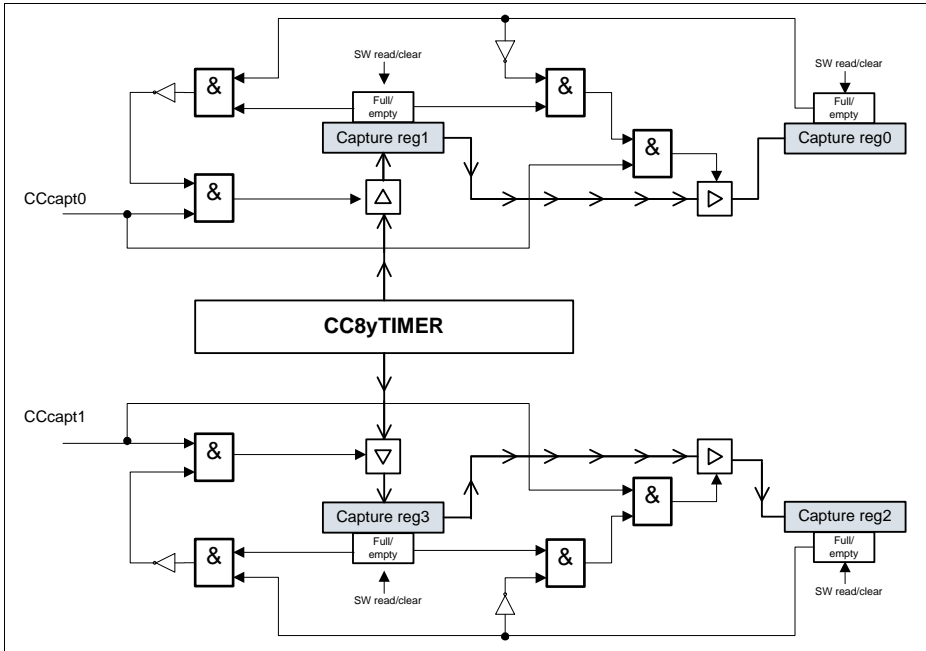


Figure 22-44 Slice capture logic

Same Capture Event (SCE = 1_B)

Setting the field **CC8yTC.SCE = 1_B**, enables the possibility of having 4 capture registers linked with the same capture event, **Figure 22-46**. The functionality that controls the capture is the **CCcapt1**.

The capture logic follows the same structure shown in **Figure 22-44** but extended to a four register chain, see **Figure 22-45**. The same full flag lock rules are applied to the four register chain (it also can be disabled by setting the **CC8yTC.CCS = 1_B**):

$$CC8yC3V_{capt} = \text{NOT}(CC8yC3V_{full_flag} \text{ AND } CC8yC2V_{full_flag} \text{ AND } CC8yC2V_{full_flag} \text{ AND } CC8yC1V_{full_flag}) \quad (22.6)$$

$$CC8yC2V_{capt} = CC8yC3V_{full_flag} \text{ AND NOT}(CC8yC2V_{full_flag} \text{ AND } CC8yC1V_{full_flag} \text{ AND } CC8yC0V_{full_flag}) \quad (22.7)$$

$$CC8yC1V_{capt} = CC8yC2V_{full_flag} \text{ AND NOT}(CC8yC1V_{full_flag} \text{ AND } CC8yC0V_{full_flag}) \quad (22.8)$$

$$CC8yC0V_{capt} = CC8yC1V_{full_flag} \text{ AND NOT}(CC8yC0V_{full_flag}) \quad (22.9)$$

Capture/Compare Unit 8 (CCU8)

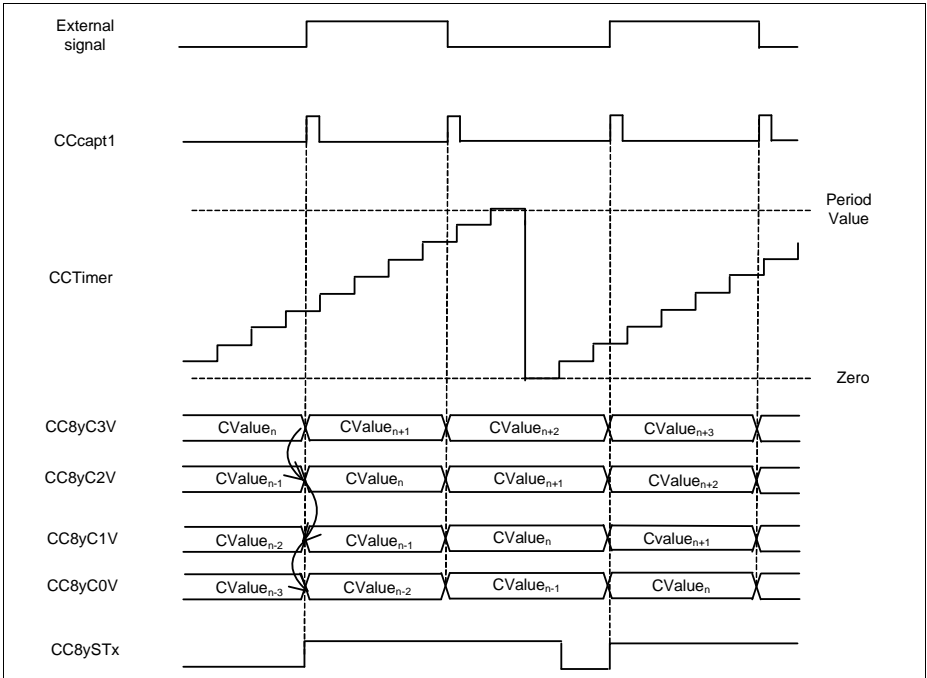


Figure 22-45 External Capture - $CC8yTC.SCE = 1_B$

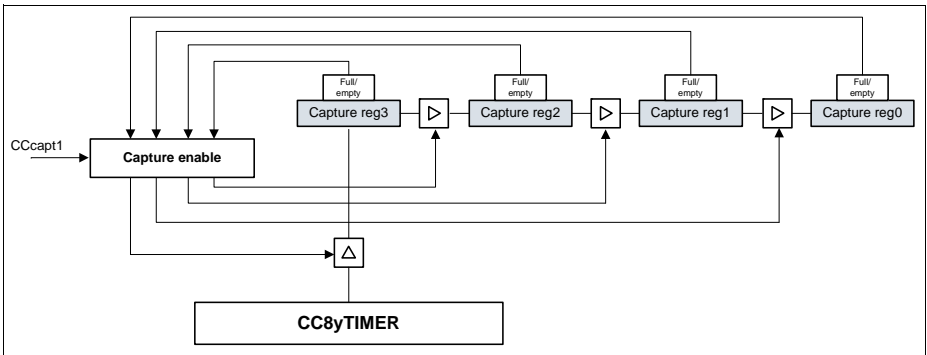


Figure 22-46 Slice Capture Logic - $CC8yTC.SCE = 1_B$

22.2.8.7 Capture Extended Read Back Mode

Each Timer Slice capture logic can operate in a FIFO read back mode. This mode can be enabled by setting the **CC8yTC.ECM** = 1_B. This Extended Read back mode allows the software to read back the capture data always from the same address (**CC8yECRD0** for the structure linked with the capture trigger 0 or **CC8yECRD1** for the one linked with capture trigger 1). This read back will always return the oldest captured value, enabling an easy software routine implementation for reconstructing the capture data.

This function allows the usage of a FIFO structure for each capturing trigger. This relaxes the software read back routine when multiple capture triggers are present, and the software is not fast enough to perform a read operation in each capture event.

This FIFO read back function is present for a depth-4 and depth-2 FIFO structure.

The read back data contains also a lost value bitfield, that indicates if a capture trigger was lost due to the fact that the FIFO structure was full. This bitfield is set whenever a capture event was sensed and the FIFO was full (regardless if the continuous capture mode was enabled or not). This bitfield is cleared automatically by HW whenever the next read of the **CC8yECRD0/CC8yECRD1** register occurs. This bitfield does not indicate how many capture events were lost, it just indicates that between two ECRD reads at least a capture event was lost (this can help the SW evaluate which part of the data read, can be used for calculation).

*Note: When the ECM bitfield is set, reading the individual capture registers is still possible. Nevertheless the full flags can only be cleared by the HW when a read back is done via the **CC8yECRD0/CC8yECRD1** address.*

Depth 4 Structure

The FIFO depth-4 structure is present in the hardware when the capture trigger 1 is enabled and the **CC8yTC.SCE** = 1_B (same capture event), **Figure 22-47**.

Capture/Compare Unit 8 (CCU8)

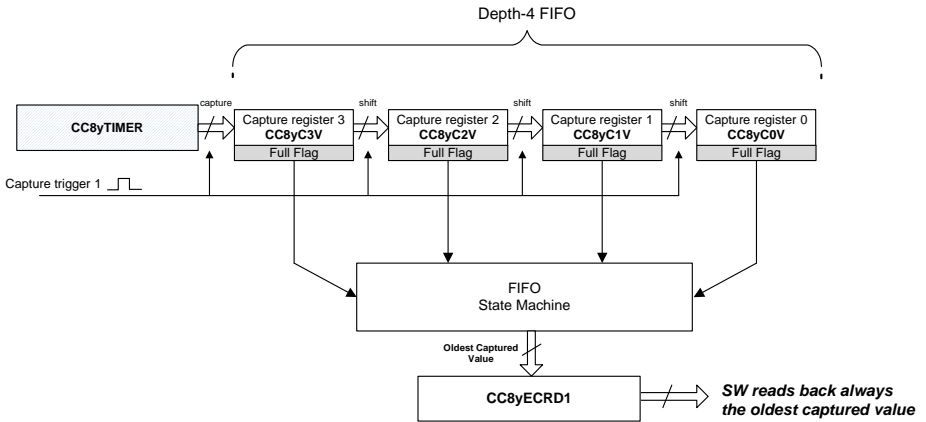


Figure 22-47 Capture Extended Read Back - Depth 4

Capture/Compare Unit 8 (CCU8)

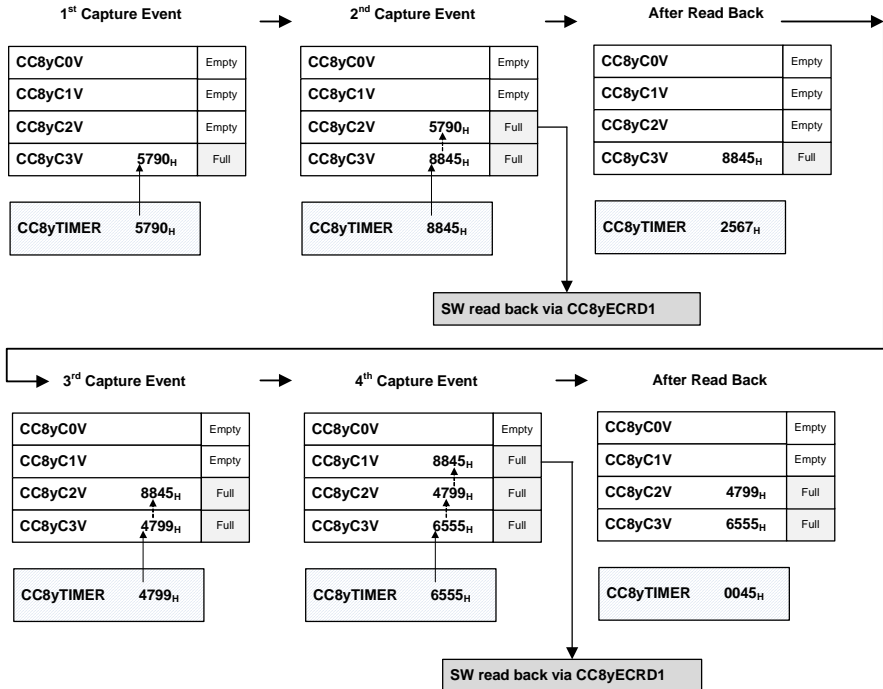


Figure 22-48 Depth 4 software access example

Depth 2 Structure

Each Timer Slice can have two capture structures of depth-2: one used with capture trigger 0 and another with capture trigger 1.

The one linked with capture trigger 0, is accessed via the **CC8yECDR0** while the one linked with the capture trigger 1 is accessed via the **CC8yECDR1**, **Figure 22-49**.

Capture/Compare Unit 8 (CCU8)

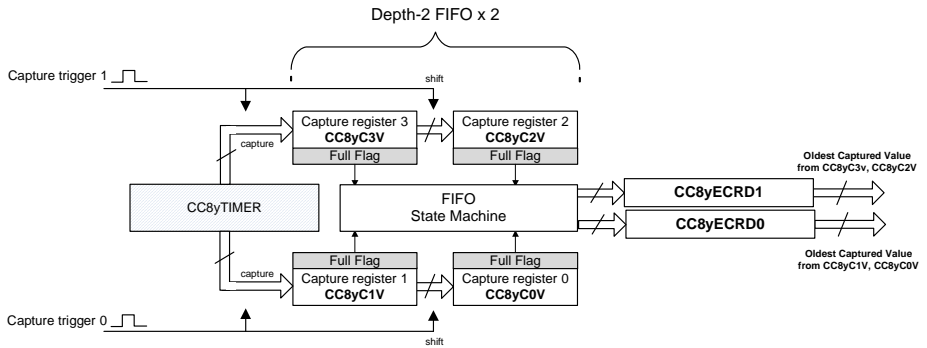


Figure 22-49 Capture Extended Read Back - Depth 2

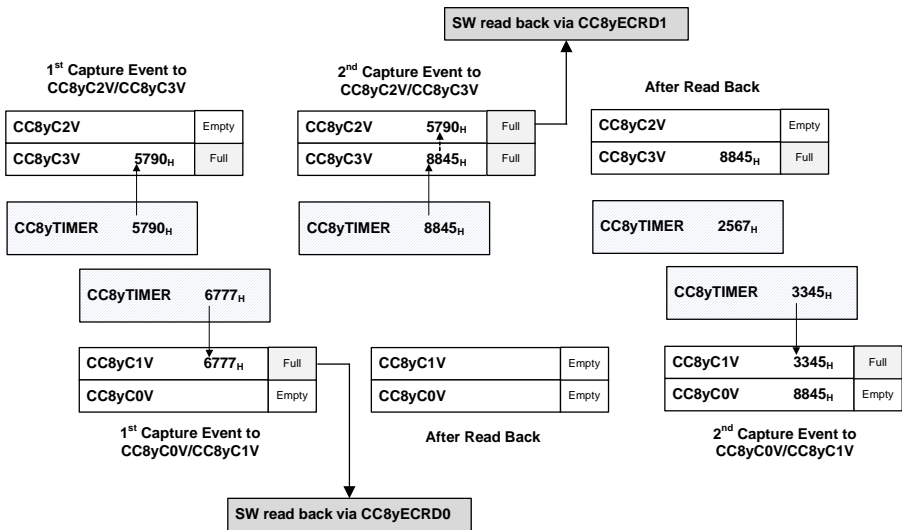


Figure 22-50 Depth 2 software access example

22.2.8.8 External Modulation

An external signal can be used also to perform a modulation at the output of each slice. To select an external modulation signal, one should map one of the input signals to one of the events, by setting the required value in the **CC8yINS.EVxIS** register and indicating the active level of the signal on the **CC8yINS.EVxLM** register. This event should be then

Capture/Compare Unit 8 (CCU8)

mapped to the modulation functionality by setting the **CC8yCMC.MOS** = 01_B if event 0 is being used, **CC8yCMC.MOS** = 10_B if event 1 or **CC8yCMC.MOS** = 11_B if event 2.

Notice that the modulation function is level active and therefore the active/passive configuration is set only by the **CC8yINS.EVxLM**.

The external modulation signal can be applied to each compare channel independently, or it can be applied to both channels, by setting **CC8yTC.EME** = 11_B.

The modulation has two modes of operation:

- modulation event is used to reset the CC8ySTx bit - **CC8yTC.EMT** = 0_B
- modulation event is used to gate the outputs - **CC8yTC.EMT** = 1_B

On **Figure 22-51**, we have an external signal configured to act as modulation source that clears the ST bit, **CC8yTC.EMT** = 0_B. It was programmed to be an active LOW event and therefore, when this signal is LOW the output value is following the normal ACTIVE/PASSIVE rules.

When the signal is HIGH (inactive state), then the CC8ySTx bit is cleared and the output is forced into the PASSIVE state. Notice that the values of the status bit, CC8ySTx and the specific output CCU8x.OUTy are not linked together. One can choose for the output to be active LOW through the **CC8yPSL.PSLx** bit.

The exit of the external modulation inactive state is synchronized with the PWM period due to the fact that the CC8ySTx bit is cleared and cannot be set while the modulation signal is inactive.

The entering into inactive state also can be synchronized with the PWM period, by setting **CC8yTC.EMS** = 1_B. With this all possible glitches at the output are avoided, see **Figure 22-52**.

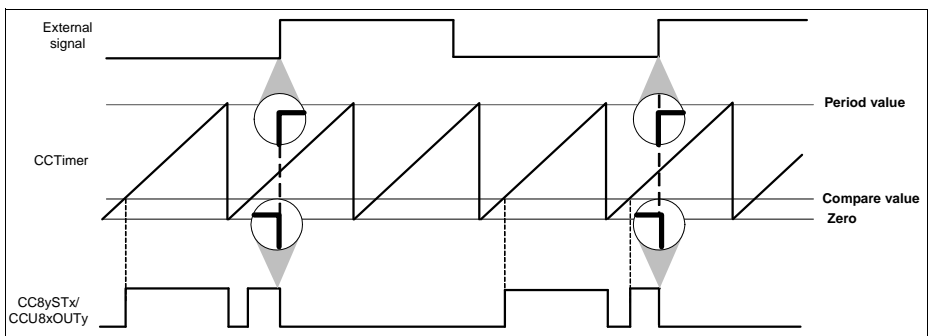


Figure 22-51 External modulation resets the ST bit - **CC8yTC.EMS = 0_B**

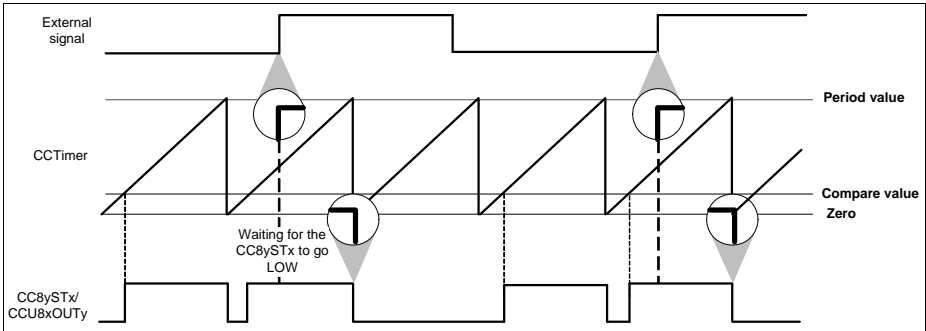


Figure 22-52 External modulation clearing the ST bit - $CC8yTC.EMS = 1_B$

On [Figure 22-53](#), the external modulation event was used as gating signal of the outputs, $CC8yTC.EMT = 1_B$. The external signal was configured to be active HIGH, $CC8yINS.EVxLM = 0_B$, which means that when the external signal is HIGH the outputs are set to the PASSIVE state. In this mode, the gating event can also be synchronized with the PWM signal by setting the $CC8yTC.EMS = 1_B$.

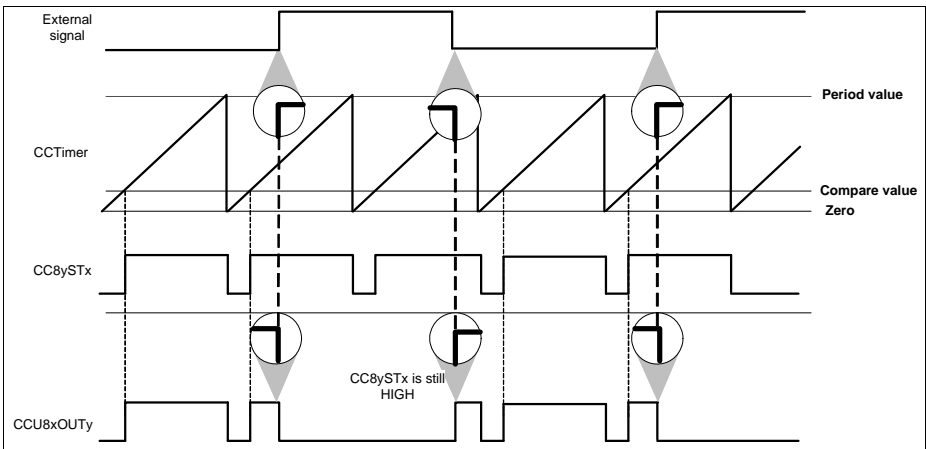


Figure 22-53 External modulation gating the output - $CC8yTC.EMT = 1_B$

22.2.8.9 Trap Function

The TRAP functionality allows the PWM outputs to react on the state of an input pin. This functionality can be used to switch off the power devices if the TRAP input becomes active.

Capture/Compare Unit 8 (CCU8)

To select the trap functionality, one should map one of the input signals to event number 2, by setting the required value in the **CC8yINS.EV2IS** register and indicating the active level of the signal on the **CC8yINS.EV2LM** register. This event should be then mapped to the trap functionality by setting the **CC8yCMC.TS = 1_B**.

Notice that the trap function is level active and therefore the active/passive configuration is set only by the **CC8yINS.EV2LM**.

There are two bitfields that can be monitored via software to crosscheck the TRAP function, **Figure 22-54**:

- The TRAP state bit, **CC8yINTS.E2AS**. This bitfield if the TRAP is currently active or not. This bitfield is therefore setting the specific Timer Slice output, into ACTIVE or PASSIVE state.
- The TRAP Flag, **CC8yINTS.TRPF**. This bitfield is used as a remainder in the case that the TRAP condition is cleared automatically via hardware. This field needs to be cleared by the software.

The E2AS can be configured to affect all of the CCU8 slice outputs, or a specific sub set of outputs via the **CC8yTC.TRAPEy** bit fields.

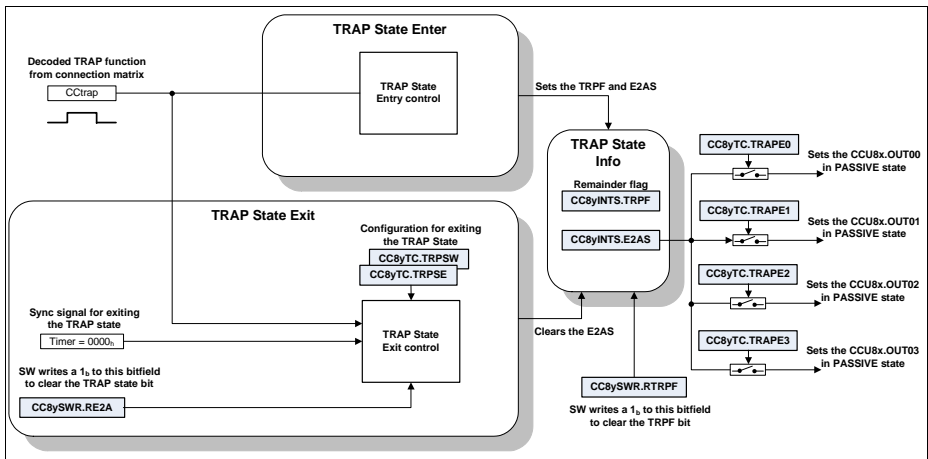


Figure 22-54 Trap control diagram

When a TRAP condition is detected at the selected input pin, both the Trap Flag and the Trap State bit are set to 1_B. The Trap State is entered immediately, by setting the CCU8xOUTy into the programmed PASSIVE state, **Figure 22-55**.

Exiting the Trap State can be done in two ways (**CC8yTC.TRPSW** register):

- automatically via HW, when the TRAP signal becomes inactive - **CC8yTC.TRPSW = 0_B**

Capture/Compare Unit 8 (CCU8)

- by SW only, by clearing the **CC8yINTS.E2AS**. The clearing is only possible if the input TRAP signal is in inactive state - **CC8yTC.TRPSW = 1_B**

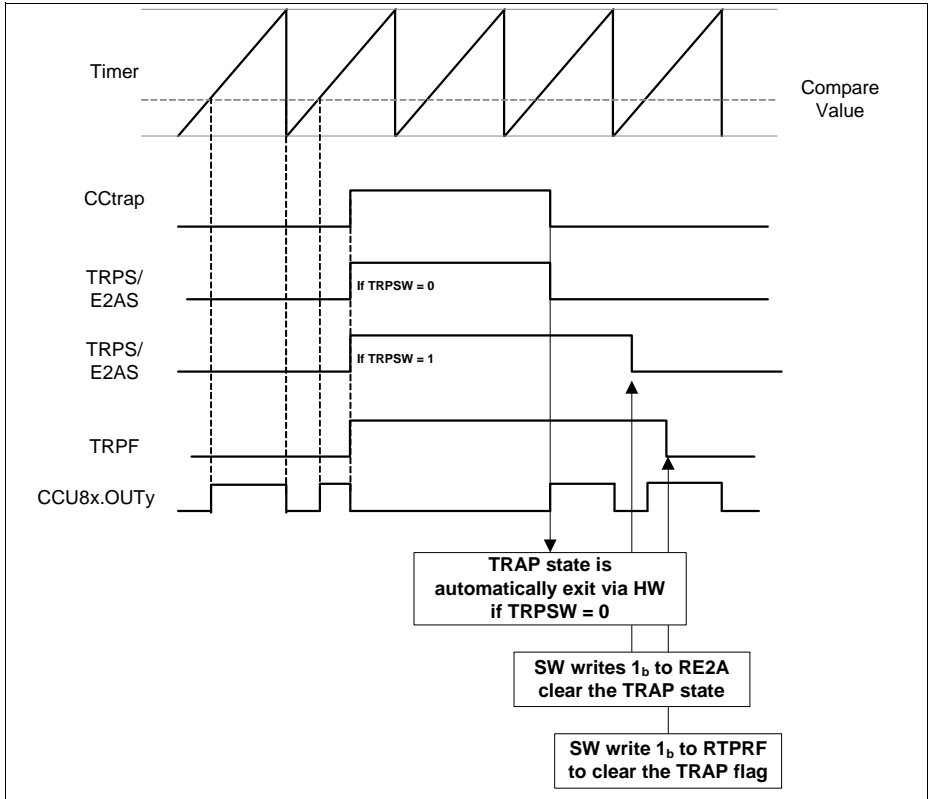


Figure 22-55 Trap timing diagram, **CC8yTCST.CDIR = 0** **CC8yPSL.PSL = 0**

It is also possible to synchronize the exiting of the TRAP state with the PWM signal, **Figure 22-56**. This function is enabled when the bitfield **CC8yTC.TRPSE = 1_B**.

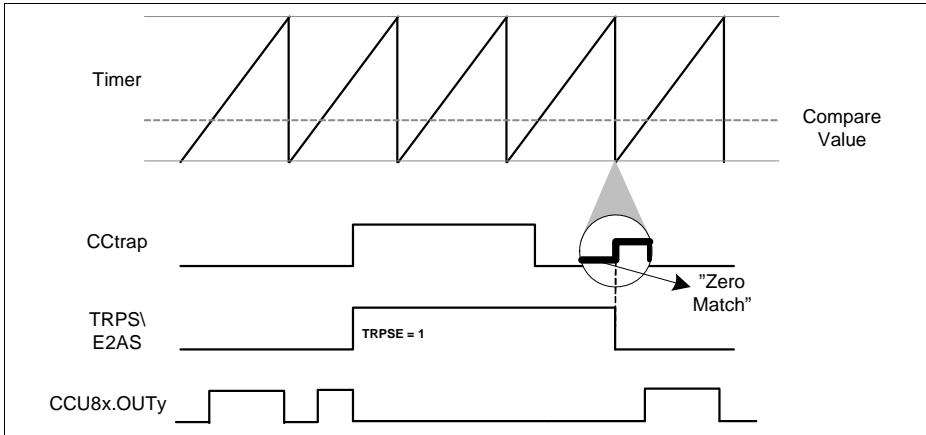


Figure 22-56 Trap synchronization with the PWM signal

22.2.8.10 Status Bit Override

For complex timed output control, each slice has a functionality that enables the override of the status bit of compare channel 1 (CC8yST1) with a value passed through an external signal.

The override of the status bit, can then lead to a change on the output pins CCU8x.OUTy0 and CCU8x.OUTy1 (from inactive to active or vice versa).

To enable this functionality, two signals are needed:

- One signal that acts as a trigger to override the status bit (edge active)
- One signal that contains the value to be set in the status bit (level active)

To select the status bit override functionality, one should map the signal that acts as trigger to the event number 1, by setting the required value in the **CC8yINS.EV1IS** register and indicating the active edge of the signal on the **CC8yINS.EV1EM** register.

The signal that carries the value to be set on the status bit, needs to be mapped to the event number 2, by setting the required value in the **CC8yINS.EV2IS** register. The **CC8yINS.EV2LM** register should be set to 0_B if no inversion on the signal is needed and to 1_B otherwise.

The events should be then mapped to the status bit functionality by setting the **CC8yCMC.OFS = 1_B**.

Figure 22-57 shows the functionality of the status bit override, when the external signal(1) was selected as trigger source (rising edge active) and the external signal(2) was selected as override value.

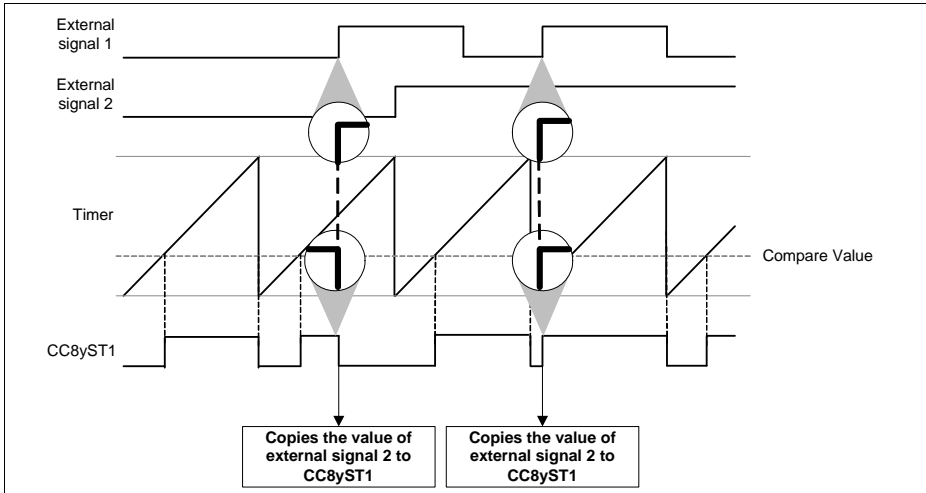


Figure 22-57 Status bit override

22.2.9 Multi-Channel Support

The multi channel control mode is selected individually in each slice by setting the **CC8yTC.MCME_x = 1_B**.

With this mode, the output state of the Timer Slices PWM signal(s) (the ones set in multichannel mode) can be controlled in parallel by a single pattern.

The pattern is controlled via the CCU8 inputs, CCU8x.MCIy[3:0]. Each group of these inputs is connected accordingly to the specific Timer Slice: for slice 0, CCU8xMCI1[3:0] for slice 1, CCU8xMCI2[3:0] for slice 2 and CCU8xMCI3[3:0] for slice 3.

This pattern can be controlled directly by one of the POSIF modules and be updated in parallel for all the Timer Slices.

Using the POSIF module in conjunction with the Multi Channel support of the CCU8, one can achieve a complete synchronicity between the output state update, CCU8x.OUTy and the update of a new pattern, **Figure 22-58**.

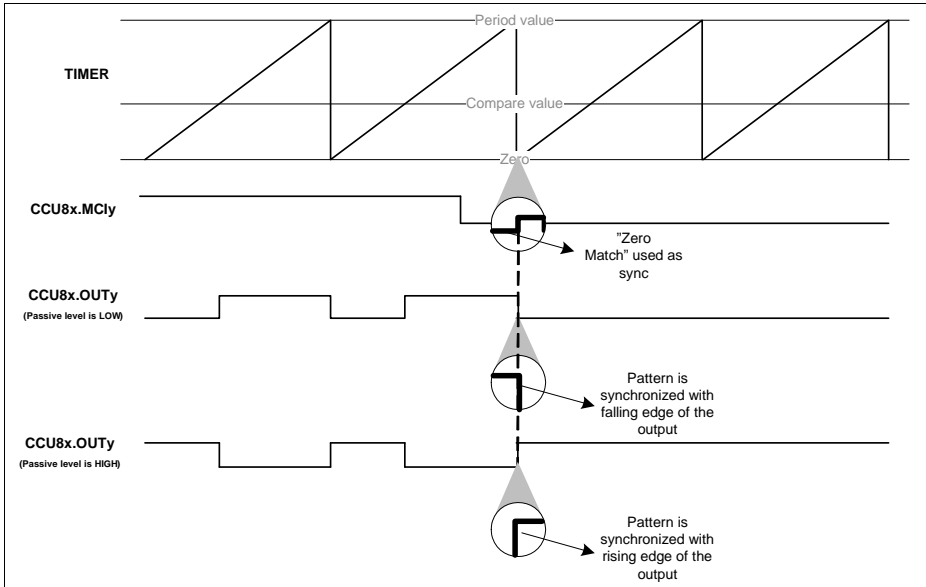


Figure 22-58 Multi channel pattern synchronization

These pattern inputs are going to be used in the output modulation control unit to put the specific PWM output into active or passive state: CCU8x.MCIy[0] has effect on the CC8yST1 path and therefore controls the CC8xOUT00 pin, CCU8x.MCIy[1] is used in the same manner for the inverted CC8yST1 path, CCU8x.MCIy[2] and CCU8x.MCIy[3] are linked to the CC8yST2 and inverted CC8yST2 path respectively. [Figure 22-59](#) shows the simplified scheme for the multi channel control.

[Figure 22-60](#), shows the usage of the multi channel mode in conjunction with two Timer Slices of the CCU8. The multi channel pattern is driven via the POSIF module, which enables a glitch free update of all the outputs of the CCU8.

Capture/Compare Unit 8 (CCU8)

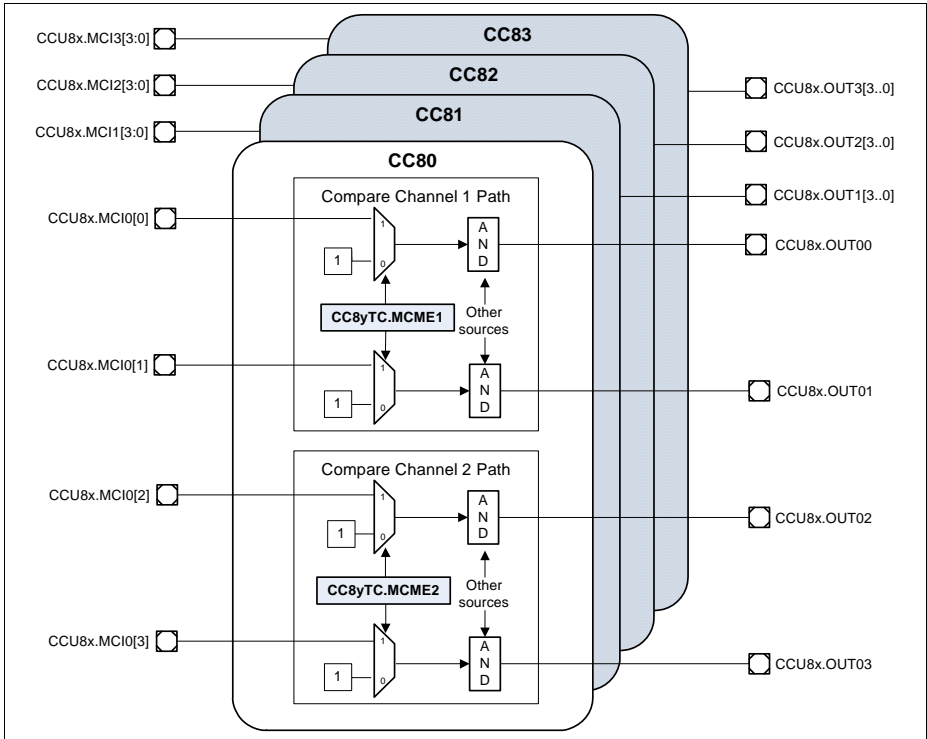


Figure 22-59 CCU8 Multi Channel overview

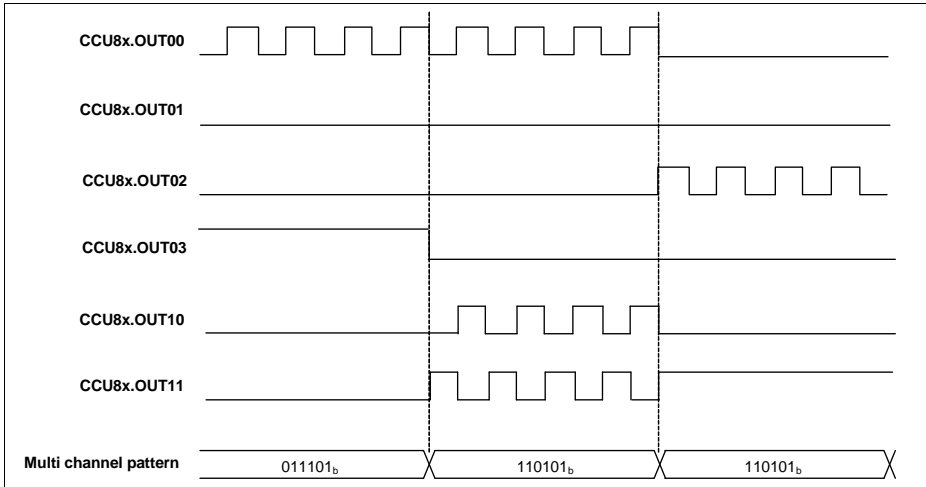


Figure 22-60 Multi Channel mode for multiple Timer Slices

The synchronization between the CCU8 and the POSIF is achieved, by adding a 3 cycle delay on the output path of each Timer Slice (between the status bit, CC8ySTx and the direct control of the output pin). This path is only selected when **CC8yTC.MCME_x = 1_B**.

On [Figure 22-61](#) the control of the CC8yST1 path is represented. The control of remaining paths follows the same mechanism (the multi channel is only enabled for the CC8yST2 path if **CC8yTC.MCME₂ = 1_B**).

The multi pattern input synchronization can be seen on [Figure 22-62](#). To achieve a synchronization between the update of the status bit, the sampling of a new multi channel pattern input is controlled by the period match or one match signal.

In a normal operation, where no external signal is used to control the counting direction, the signal used to enable the sampling of the pattern is always the period match when in edge aligned and the one match when in center aligned mode. When an external signal is used to control the counting direction, depending if the counter is counting up or counting down, the period match or the one match signal is used, respectively.

Capture/Compare Unit 8 (CCU8)

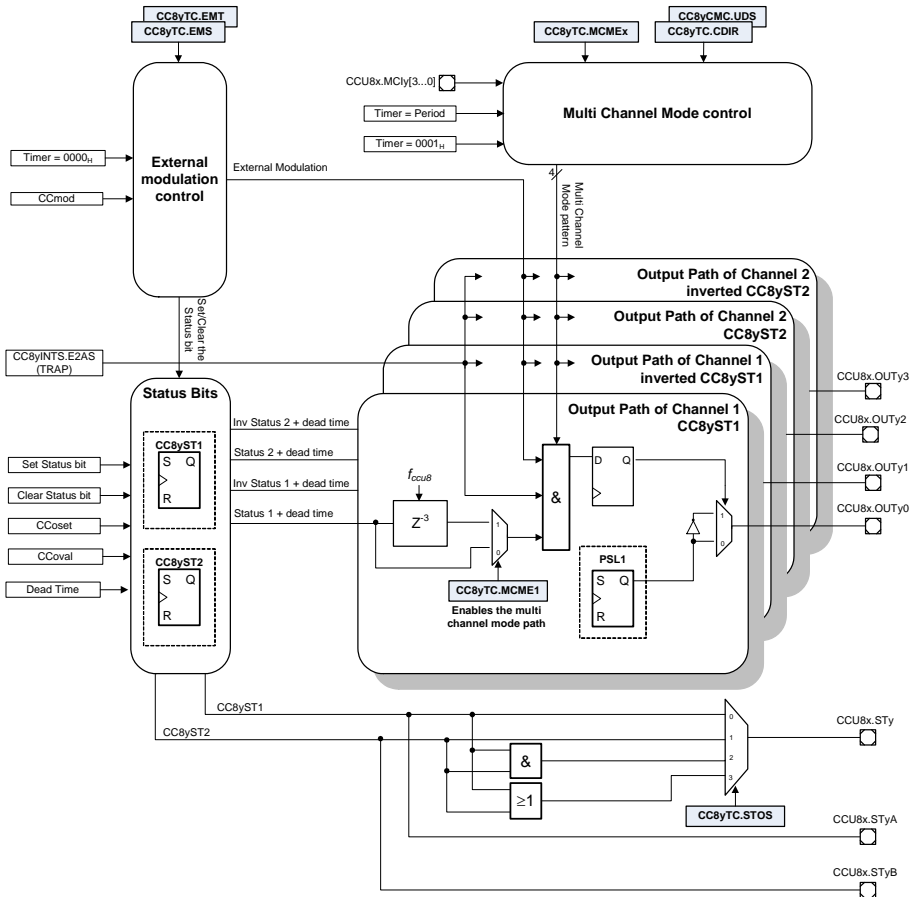


Figure 22-61 Output Control Diagram

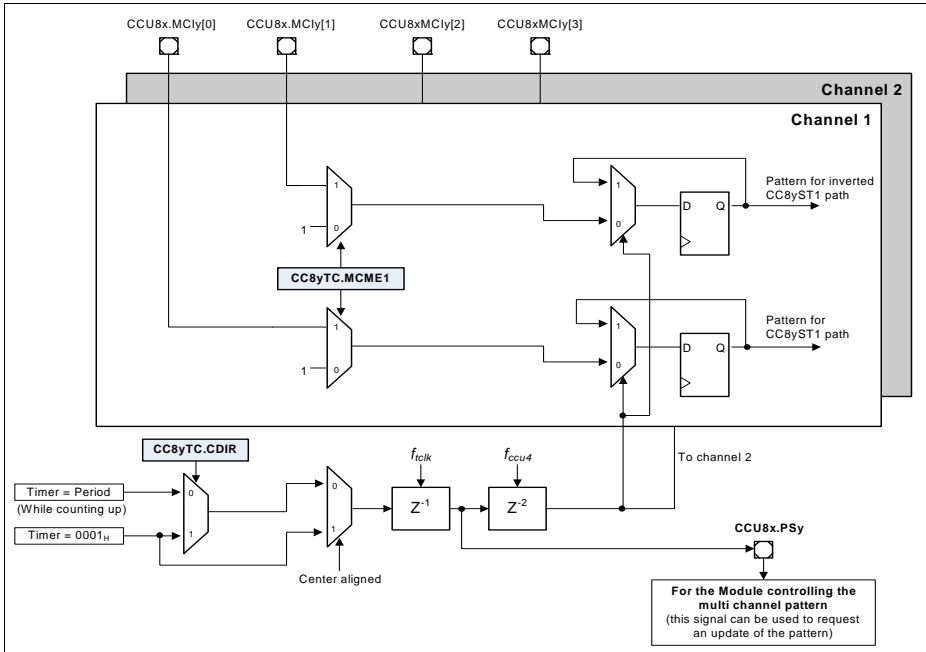


Figure 22-62 Multi Channel Pattern Synchronization Control

22.2.10 Timer Concatenation

The CCU8 offers a very easy mechanism to perform a synchronous timer concatenation. This functionality can be used by setting the **CC8yCMC.TCE** = 1_B. By doing this the user is doing a concatenation of the actual CCU8 slice with the previous one, see [Figure 22-63](#).

Notice that it is not possible to perform concatenation with non adjacent slices and that timer concatenation automatically sets the slice mode into Edge Aligned. It is not possible to perform timer concatenation in Center Aligned mode.

To enable a 64 bit timer, one should set the **CC8yCMC.TCE** = 1_B in all the slices (with the exception of the CC80 due to the fact that it doesn't contain this control field).

To enable a 48 bit timer, one should set the **CC8yCMC.TCE** = 1_B in two adjacent slices and to enable a 32 bit timer, the **CC8yCMC.TCE** is set to 1_B in the slice containing the MSBs. Notice that the timer slice containing the LSBs should always have the TCE bitfield set to 0_B.

Several combinations for timer concatenation can be made inside a CCU8 module:

- one 64 bit timer

Capture/Compare Unit 8 (CCU8)

- one 48 bit timer plus a 16 bit timer
- two 32 bit timers
- one 32 bit timer plus two 16 bit timers

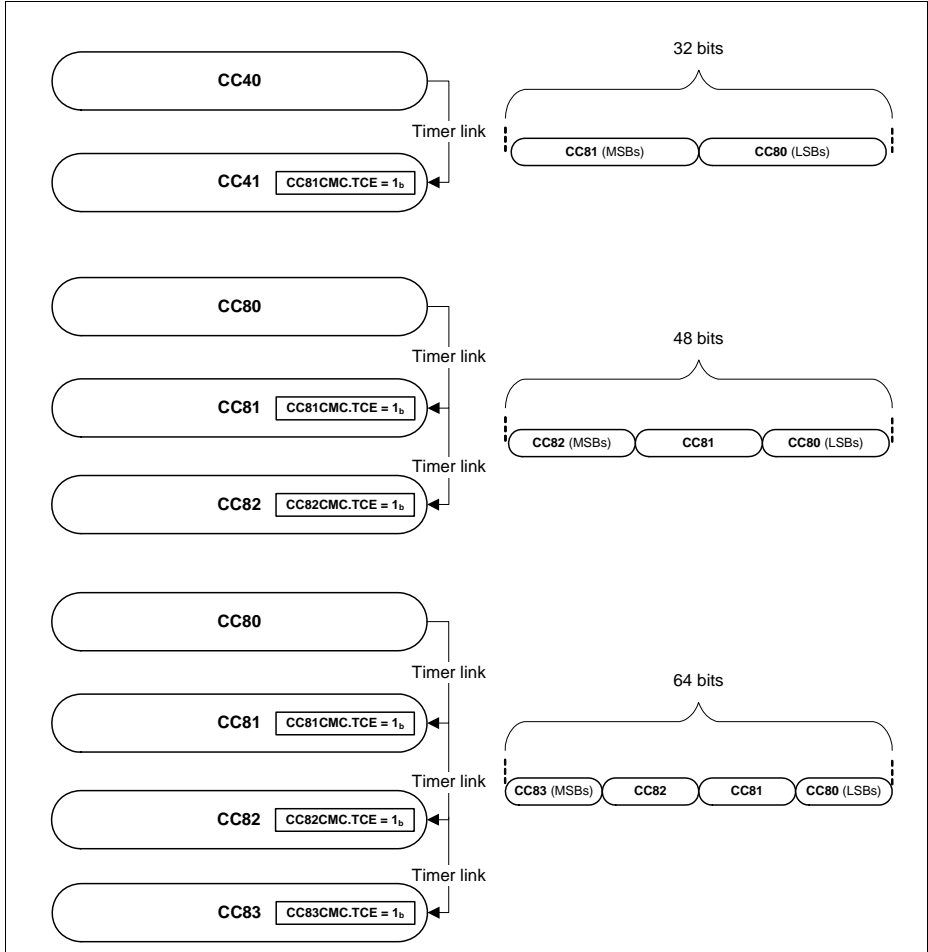


Figure 22-63 Timer concatenation example

Each Timer Slice is connected to the adjacent Timer Slices via a dedicated concatenation interface. This interface allows the concatenation of not only the Timer counting operation, but also a synchronous input trigger handling for capturing and loading operations, [Figure 22-64](#).

Note: For all the cases, CC80 and CC83 are not considered adjacent slices

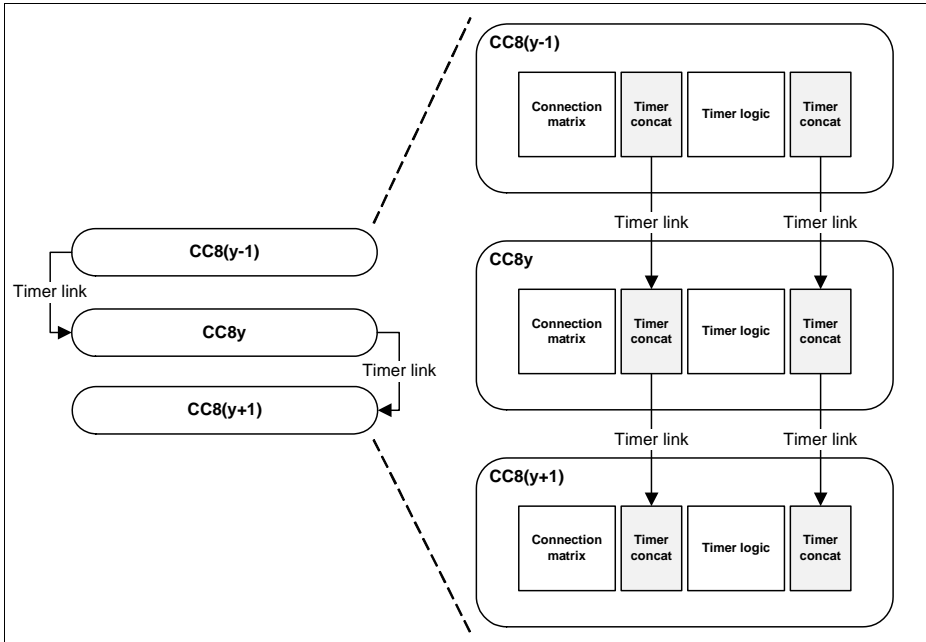


Figure 22-64 Timer concatenation link

Eight signals are present in the timer concatenation interface:

- Timer Period Match (CC8yPM)
- Timer Zero Match (CC8yZM)
- Timer Compare Match from channel 1 (CC8yCM1)
- Timer Compare Match from channel 2 (CC8yCM2)
- Timer counting direction function (CCupd)
- Timer load function (CCload)
- Timer capture function for CC8yC0V and CC8yC1V registers (CCcap0)
- Timer capture function for CC8yC2V and CC8yC3V registers (CCcap1)

The first five signals are used to perform the synchronous timing concatenation at the output of the Timer Logic, like it is seen in [Figure 22-64](#). With this link, the timer length can be easily adjusted to 32, 48 or 64 bits (counting up or counting down)

The last three signals are used to perform a synchronous link between the capture and load functions, for the concatenated timer system. This means that the user can have a capture or load function programmed in the first Timer Slice, and propagate this capture or load trigger synchronously from the LSBs until the MSBs, [Figure 22-65](#).

Capture/Compare Unit 8 (CCU8)

The capture or load function only needs to be configured in the first Timer Slice (the one holding the LSBs). From the moment that **CC8yCMC.TCE** is set to 1_B, in the following Timer Slices, the link between these functions is done automatically by the hardware.

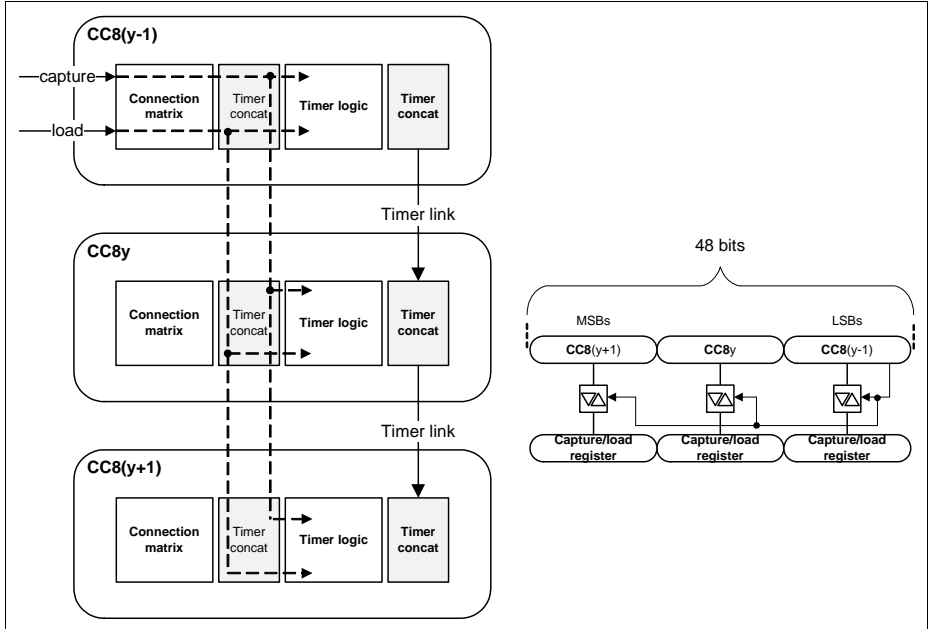


Figure 22-65 Capture/Load Timer Concatenation

the period match (CC8yPM) or zero match (CC8yZM) from the previous Timer Slice (with the immediately next lower index) are used in concatenated mode, as gating signal for the counter. This means that the counting operation of the MSBs only happens when a wrap around condition is detected (in the previous Timer Slice), avoiding additional DSP operations to extract the counting value.

With the same methodology, the compare match (CC8yCM1 and CC8yCM2), zero match and period match are gated with the specific signals from the previous Timer Slice. This means that the timing information is propagated throughout all the slices, enabling a completely synchronous match between LSB and MSB count, see [Figure 22-66](#).

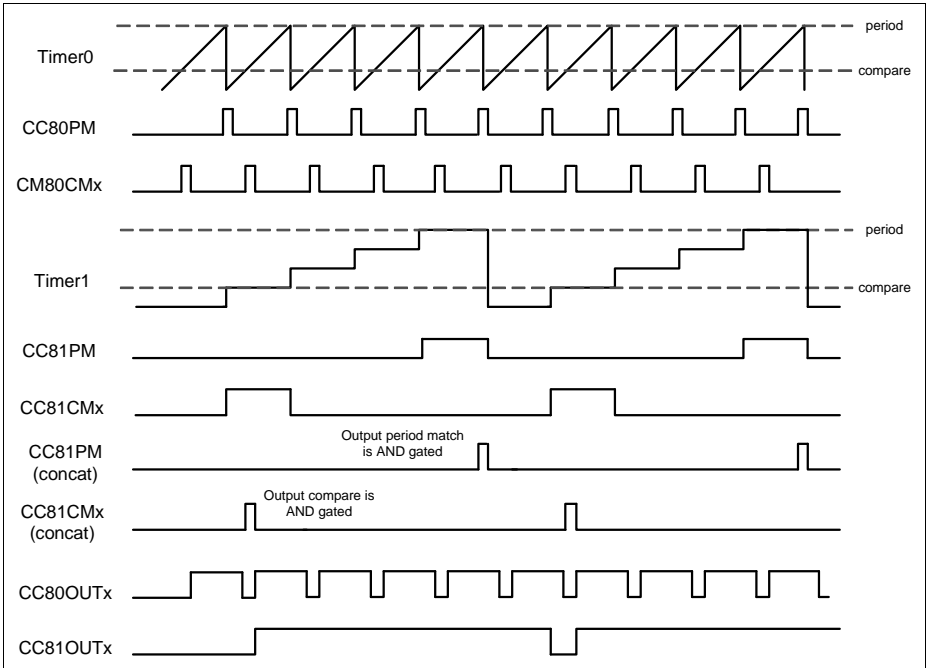


Figure 22-66 32 bit concatenation timing diagram

Note: the counting direction of the concatenated timer needs to be fixed. The timer can count up or count down, but the direction cannot be updated on the fly.

Figure 22-67 gives an overview of the timer concatenation logic. Notice that all the mechanism is controlled solely by the **CC8yCMC.TCE** bitfield.

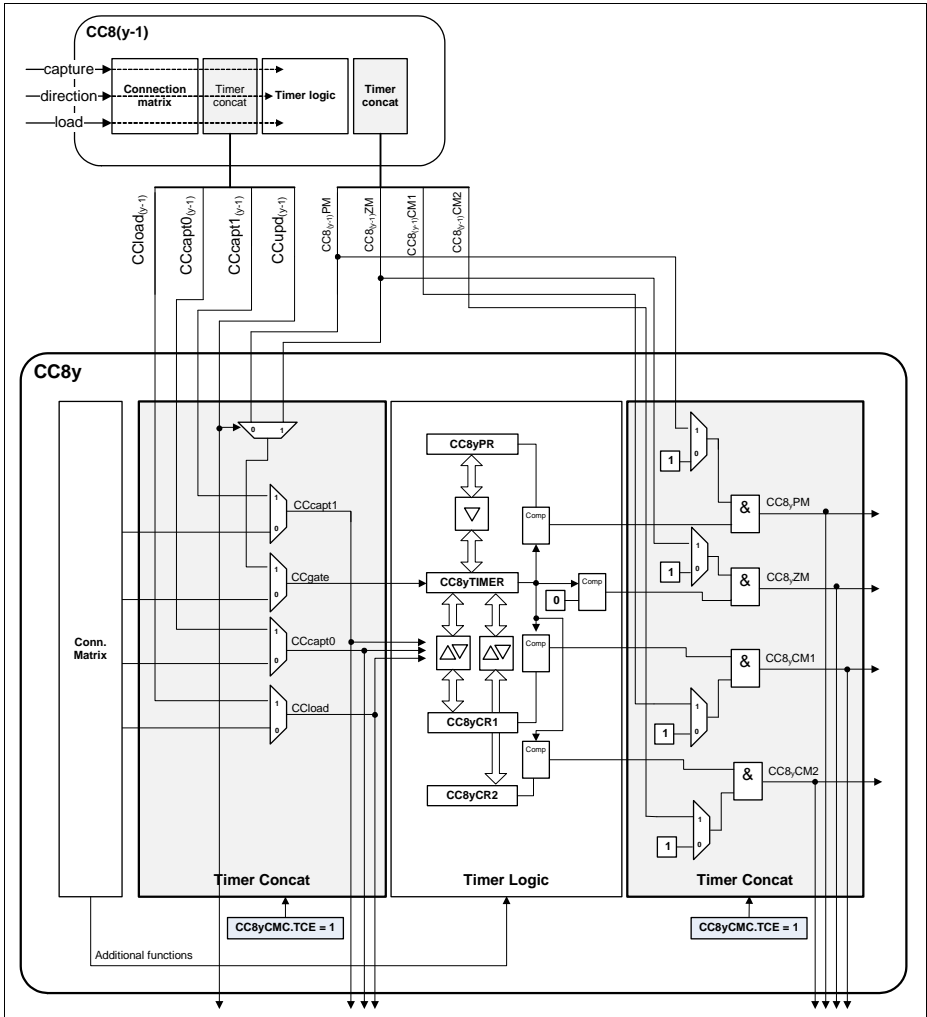


Figure 22-67 Timer concatenation control logic

22.2.11 Output Parity Checker

The parity checker function can be enabled by setting the **GIDLC.PCH** bit field to 1_B (parity checker is disabled while **GSTAT.PCRB** is 0_B).

Capture/Compare Unit 8 (CCU8)

This parity checker function, crosschecks the value at the output of the CCU8 module versus an input signal that should be connected to a driver XOR structure.

It is also possible to add a delay between the switching of the outputs and the evaluation of the input signal coming from the driver structure, and select which type of parity, even or odd (via the **GPCHK.PCTS** bit field). **Figure 22-68** shows the structure of the parity checker unit.

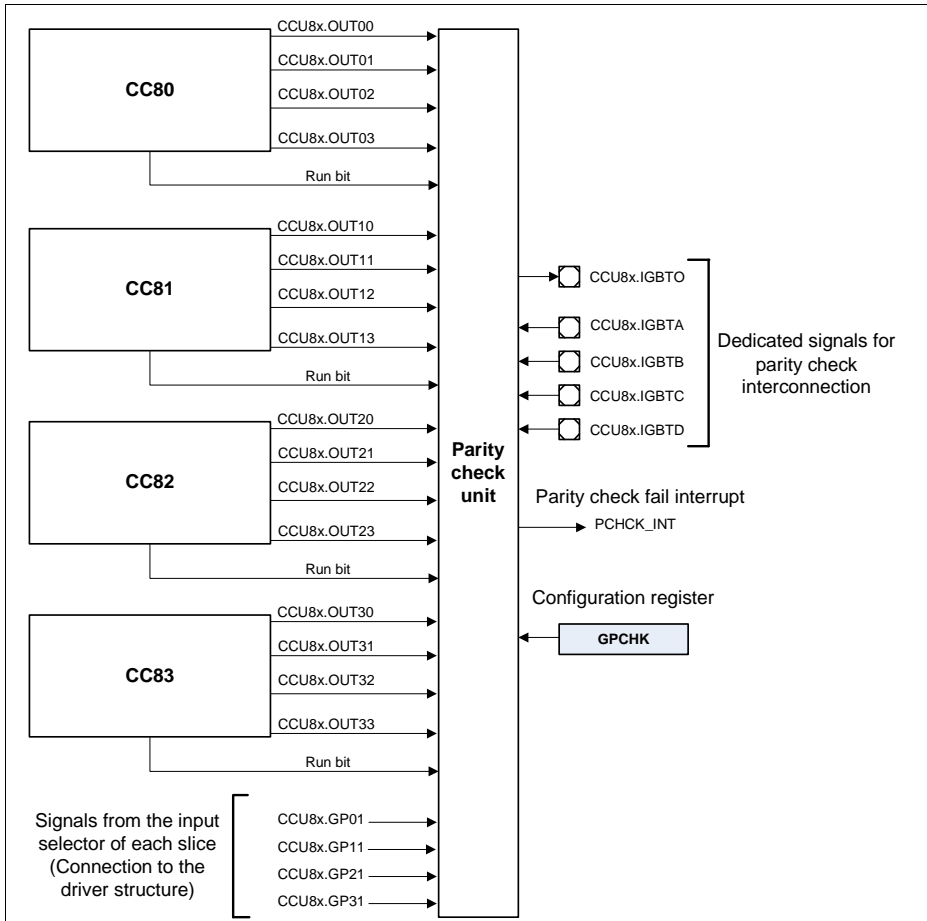


Figure 22-68 Parity checker structure

To use the parity check function, the user must select which signal is connected to the driver parity structure:

Capture/Compare Unit 8 (CCU8)

- The signal can be connected to any of the slices inputs
- The signal must be selected throughout the input selector mux of each slice. The signal must be mapped to the Event 1 of a slice.

Each of the CCU8 outputs can be individually selected to be part of the parity string and an interrupt is generated every time the input signal, coming from the driver structure, does not match the internally generated XOR result.

The interrupt is connected to the E1AS status bit of the slice where the driver parity output is connected:

- If **GPCHK.PISEL** = 00_B then the error status is in CC80INTS.E1AS
- If **GPCHK.PISEL** = 01_B then the error status is in CC81INTS.E1AS
- If **GPCHK.PISEL** = 10_B then the error status is in CC82INTS.E1AS
- If **GPCHK.PISEL** = 11_B then the error status is in CC83INTS.E1AS

The logic structure of the parity check is described on [Figure 22-69](#). For a more detailed description of resource usage for the parity checker function, please address [Section 22.2.14.5](#).

Configuration Example:

Driver parity output is connected to the input CCU8x.IN1B (where x = CCU8 unit). The input used to control the switching delay is the CCU8x.IGBTCCU8. The driver is using 12 outputs coming from the first three slices with an even parity (PCTS field is in default).

The following registers should then be programmed:

CC8yINS.EV1IS = 0001_B; selects the input CCU8x.IN1B

GPCHK.PISEL = 01_B; selects the Event 1 coming from slice 1

GPCHK.PCDS = 10_B; selects the CCU8x.IGTBC input for delay control

GPCHK.PCSEL = 0FFF_H; selects only the output signals of the first three slices for parity check

GIDLC.SPCH = 1_B; starts the parity function

When a mismatch between the driver output and the parity checker is detected, an interrupt is generated on Timer Slice 1. The interrupt status bit that stores the information is **CC8yINTS.E1AS**.

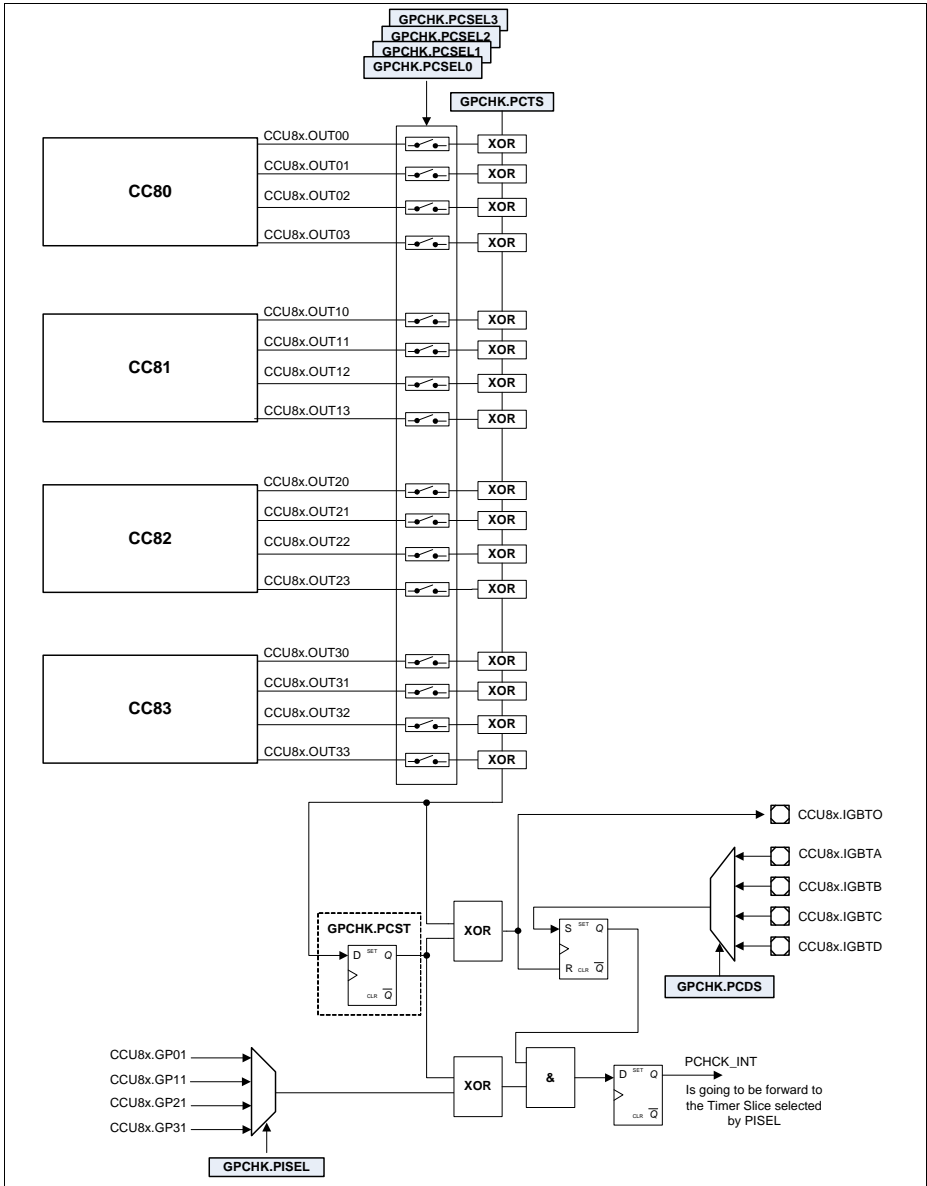


Figure 22-69 Parity checker logic

22.2.12 PWM Dithering

The CCU8 has an automatic PWM dithering insertion function. This functionality can be used with very slow control loops that cannot update the period/compare values in a fast manner, and by that fact the loop can lose precision on long runs. By introducing dither on the PWM signal, the average frequency/duty cycle is then compensated against that error.

Each slice contains a dither control unit, see [Figure 22-70](#).

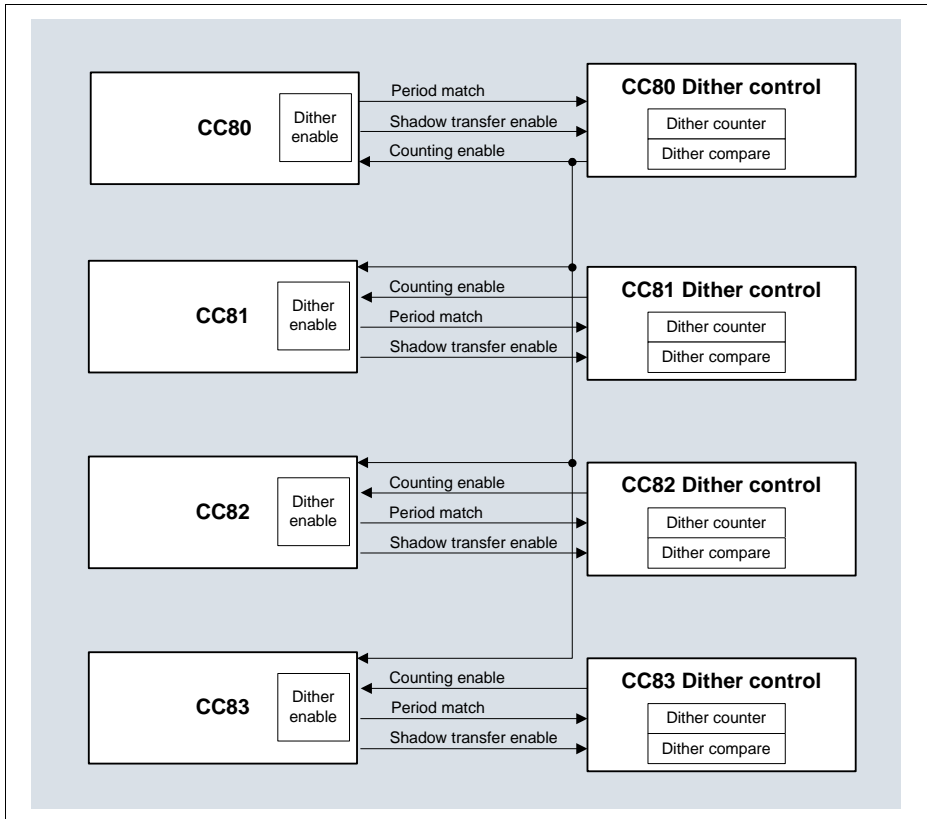


Figure 22-70 Dither structure overview

The dither control unit contains a 4 bit counter and a compare value. The four bit counter is incremented every time that a period match occurs. The counter works in a bit reverse mode so the distribution of increments stays uniform over 16 counter periods, see [Table 22-6](#).

Table 22-6 Dither bit reverse counter

counter[3]	counter[2]	counter[1]	counter[0]
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	0
0	0	0	1
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	1

The counter is then compared against a programmed value, **CC8yDIT.DCV**. If the counter value is smaller than the programmed value, a gating signal is generated that can be used to extend the period, to delay the compare or both (controlled by the **CC8yTC.DITHE** field, see **Table 22-7**) for one clock cycle.

Table 22-7 Dither modes

DITHE[1]	DITH[0]	Mode
0	0	Dither is disabled
0	1	Period is increased by 1 cycle
1	0	Compare match is delayed by 1 cycle
1	1	Period is increased by 1 cycle and compare is delayed by 1 cycle

The dither compare value also has an associated shadow register that enables concurrent update with the period/compare registers of each CC8y. The control logic for the dithering unit is represented on **Figure 22-71**.

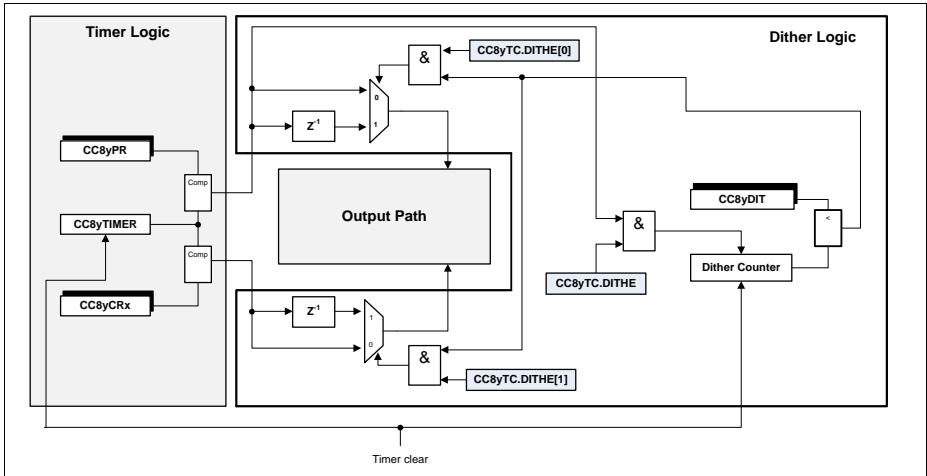


Figure 22-71 Dither control logic

Figure 22-72 to Figure 22-77 show the effect of the different configurations of the dither, **CC8yTC.DITHE**, for both counting schemes, Edge and Center Aligned mode. In each figure, the bit reverse scheme is represented for the dither counter and the compare value was programmed with the value 8_H . In each figure, the variable T , represents the period of the counter, while the variable d indicates the duty cycle (status bit is set HIGH).

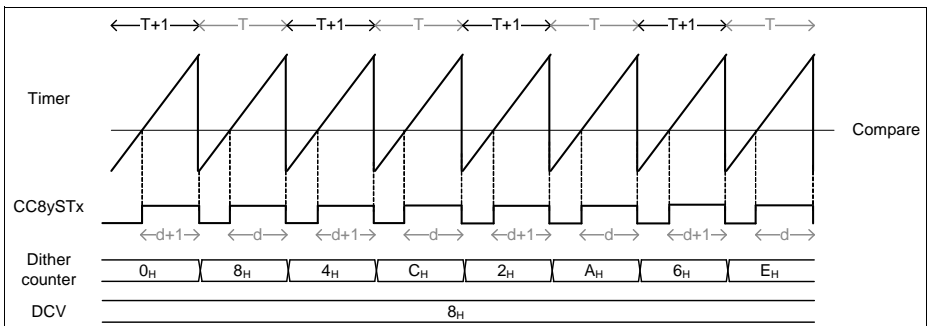


Figure 22-72 Dither timing diagram in edge aligned - **CC8yTC.DITHE = 01_B**

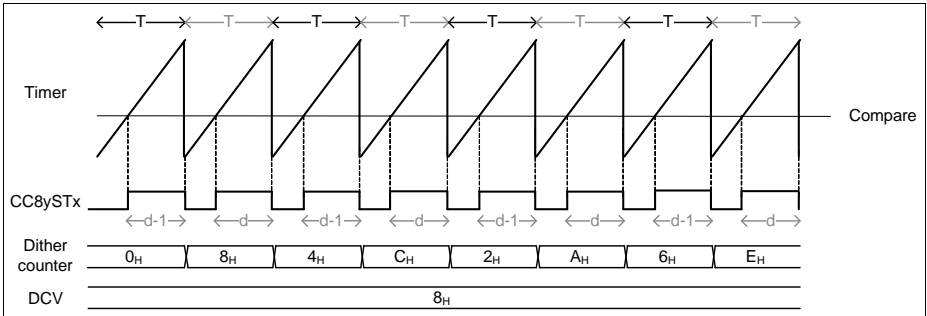


Figure 22-73 Dither timing diagram in edge aligned - $CC8yTC.DITHE = 10_B$

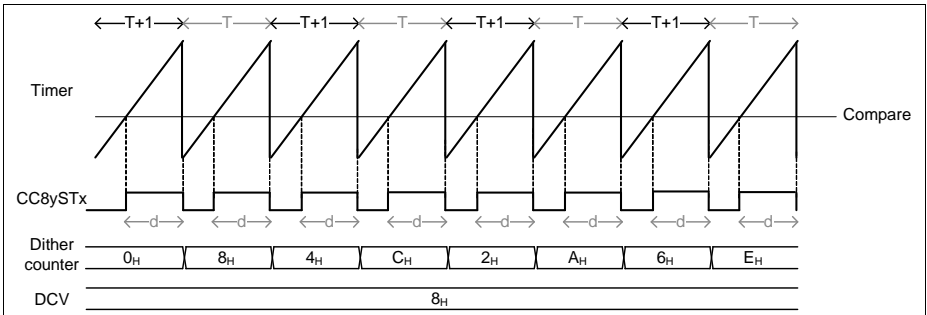


Figure 22-74 Dither timing diagram in edge aligned - $CC8yTC.DITHE = 11_B$

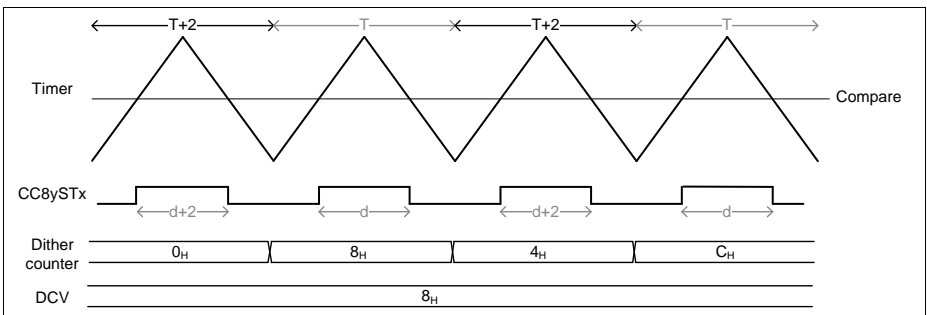


Figure 22-75 Dither timing diagram in center aligned - $CC8yTC.DITHE = 01_B$

Capture/Compare Unit 8 (CCU8)

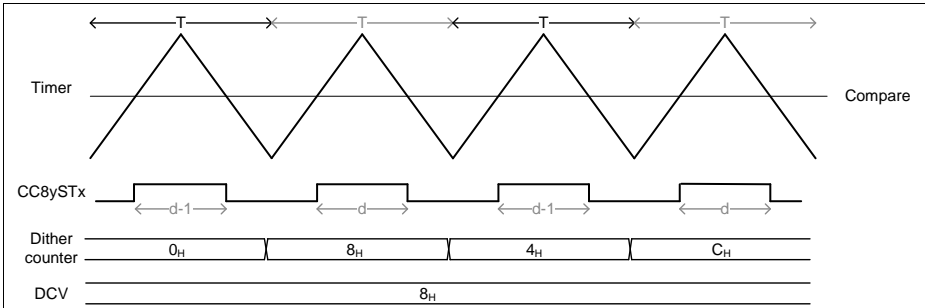


Figure 22-76 Dither timing diagram in center aligned - CC8yTC.DITHE = 10_B

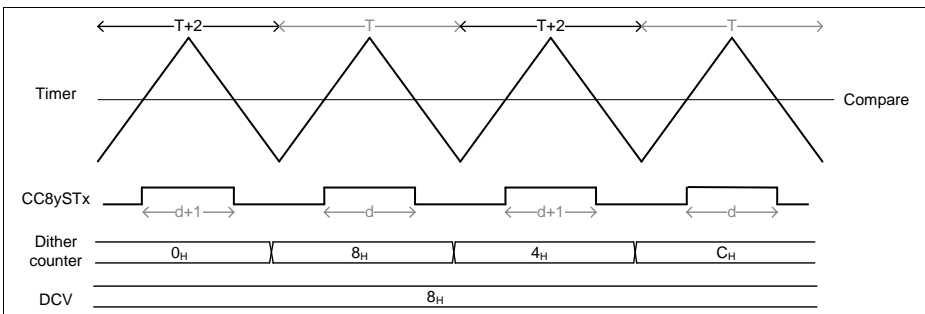


Figure 22-77 Dither timing diagram in edge aligned - CC8yTC.DITHE = 11_B

Note: When using the dither, is not possible to select a period value of FS when in edge aligned mode. In center aligned mode, the period value must be at least FS - 2.

22.2.13 Prescaler

The CCU8 contains a 4 bit prescaler that can be used in two operating modes for each individual slice:

- normal prescaler mode
- floating prescaler mode

The run bit of the prescaler can be set/cleared by SW by writing into the registers, **GIDL**.SPRB and **GIDL**.CPRB respectively or can also be cleared by the run bit of a specific slice. With the last mechanism, the run bit of the prescaler is cleared one clock cycle after the clear of the run bit of the selected sliced. To select which slice can perform this action, one should program the **GCTRL**.PRBC register.

22.2.13.1 Normal Prescaler Mode

In Normal prescaler mode the clock fed to the CC8y counter is a normal fixed division by N, accordingly to the value set in the **CC8yPSC**.PSIV register. The values for the possible division values are listed in **Table 22-8**. The **CC8yPSC**.PSIV value is only modified by a SW access. Notice that each slice has a dedicated prescaler value selector (**CC8yPSC**.PSIV), which means that the user can select different counter clocks for every and each Timer Slice (CC8y).

Table 22-8 Timer clock division options

CC8yPSC.PSIV	Resulting clock
0000 _B	f_{CCU8}
0001 _B	$f_{CCU8}/2$
0010 _B	$f_{CCU8}/4$
0011 _B	$f_{CCU8}/8$
0100 _B	$f_{CCU8}/16$
0101 _B	$f_{CCU8}/32$
0110 _B	$f_{CCU8}/64$
0111 _B	$f_{CCU8}/128$
1000 _B	$f_{CCU8}/256$
1001 _B	$f_{CCU8}/512$
1010 _B	$f_{CCU8}/1024$
1011 _B	$f_{CCU8}/2048$
1100 _B	$f_{CCU8}/4096$
1101 _B	$f_{CCU8}/8192$
1110 _B	$f_{CCU8}/16384$
1111 _B	$f_{CCU8}/32768$

22.2.13.2 Floating Prescaler Mode

The floating prescaler mode can be used individually in each slice by setting the register **CC8yTC**.FPE = 1_B. With this mode, the user can not only achieve a better precision on the counter clock for compare operations but also reduce the SW read access for the capture mode.

The floating prescaler mode contains additionally to the initial value register, **CC8yPSC**.PSIV, a compare register, **CC8yFPC**.PCMP with an associated shadow register.

Capture/Compare Unit 8 (CCU8)

Figure 22-78 shows the structure of the prescaler in floating mode when the specific slice is in compare mode (no external signal is used for capture). In this mode, the value of the clock division is incremented by 1_D every time that a timer overflow/underflow (overflow if in Edge Aligned Mode, underflow if in Center Aligned Mode) occurs.

In this mode, the Compare Match (both channels) from the timer is AND gated with the Compare Match of the prescaler and every time that this event occurs, the value of the clock division is updated with the **CC8yPSC.PSIV** value in the immediately next timer overflow/underflow event.

To use just one compare channel to control the floating prescaler, the other compare channel must be disabled. To do this, the compare value, **CC8yCR1** or **CC8yCR2** (depending on which channel is used) needs to be set with a value bigger than the period, **CC8yPR**. This means that in edge aligned mode, the maximum value for the timer period is 65534_D , because the compare value of one channel needs to be set to 65535_D .

The shadow transfer of the floating prescaler compare value, **CC8yFPC.PCMP**, is done following the same rules described on **Section 22.2.5.2**.

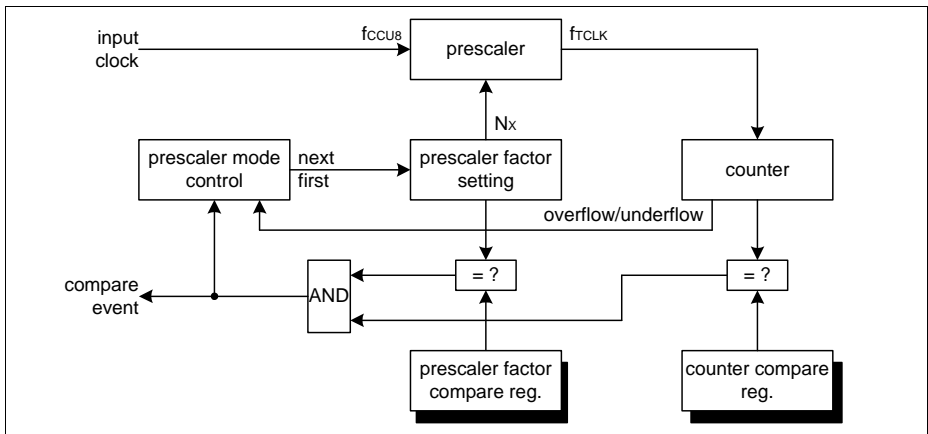


Figure 22-78 Floating prescaler in compare mode overview

When the specific CCU8 is operating in capture mode (when at least one external signal is decoded as capture functionality), the actual value of the clock division also needs to be stored every time that a capture event occurs. The floating prescaler can have up to 4 capture registers (the maximum number of capture registers is dictated by the number of capture registers used in the specific slice).

The clock division value continues to be incremented by 1 every time that a timer overflow (in capture mode, the slice is always operating in Edge Aligned Mode) occurs and it is loaded with the PSIV value every time that a capture triggers is detected.

Capture/Compare Unit 8 (CCU8)

See the [Section 22.2.14](#) for a full description of the usage of the floating prescaler mode in conjunction with compare and capture modes.

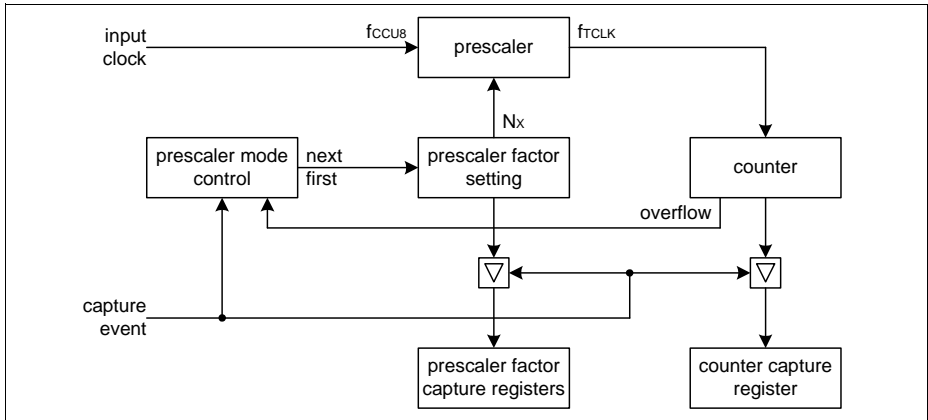


Figure 22-79 Floating Prescaler in capture mode overview

22.2.14 CCU8 Usage

22.2.14.1 PWM Signal Generation

The CCU8 offers a very flexible range in duty cycle configurations. This range is comprised between 0 to 100%.

To generate a PWM signal with a 100% duty cycle in Edge Aligned Mode, one should program the compare value, **CC8yCR1.CR1/CC8yCR2.CR2**, to 0000_H, [Figure 22-80](#).

In the same manner a 100% duty cycle signal can be generated in Center Aligned Mode, [Figure 22-81](#).

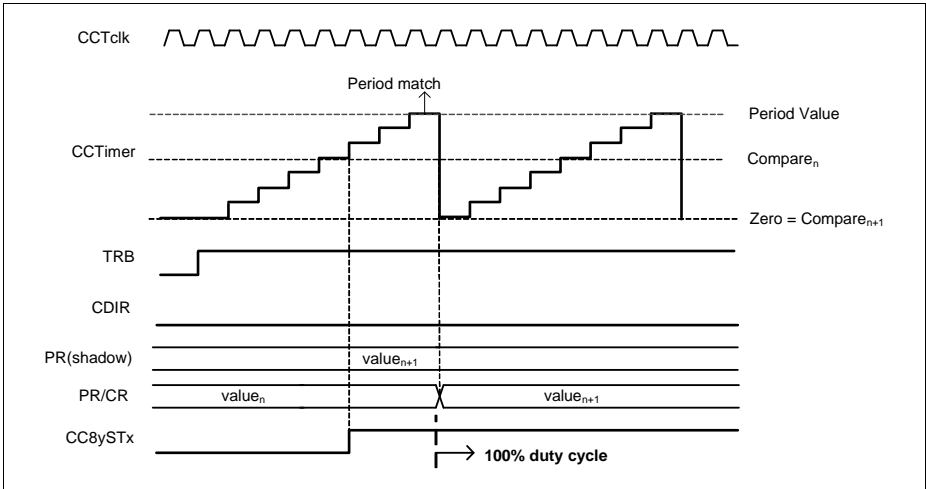


Figure 22-80 PWM with 100% duty cycle - Edge Aligned Mode

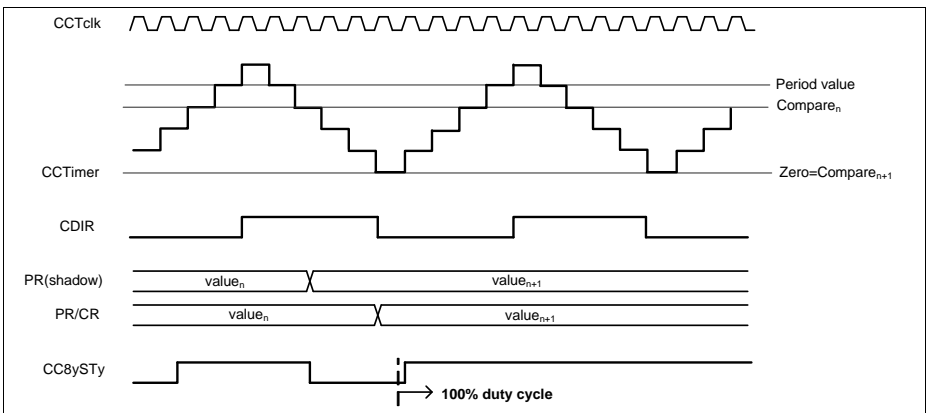


Figure 22-81 PWM with 100% duty cycle - Center Aligned Mode

To generate a PWM signal with 0% duty cycle in Edge Aligned Mode, the compare register should be set with the value programmed into the period value plus 1. In the case that the timer is being used with the full 16 bit capability (counting from 0 to 65535), setting a value bigger than the period value into the compare register is not possible and therefore the smallest duty cycle that can be achieved is 1/FS, see [Figure 22-82](#).

In Center Aligned Mode, the counter is never running from 0_D to 65535_D, due to the fact that it has to overshoot for one clock cycle the value set in the period register. Therefore

Capture/Compare Unit 8 (CCU8)

the user never has a FS counter, which means that generating a 0% duty cycle signal is always possible by setting a value in the compare register bigger than the one programmed into the period register, see [Figure 22-83](#).

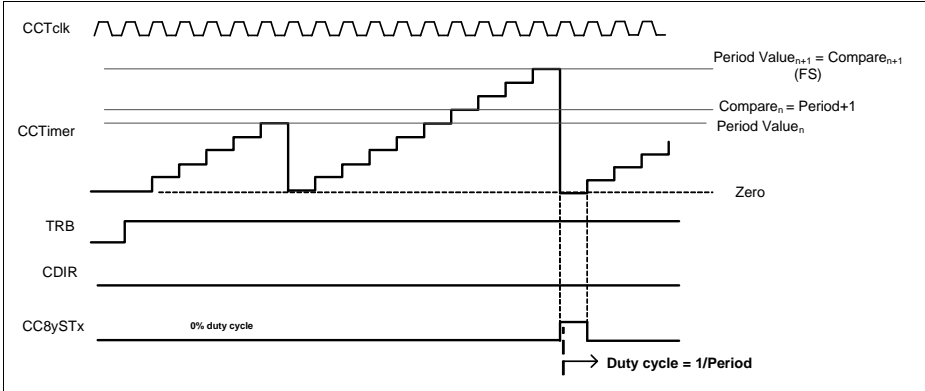


Figure 22-82 PWM with 0% duty cycle - Edge Aligned Mode

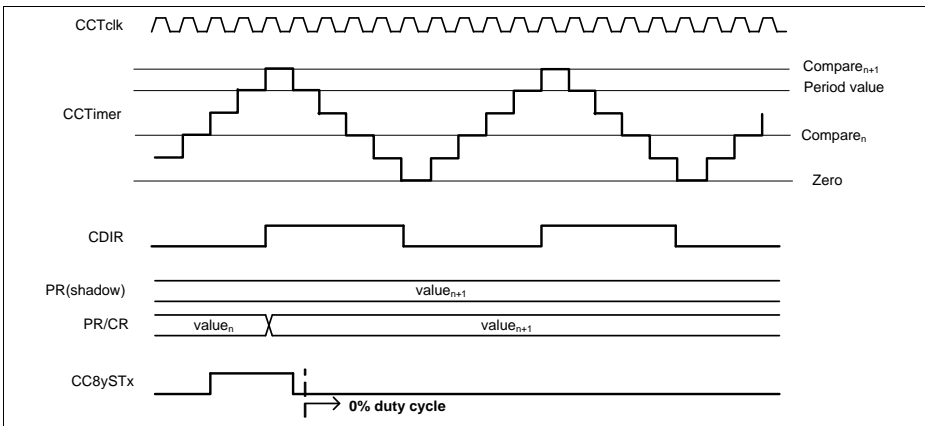


Figure 22-83 PWM with 0% duty cycle - Center Aligned Mode

22.2.14.2 Prescaler Usage

In Normal Prescaler Mode, the frequency of the f_{clk} fed to the specific CC8y is chosen from the [Table 22-8](#), by setting the `CC8yPSC.PSIV` with the required value.

In Floating Prescaler Mode, the frequency of the f_{clk} can be modified over a selected timeframe, within the values specified in [Table 22-8](#). This mechanism is specially useful if, in capture mode, the dynamic of the capture triggers is very slow or unknown.

Capture/Compare Unit 8 (CCU8)

In Capture Mode, the Floating Prescaler value is incremented by 1_D every time that a timer overflow happens and it is set with the initial programmed value when a capture event happens, see **Figure 22-84**.

When using the Floating Prescaler Mode in Capture Mode, the timer should be cleared each time that a capture event happens, **CC8yTC.CAPC** = 11_B . By operating the Capture mode in conjunction with the Floating Prescaler, even for capture signals that have a periodicity bigger than 16 bits, it is possible to use just a single CCU8 slice without monitoring the interrupt events triggered by the timer overflow. For this the user just needs to know what is the timer capture value and the actual prescaler configuration at the time that the capture event occurred. These values are contained in each CC8yCxV register.

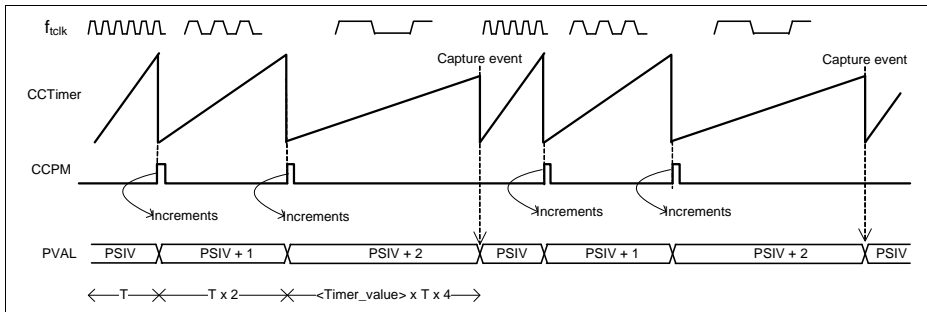


Figure 22-84 Floating Prescaler capture mode usage

When used in Compare Mode, the Floating Prescaler function may be used to achieve a fractional PWM frequency or to perform some frequency modulation.

The same incrementing by 1_D mechanism is done every time that an overflow/underflow of the Timer occurs (from any of the compare channels) and the actual Prescaler value doesn't match the one programmed into the **CC8yFPC.PCMP** register.

When a Compare Match from the Timer (from any of the compare channels) occurs and the actual Prescaler value is equal to the one programmed on the **CC8yFPC.PCMP** register, then the Prescaler value is set with the initial value, **CC8yPSC.PSIV**, in the immediately next occurrence of a timer overflow/underflow.

To use just one compare channel to control the floating prescaler, the other compare channel must be disabled. To do this, the compare value, **CC8yCR1** or **CC8yCR2** (depending on which channel is used) needs to be set with a value bigger than the period, **CC8yPR**. This means that in edge aligned mode, the maximum value for the timer period is 65534_D , because the compare value of one channel needs to be set to 65535_D (so the compare match of the associated channel is disabled).

In **Figure 22-85**, the Compare value of the Floating Prescaler was set to **PSIV + 2**. Every time that a timer overflow occurs, the value of the Prescaler is incremented by 1, which

Capture/Compare Unit 8 (CCU8)

means that if we give f_{tclk} as the reference frequency for the **CC8yPSC**.PSIV value, we have $f_{\text{tclk}}/2$ for **CC8yPSC**.PSIV + 1 and $f_{\text{tclk}}/4$ for **CC8yPSC**.PSIV + 2. With the period overtime of the counter becomes:

$$\text{Period} = (1/f_{\text{tclk}} + 2/f_{\text{tclk}} + 4/f_{\text{tclk}})/3$$

The same mechanism is used in Center Aligned Mode, but to keep the rising arcade and falling arcade always symmetrical, instead of the overflow of the timer, the underflow of the timer, see [Figure 22-86](#).

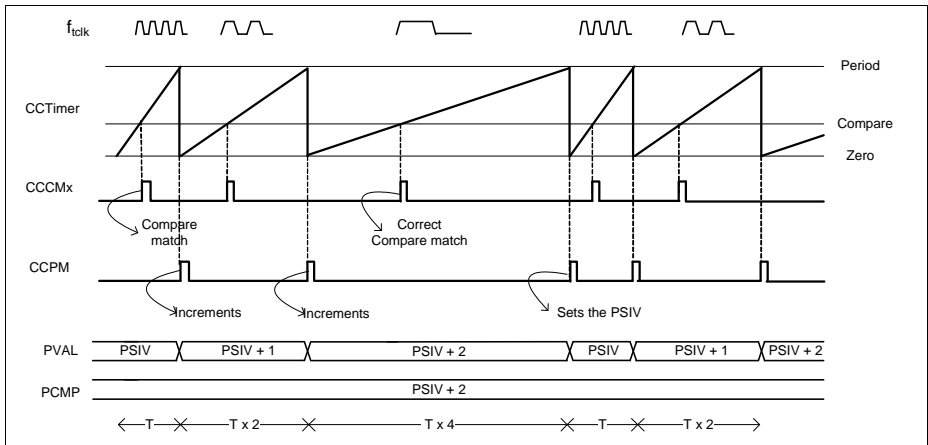


Figure 22-85 Floating Prescaler compare mode usage - Edge Aligned

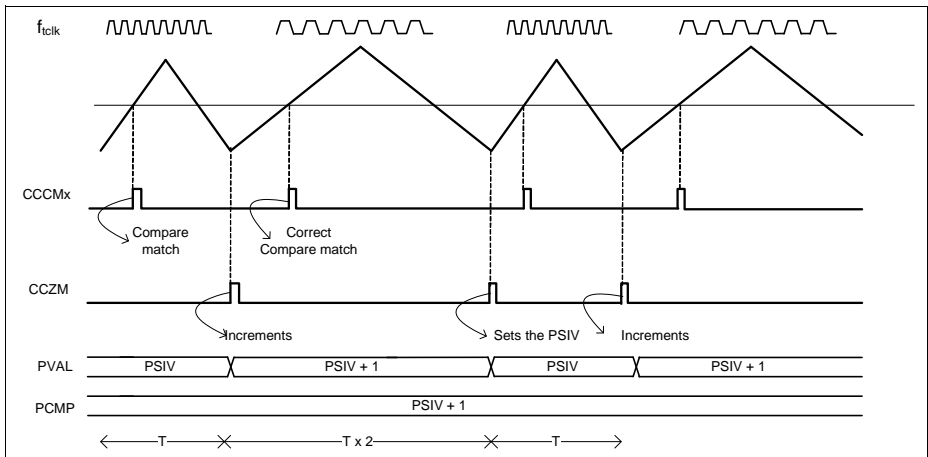


Figure 22-86 Floating Prescaler compare mode usage - Center Aligned

22.2.14.3 PWM Dither

The Dither function can be used to achieve a very fine precision on the periodicity of the output state in compare mode. The value set in the dither compare register, **CC8yDIT.DCV** is crosschecked against the actual value of the dither counter and every time that the dither counter is smaller than the comparison value one of the follows actions is taken:

- The period is extended for 1 clock cycle - **CC8yTC.DITHE** = 01_B; in edge aligned mode
- The period is extended for 2 clock cycles - **CC8yTC.DITHE** = 01_B; in center aligned mode
- The comparison match while counting up (**CC8yTCST.CDIR** = 0_B) is delayed (this means that the status bit is going to stay in the SET state 1 cycle less) for 1 clock cycle - **CC8yTC.DITHE** = 10_B;
- The period is extended for 1 clock cycle and the comparison match while counting up is delayed for 1 clock cycle - **CC8yTC.DITHE** = 11_B; in edge aligned mode
- The period is extended for 2 clock cycles and the comparison match while counting up is delayed for 1 clock cycle; center aligned mode

The bit reverse counter distributes the number programmed in the **CC8yDIT.DCV** throughout 16 timer periods.

Table 22-9, describes the bit reverse distribution versus the programmed value on the **CC8yDIT.DCV** field. The fields marked as '0' indicate that in that counter period, one of the above described actions, is going to be performed.

Table 22-9 Bit reverse distribution

Dither counter	DCV															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
4	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
C	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
2	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
6	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
E	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0

Table 22-9 Bit reverse distribution (cont'd)

Dither counter	DCV															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
3	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
B	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
7	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
F	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The bit reverse distribution versus the programmed **CC8yDIT.DCV** value results in the following values for the Period and duty cycle:

$$\text{DITHE} = 01_{\text{B}}$$

$$\text{Period} = [(16 - \text{DCV}) \times T + \text{DCV} \times (T + 1)]/16; \text{ in Edge Aligned Mode} \quad (22.10)$$

$$\text{Duty cycle} = [(16 - \text{DCV}) \times d/T + \text{DCV} \times (d+1)/(T + 1)]/16; \text{ in Edge Aligned Mode} \quad (22.11)$$

$$\text{Period} = [(16 - \text{DCV}) \times T + \text{DCV} \times (T + 2)]/16; \text{ in Center Aligned Mode} \quad (22.12)$$

$$\text{Duty cycle} = [(16 - \text{DCV}) \times d/T + \text{DCV} \times (d+2)/(T + 2)]/16; \text{ in Center Aligned Mode} \quad (22.13)$$

$$\text{DITHE} = 10_{\text{B}}$$

$$\text{Period} = T ; \text{ in Edge Aligned Mode} \quad (22.14)$$

$$\text{Duty cycle} = [(16 - \text{DCV}) \times d/T + \text{DCV} \times (d-1)/T]/16; \text{ in Edge Aligned Mode} \quad (22.15)$$

$$\text{Period} = T ; \text{ in Center Aligned Mode} \quad (22.16)$$

$$\text{Duty cycle} = [(16 - \text{DCV}) \times d/T + \text{DCV} \times (d-1)/T]/16; \text{ in Center Aligned Mode} \quad (22.17)$$

$$\text{DITHE} = 11_{\text{B}}$$

$$\text{Period} = [(16 - \text{DCV}) \times T + \text{DCV} \times (T + 1)]/16; \text{ in Edge Aligned Mode} \quad (22.18)$$

$$\text{Duty cycle} = [(16 - \text{DCV}) \times d/T + \text{DCV} \times d/(T + 1)]/16; \text{ in Edge Aligned Mode} \quad (22.19)$$

$$Period = [(16 - DCV) \times T + DCV \times (T + 2)]/16; \text{ in Center Aligned Mode} \quad (22.20)$$

$$Duty\ cycle = [(16 - DCV) \times d/T + DCV \times (d+1)/(T + 2)]/16; \text{ in Center Aligned Mode} \quad (22.21)$$

where:

T - Original period of the signal, see [Section 22.2.5.1](#)

d - Original duty cycle of the signal, see [Section 22.2.5.1](#)

22.2.14.4 Capture Mode Usage

Each slice has the possibility of using 2 or 4 capture registers. Using only 2 capture registers means that only 1 Event was linked to a captured trigger. To use the four capture registers, both capture triggers need to be mapped into an Event (it can be the same signal with different edges selected or two different signals) or the **CC8yTC.SCE** field needs to be set to 1_B, which enables the linking of the 4 capture registers.

The internal slice mechanism for capturing is the same for the capture trigger 1 or capture trigger 0.

Different Capture Events - SCE = 0_B

Capture trigger 1 (CCcapt1) is appointed to the capture register 2, **CC8yC2V** and capture register 3, **CC8yC3V**, while trigger 0 is appointed to capture register 1, **CC8yC1V** and 0, **CC8yC0V**.

In each CCcapt0 event, the timer value is stored into **CC8yC1V** and the value of the **CC8yC1V** is transferred into the **CC8yC0V**.

In each CCcapt1 event, the timer value is stored into capture register **CC8yC3V** and the value of the capture register **CC8yC3V** is transferred into **CC8yC2V**.

The previous capture/transfer mechanism only happens if the specific register is not full. A capture register becomes full when receives a new value and becomes empty after the SW has read back the value.

The full flag is cleared every time that the SW reads back the **CC8yC0V**, **CC8yC1V**, **CC8yC2V** or **CC8yC3V** register. The SW can be informed of a new capture trigger by enabling the interrupt source linked to the specific Event. This means that every time that a capture is made an interrupt pulse is generated.

In the case that the Floating Prescaler Mode is being used, the actual value of the clock division is also stored in the capture register (CC8yCxV).

Figure 22-87 shows an example of how the capture/transfer may be used in a Timer Slice that is using an external signal as count function (to measure the velocity of a rotating device), and an equidistant capture trigger that is used to dictate the timestamp

Capture/Compare Unit 8 (CCU8)

for the velocity calculation (two Timer waveforms are plotted, one that exemplifies the clearing of the timer in each capture event and another without the clearing function active).

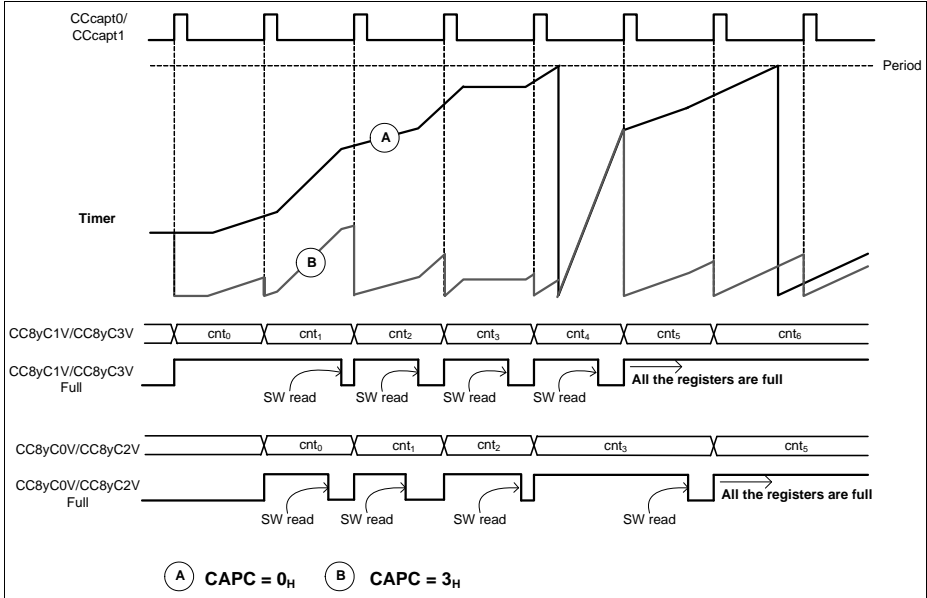


Figure 22-87 Capture mode usage - single channel

Same Capture Event - SCE = 1_B

If **CC8yTC**.SCE is set to 1_B, all the four capture registers are chained together, emulating a fifo with a depth of 4. In this case, only the capture trigger 1, CCcapt1, is used to perform a capture event.

As an example for this mode, one can consider the case where one Timer Slice is being used in capture mode with SCE = 1_B, with another external signal that controls the counting. This timer slice can be incremented at different speeds, depending on the frequency of the counting signal.

An additional Timer Slice is used to control the capture trigger, dictating the time stamp for the capturing.

A simple scheme for this can be seen in [Figure 22-88](#). The CC80ST output of slice 0 was used as capture trigger in the CC81 slice (active on rising and falling edge). The CC80ST output is used as known timebase marker, while the slice timer used for capture is being controlled by external events, e.g. external count.

Capture/Compare Unit 8 (CCU8)

Due to the fact that we have 4 capture registers available, every time that the SW reads back the complete set of values, 3 speed profiles can be measured.

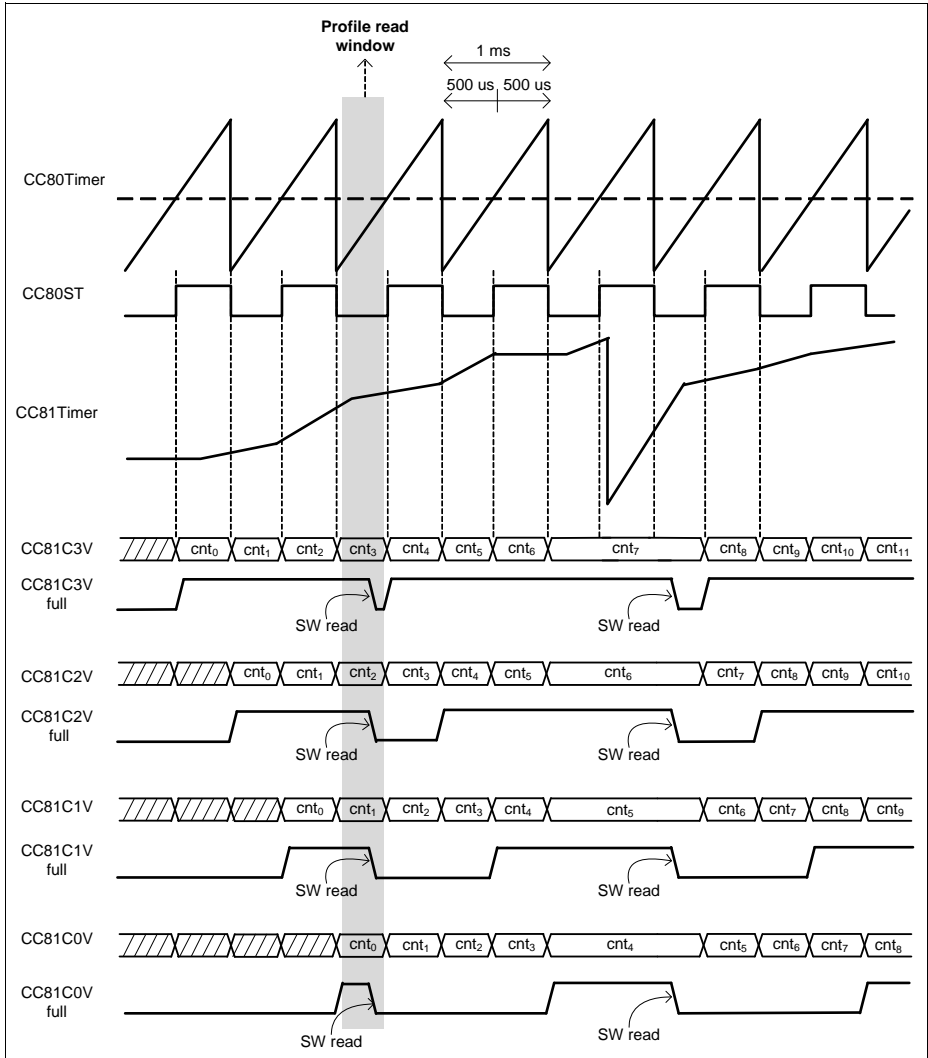


Figure 22-88 Three Capture profiles - $CC8yTC.SCE = 1_B$

To calculate the three different profiles in [Figure 22-88](#), the 4 capture registers need to be read during the pointed read window. After that, the profile calculation is done:

Profile 1 = $CC81C1V_{info} - CC81C0V_{info}$

Profile 2 = $CC81C2V_{info} - CC81C1V_{info}$

Profile 3 = $CC81C3V_{info} - CC81C2V_{info}$

Note: This is an example and therefore several Timer Slice configurations and software loops can be implemented.

High Dynamics Capturing

In some cases the dynamics of the capture trigger(s) may vary greatly over time. This will impose that the software needs to be prepared for the worst case scenario, where the frequency of the capture triggers may be very high. In applications where cycle-by-cycle calculation is needed (calculation in each capture trigger), then this constraints needs to be met by the software. Nevertheless for applications where a cycle-by-cycle calculation is not needed, the software can read back the FIFO data register in a periodic base and fetch all the data that has been captured so far, [Figure 22-89](#).

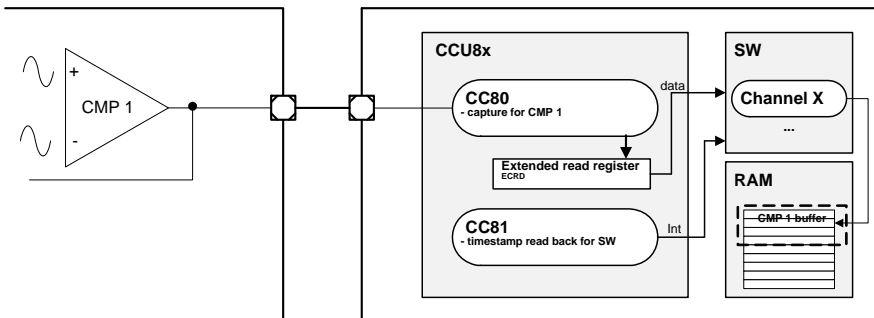


Figure 22-89 High dynamics capturing with software controlled timestamp

In this scenario, the software will read back the complete set of capture registers (2 or 4 depending on the chosen configuration), every time that an interrupt is triggered from the timestamp timer (the periodicity of this timer can also be adjusted on-the-fly).

Due to the fact that every capture register offers a full flag status bit, the software can always read back the complete set of registers. At the time of the data processing, this full flag is then checked, indicating if this value needs to be processed or not.

This FIFO read back functionality can also be used for applications that impose a heavy load on the system, which may not guarantee fixed access times to read back the captured data.

Capture/Compare Unit 8 (CCU8)

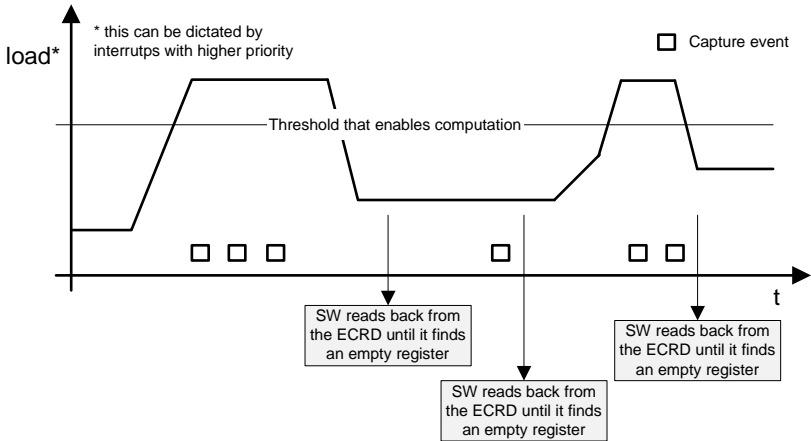


Figure 22-90 Extended read back during high load

Capture Grouping

In applications where multiple capture Timers are needed and the priority of the capture routines, does not imply that a cycle-by-cycle calculation needs to be done for every event, it may be suitable to group all the timers in the same CCU4x unit, [Figure 22-91](#).

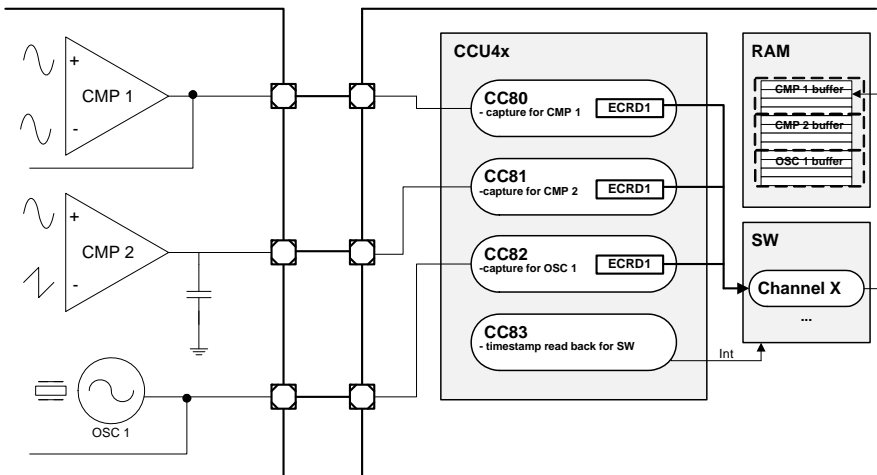


Figure 22-91 Capture grouping with extended read back

Capture/Compare Unit 8 (CCU8)

By setting the ECM bitfield for the Timer Slices used for capturing, the extended read back mode enables the reading back of data always in the proper capture order (from oldest to newest data). A timestamp timer is then used to trigger the CPU/Software to read back all the capture data present in the Timer Slices.

Every time that the interrupt is sensed, the CPU/Software (in this example) reads back the complete set of capture registers (via the ECRD address) for all Timer Slices. Due to the fact that each data read has a full flag indicator, the CPU/Software can read back the complete set of capture registers from all timers. This allows a fixed memory allocation that is as big as the number of captured registers, **Figure 22-92** (in this example 4 capture registers for each Timer Slice are being used).

The additional lost value bitfield (LCV) on the header of each ECRD data, will indicate if any capture trigger was lost between read operations (this can happen if the capture triggers are faster than the routine that is reading back the values).

Capture/Compare Unit 8 (CCU8)

	1st Read Back		2nd Read Back		2nd Read Back	
MEMORY	Timer 2 previous data	Previous	Timer 2 previous data	Previous	Timer 2 previous data	Previous
	Timer 2 previous data	Previous	Timer 2 previous data	Previous	Timer 2 previous data	Previous
	Timer 2 previous data	Previous	Timer 2 previous data	Previous	Timer 2 previous data	Previous
	Timer 2 previous data	Previous	Timer 2 previous data	Previous	Timer 2 previous data	Previous
	Timer 1 previous data	Previous	Timer 1 previous data	Previous	Timer 1 previous data	Previous
	Timer 1 previous data	Previous	Timer 1 previous data	Previous	Timer 1 previous data	Previous
	Timer 1 previous data	Previous	Timer 1 previous data	Previous	Timer 1 previous data	Previous
	Timer 1 previous data	Previous	Timer 1 previous data	Previous	Timer 1 previous data	Previous
	Timer 0 previous data	Previous	Timer 0 previous data	Previous	Timer 0 previous data	Previous
	Timer 0 previous data	Previous	Timer 0 previous data	Empty	Timer 0	xxxx_H Empty
	Timer 0 previous data	Previous	Timer 0	5888_H Full	Timer 0	5888_H Full
	Timer 0	5790_H Full	Timer 0	5790_H Full	Timer 0	5790_H Full
	...					
	10th Read Back		11th Read Back		12th Read Back	
	Timer 2 previous data	Previous	Timer 2 previous data	Previous	Timer 2	xxxx_H Empty
	Timer 2 previous data	Previous	Timer 2	xxxx_H Empty	Timer 2	xxxx_H Empty
	Timer 2	xxxx_H Empty	Timer 2	xxxx_H Empty	Timer 2	xxxx_H Empty
	Timer 2	0009_H Full	Timer 2	0009_H Full	Timer 2	0009_H Full
	Timer 1	0FCC_H Full	Timer 1	0FCC_H Full	Timer 1	0FCC_H Full
	Timer 1	0FC0_H Full	Timer 1	0FC0_H Full	Timer 1	0FC0_H Full
	Timer 1	0F09_H Full	Timer 1	0F09_H Full	Timer 1	0F09_H Full
	Timer 1	0EFF_H Full	Timer 1	0EFF_H Full	Timer 1	0EFF_H Full
	Timer 0	xxxx_H Empty	Timer 0	xxxx_H Empty	Timer 0	xxxx_H Empty
	Timer 0	xxxx_H Empty	Timer 0	xxxx_H Empty	Timer 0	xxxx_H Empty
	Timer 0	5888_H Full	Timer 0	5888_H Full	Timer 0	5888_H Full
	Timer 0	5790_H Full	Timer 0	5790_H Full	Timer 0	5790_H Full

Figure 22-92 Memory structure for extended read back

22.2.14.5 Parity Checker Usage

The parity checker function available on the CCU8 uses of one CCU4 timer slice, to control the delay between the update of the outputs and the consequent update on the external switch/driver.

Capture/Compare Unit 8 (CCU8)

This CCU4 timer slice is configured to work in edge aligned mode, with single shot mode active and with a configured external flush & start function. This function is connected to the parity checker update output signal, CCU8x.IGBT0. The timer slice compare and period values need to be programmed accordingly to the delay that is foreseen between the outputs of the CCU8 and the consequent update on the driver/switch parity output. The connections between the CCU8, CCU4 and the external HW can be seen on [Figure 22-93](#).

[Figure 22-94](#) shows the timing waveforms for an usage example of the parity checker (case of even parity, **GPCHK**.PCTS field takes the default value). In this example only two outputs of CCU8 were considered to avoid extreme complexity on the diagram.

Every time an update of the selected outputs leads to a modification of the value **GPCHK**.PCST (parity checker status), or in other words, every time the result of the XOR chain changes, a trigger is generated on the CCU8x.IGBT0. This signal, as described previously, is used as a flush & start for a CCU4 slice.

When the CCU4 slice timer reaches the compare value, the specific status output, CCU4x.STy is asserted. After this elapsed delay, the value on the CCU8x.GPy1 (the parity value coming from the external HW) is crosschecked against the result of the internal XOR chain (**GPCHK**.PCST). If the values are different, a Service Request pulse can be generated, if it was previously enabled.

Notice that when an update of the CCU8 outputs leads to an equal result on the XOR chain, the CCU4 slice is not retriggered (the output parity from the external hardware remains with the same value).

Capture/Compare Unit 8 (CCU8)

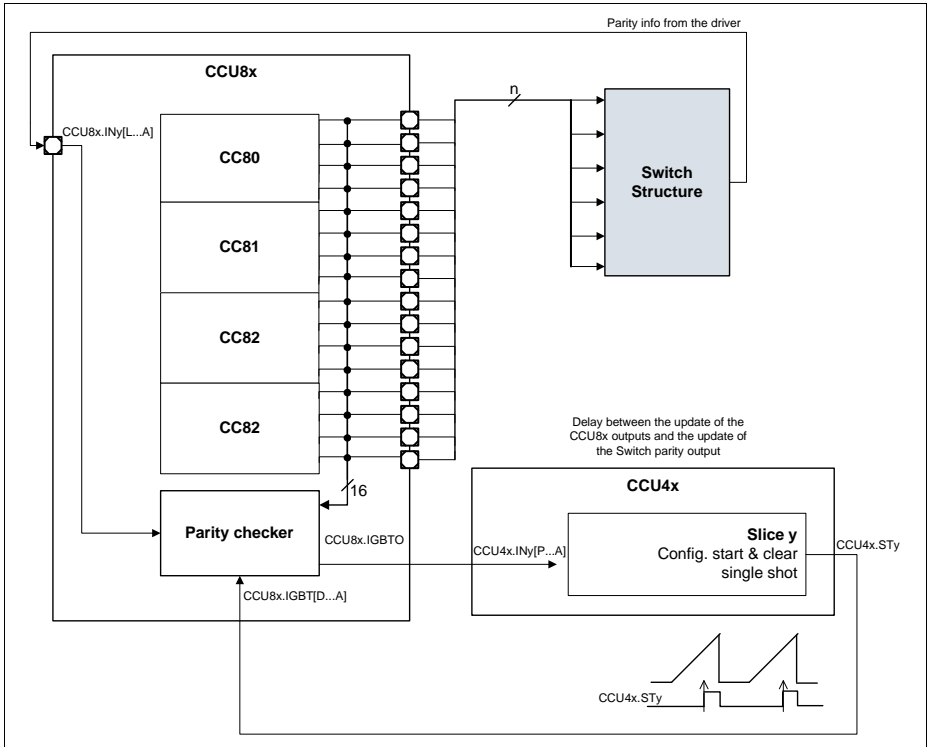


Figure 22-93 Parity Checker connections

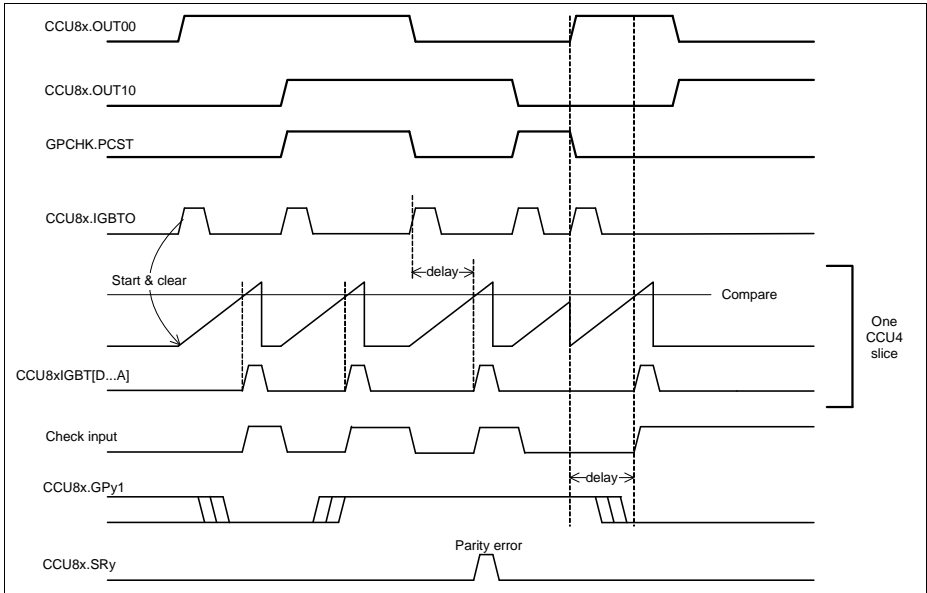


Figure 22-94 Parity Checker timing example

22.3 Service Request Generation

Each CCU8 slice has an interrupt structure as the one seen in [Figure 22-95](#). The register [CC8yINTS](#) is the status register for the interrupt sources. Each dedicated interrupt source can be set or cleared by SW, by writing into the specific bit in the [CC8ySWS](#) and [CC8ySWR](#) registers respectively.

Each interrupt source can be enabled/disabled via the [CC8yINTE](#) register. An enabled interrupt source will always generate a pulse on the service request line even if the specific status bit was not cleared. [Table 22-10](#) describes the interrupt sources of each CCU8 slice.

The interrupt sources, Period Match while counting up and one Match while counting down are ORed together. The same mechanism is applied to the Compare Match while counting up and Compare Match while counting down of both compare channels.

The interrupt sources for the external events are directly linked with the configuration set on the [CC8yINS.EVxEM](#). If an event is programmed to be active on both edges, that means that service request pulse is going to be generated when any transition on the external signal is detected. If the event is linked with a level function, the [CC8yINS.EVxEM](#) still can be programmed to enable a service request pulse. The TRAP

Capture/Compare Unit 8 (CCU8)

event doesn't need any extra configuration for generating the service request pulse when the slice enters the TRAP state.

Table 22-10 Interrupt sources

Signal	Description
CCINEV0_E	Event 0 edge(s) information from event selector. Used when an external signal should trigger an interrupt.
CCINEV1_E	Event 1 edge(s) information from event selector. Used when an external signal should trigger an interrupt (It also can be the parity checker pattern fail information, if the parity checker was enabled).
CCINEV2_E	Event 2 edge(s) information from event selector. Used when an external signal should trigger an interrupt.
CCPM_U	Period Match while counting up.
CCCM1_U	Compare Match while counting up from compare channel 1.
CCCM1_D	Compare Match while counting down from compare channel 1.
CCCM2_U	Compare Match while counting up from compare channel 2.
CCCM2_D	Compare Match while counting down from compare channel 2.
CCOM_D	One Match while counting down.
Trap state set	Entering Trap State. Will set the E2AS.

Each of the interrupt events can then be forwarded to one, of the slice's four service request lines, [Figure 22-96](#). The value set on the **CC8ySRS** controls which interrupt event is mapped into which service request line.

Capture/Compare Unit 8 (CCU8)

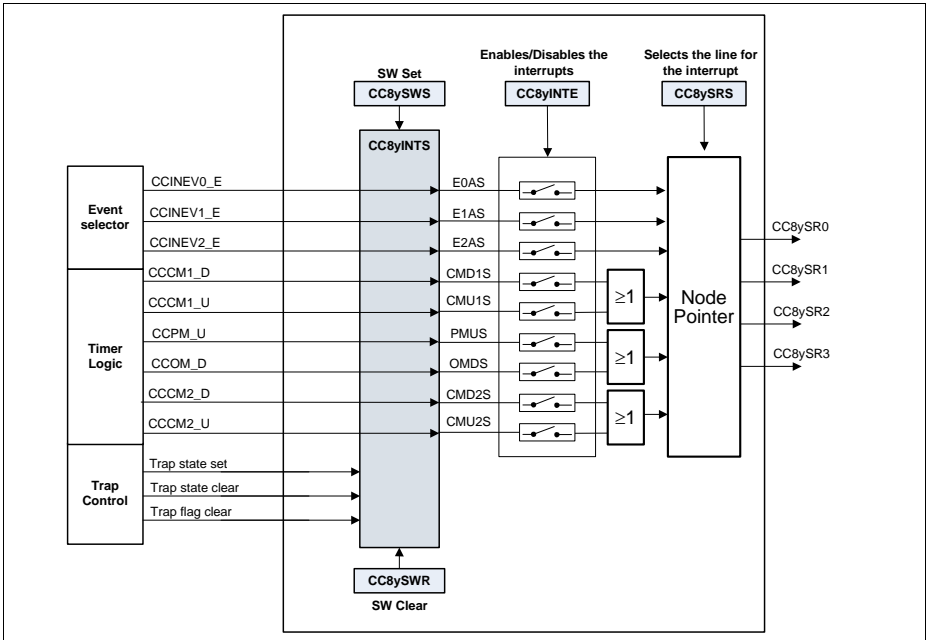


Figure 22-95 Slice interrupt node pointer overview

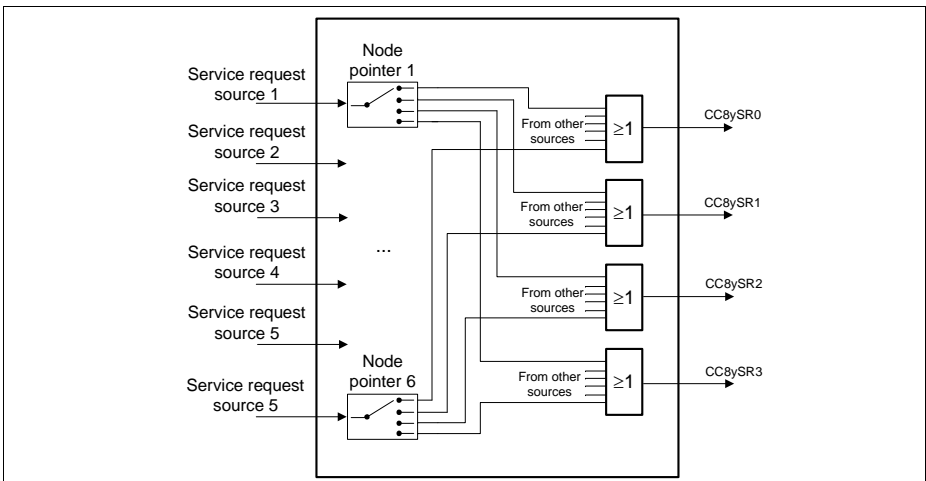


Figure 22-96 Slice interrupt selector overview

Capture/Compare Unit 8 (CCU8)

The four service request lines of each slice are OR together inside the kernel of the CCU8, see **Figure 22-97**. This means that there are only four service request lines per CCU8, that can have in each line interrupt requests coming from different slices.

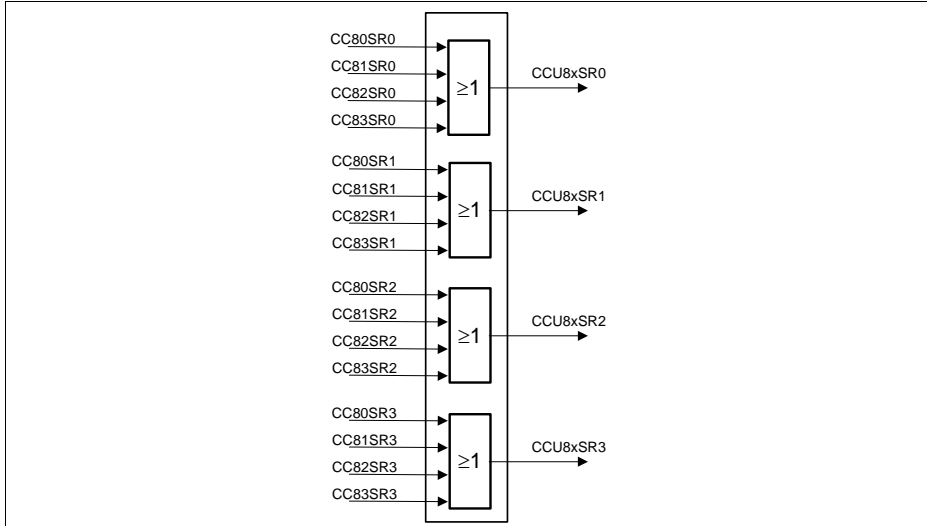


Figure 22-97 CCU8 service request overview

22.4 Debug Behavior

In suspend mode, the functional clocks for all slices as well the prescaler are stopped. The registers can still be accessed by the CPU (read only). This mode is useful for debugging purposes, e.g., where the current device status should be frozen in order to get a snapshot of the internal values. In suspend mode, all the slice counters are stopped. The suspend mode is non-intrusive concerning the register bits. This means register bits are not modified by hardware when entering or leaving the suspend mode.

Entry into suspend mode can be configured at the kernel level by means of the field **GCTRL.SUSCFG**.

The module is only functional after the suspend signal becomes inactive.

22.5 Power, Reset and Clock

The following sections describe the operating conditions, characteristics and timing requirements for the CCU8. All the timing information is related to the module clock, f_{CCU8} .

22.5.1 Clocks

Module Clock

The module clock of the CCU8 module is described in the SCU chapter as f_{CCU} .

The bus interface clock of the CCU8 module is described in the SCU chapter as f_{PERIPH} .

The module clock for the CCU8 is controlled via a specific control bit inside the SCU (System Control Unit), register CLKSET.

It is possible to disable the module clock for the CCU8 via the **GSTAT**, nevertheless, there may be a dependency of the f_{CCU8} through the different CCU8 instances. One should address the SCU Chapter for a complete description of the product clock scheme.

If module clock dependencies exist through different IP instances, then one can disable the module clock internally inside the specific CCU8, by disabling the prescaler (**GSTAT.PRB** = 0_B).

External Clock

It is possible to use an external clock as source for the prescaler, and consequently for all the timer Slices, CC8y. This external source can be connected to one of the CCU8x.CLK[C...A] inputs.

This external source is nevertheless synchronized against f_{CCU8} .

Table 22-11 External clock operating conditions

Parameter	Symbol	Values			Unit	Note / Test Condition
		Min.	Typ.	Max.		
Frequency	f_{eclk}	–	–	$f_{CCU8}/4$	MHz	
ON time	ton_{eclk}	$2T_{CCU8}^{1)2)}$	–	–	ns	
OFF time	$t_{off_{eclk}}$	$2T_{CCU8}^{1)2)}$	–	–	ns	Only the rising edge is used

1) Only valid if the signal was not previously synchronized/generated with the fccu4 clock (or a synchronous clock)

2) 50% duty cycle is not obligatory

22.5.2 Module Reset

Each CCU8 has one reset source. This reset source is handled at system level and it can be generated independently via a system control register, PRSET0/PRSET1 (address SCU chapter for a full description).

Capture/Compare Unit 8 (CCU8)

After reset release, the complete IP is set to default configuration. The default configuration for each register field is addressed on [Section 22.7](#).

22.5.3 Power

The CCU8 is inside the power core domain, therefore no special considerations about power up or power down sequences need to be taken. For a explanation about the different power domains, please address the SCU (System Control Unit) chapter.

An internal power down mode for the CCU8, can be achieved by disabling the clock inside the CCU8 itself. For this one should set the **GSTAT** register with the default reset value (via the idle mode set register, **GIDLS**).

22.6 Initialization and System Dependencies

22.6.1 Initialization Sequence

The initialization sequence for an application that is using the CCU8, should be the following:

- 1st Step:** Apply reset to the CCU8, via the specific SCU bitfield on the PRSET0/PRSET1 register.
- 2nd Step:** Release reset of the CCU8, via the specific SCU bitfield on the PRCLR0/PRCLR1 register.
- 3rd Step:** Enable the CCU8 clock via the specific SCU register, CLKSET.
- 4th Step:** Enable the prescaler block, by writing 1_B to the **GIDLC**.SPRB field.
- 5th Step:** Configure the global CCU8 register **GCTRL**.
- 6th Step:** Configure all the registers related to the required Timer Slice(s) functions, including the interrupt/service request configuration.
- 7th Step:** If needed, configure the startup value for a specific Compare Channel Status, of a Timer Slice, by writing 1_B to the specific **GCSS**.SyTS.
- 8th Step:** Configure the parity checker function if used, by programming the **GPCHK** register
- 9th Step:** Enable the specific timer slice(s), CC8y, by writing 1_B to the specific **GIDLC**.CSyl.
- 10th Step:** For all the Timer Slices that should be started synchronously via SW, the specific system register localized in the SCU, CCUCON, that enables a synchronous timer start should be addressed. The SCU.GSC8x input signal needs to be configured previously as a start function, see [Section 22.2.8.1](#).

22.6.2 System Dependencies

Each CCU8 may have different dependencies regarding module and bus clock frequencies. These dependencies should be addressed in the SCU and System Architecture Chapters.

Dependencies between several peripherals, regarding different clock operating frequencies may also exist. This should be addressed before configuring the connectivity between the CCU8 and some other peripheral.

The following topics must be taken into consideration for good CCU8 and system operation:

- CCU8 module clock must be at maximum two times faster than the module bus interface clock
- Module input triggers for the CCU8 must not exceed the module clock frequency (if the triggers are generated internally in the device)
- Module input triggers for the CCU8 must not exceed the frequency dictated in [Section 22.5.1](#)
- Frequency of the CCU8 outputs used as triggers/functions on other modules, must be crosschecked on the end point
- Applying and removing CCU8 from reset, can cause unwanted operations in other modules. This can occur if the modules are using CCU8 outputs as triggers/functions.

22.7 Registers

Registers Overview

The absolute register address is calculated by adding:
Module Base Address + Offset Address

Table 22-12 Registers Address Space

Module	Base Address	End Address	Note
CCU80	40020000 _H	40023FFF _H	

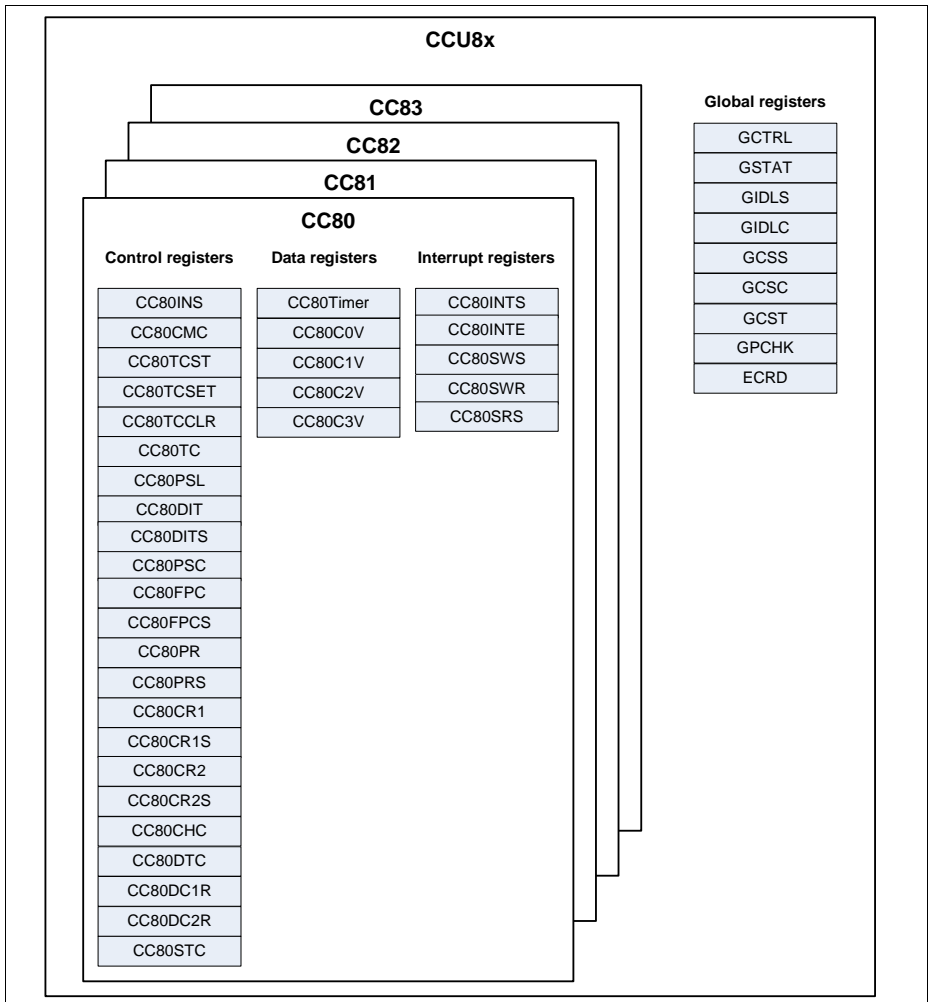


Figure 22-98 CCU8 registers overview

Table 22-13 Register Overview of CCU8

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	

CCU8 Global Registers

GCTRL	Module General Control Register	0000 _H	U, PV	U, PV	Page 22-112
GSTAT	General Slice Status Register	0004 _H	U, PV	BE	Page 22-115
GIDLS	General Idle Enable Register	0008 _H	U, PV	U, PV	Page 22-117
GIDLC	General Idle Disable Register	000C _H	U, PV	U, PV	Page 22-119
GCSS	General Channel Set Register	0010 _H	U, PV	U, PV	Page 22-120
GCSC	General Channel Clear Register	0014 _H	U, PV	U, PV	Page 22-123
GCST	General Channel Status Register	0018 _H	U, PV	BE	Page 22-126
GPCHK	Parity Checker Configuration Register	001C _H	U, PV	U, PV	Page 22-129
MIDR	Module Identification Register	0080 _H	U, PV	BE	Page 22-131

CC80 Registers

CC80INS	Input Selector Unit Configuration	0100 _H	U, PV	U, PV	Page 22-132
CC80CMC	Connection Matrix Configuration	0104 _H	U, PV	U, PV	Page 22-134
CC80TCST	Timer Run Status	0108 _H	U, PV	BE	Page 22-138
CC80TCSET	Timer Run Set	010C _H	U, PV	U, PV	Page 22-139
CC80TCCLR	Timer Run Clear	0110 _H	U, PV	U, PV	Page 22-140
CC80TC	General Timer Configuration	0114 _H	U, PV	U, PV	Page 22-141
CC80PSL	Output Passive Level Configuration	0118 _H	U, PV	U, PV	Page 22-146
CC80DIT	Dither Configuration	011C _H	U, PV	BE	Page 22-147
CC80DITS	Dither Shadow Register	0120 _H	U, PV	U, PV	Page 22-148
CC80PSC	Prescaler Configuration	0124 _H	U, PV	U, PV	Page 22-149

Capture/Compare Unit 8 (CCU8)

Table 22-13 Register Overview of CCU8 (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
CC80FPC	Prescaler Compare Value	0128 _H	U, PV	U, PV	Page 22-149
CC80FPCS	Prescaler Shadow Compare Value	012C _H	U, PV	U, PV	Page 22-150
CC80PR	Timer Period Value	0130 _H	U, PV	BE	Page 22-151
CC80PRS	Timer Period Shadow Value	0134 _H	U, PV	U, PV	Page 22-152
CC80CR1	Timer Compare Value for Channel 1	0138 _H	U, PV	BE	Page 22-152
CC80CR1S	Timer Compare Shadow Value for Channel 1	013C _H	U, PV	U, PV	Page 22-153
CC80CR2	Timer Compare Value for Channel 2	0140 _H	U, PV	BE	Page 22-154
CC80CR2S	Timer Compare Shadow Value for Channel 2	0144 _H	U, PV	U, PV	Page 22-155
CC80CHC	Channel Control	0148 _H	U, PV	U, PV	Page 22-156
CC80DTC	Dead Time Control	014C _H	U, PV	U, PV	Page 22-158
CC80DC1R	Channel 1 Dead Time Counter Values	0150 _H	U, PV	U, PV	Page 22-159
CC80DC2R	Channel 2 Dead Time Counter Values	0154 _H	U, PV	U, PV	Page 22-160
CC80TIMER	Timer Current Value	0170 _H	U, PV	U, PV	Page 22-160
CC80C0V	Capture Register 0 Value	0174 _H	U, PV	BE	Page 22-161
CC80C1V	Capture Register 1 Value	0178 _H	U, PV	BE	Page 22-162
CC80C2V	Capture Register 2 Value	017C _H	U, PV	BE	Page 22-163
CC80C3V	Capture Register 3 Value	0180 _H	U, PV	BE	Page 22-164
CC80INTS	Interrupt Status	01A0 _H	U, PV	BE	Page 22-165
CC80INTE	Interrupt Enable	01A4 _H	U, PV	U, PV	Page 22-167
CC80SRS	Interrupt Configuration	01A8 _H	U, PV	U, PV	Page 22-169
CC80SWS	Interrupt Status Set	01AC _H	U, PV	U, PV	Page 22-171
CC80SWR	Interrupt Status Clear	01B0 _H	U, PV	U, PV	Page 22-173
CC80STC	Shadow Transfer Control	01B4 _H	U, PV	U, PV	Page 22-175
CC80ECD0	Extended Read Back 0	01B8 _H	U, PV	BE	Page 22-176

Capture/Compare Unit 8 (CCU8)
Table 22-13 Register Overview of CCU8 (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
CC80ECD1	Extended Read Back 1	01BC _H	U, PV	BE	Page 22-177

CC81 Registers

CC81INS	Input Selector Unit Configuration	0200 _H	U, PV	U, PV	Page 22-132
CC81CMC	Connection Matrix Configuration	0204 _H	U, PV	U, PV	Page 22-134
CC81TCST	Timer Run Status	0208 _H	U, PV	BE	Page 22-138
CC81TCSET	Timer Run Set	020C _H	U, PV	U,PV	Page 22-139
CC81TCCLR	Timer Run Clear	0210 _H	U, PV	U, PV	Page 22-140
CC81TC	General Timer Configuration	0214 _H	U, PV	U, PV	Page 22-141
CC81PSL	Output Passive Level Configuration	0218 _H	U, PV	U, PV	Page 22-146
CC81DIT	Dither Configuration	021C _H	U, PV	BE	Page 22-147
CC81DITS	Dither Shadow Register	0220 _H	U, PV	U, PV	Page 22-148
CC81PSC	Prescaler Configuration	0224 _H	U, PV	U, PV	Page 22-149
CC81FPC	Prescaler Compare Value	0228 _H	U, PV	U, PV	Page 22-149
CC81FPCS	Prescaler Shadow Compare Value	022C _H	U, PV	U, PV	Page 22-150
CC81PR	Timer Period Value	0230 _H	U, PV	BE	Page 22-151
CC81PRS	Timer Period Shadow Value	0234 _H	U, PV	U, PV	Page 22-152
CC81CR1	Timer Compare Value for Channel 1	0238 _H	U, PV	BE	Page 22-152
CC81CR1S	Timer Compare Shadow Value for Channel 1	023C _H	U, PV	U, PV	Page 22-153
CC81CR2	Timer Compare Value for Channel 2	0240 _H	U, PV	BE	Page 22-154
CC81CR2S	Timer Compare Shadow Value for Channel 2	0244 _H	U, PV	U, PV	Page 22-155
CC81CHC	Channel Control	0248 _H	U, PV	U, PV	Page 22-156
CC81DTC	Dead Time Control	024C _H	U, PV	U, PV	Page 22-158

Capture/Compare Unit 8 (CCU8)
Table 22-13 Register Overview of CCU8 (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
CC81DC1R	Channel 1 Dead Time Counter Values	0250 _H	U, PV	U, PV	Page 22-159
CC81DC2R	Channel 2 Dead Time Counter Values	0254 _H	U, PV	U, PV	Page 22-160
CC81TIMER	Timer Current Value	0270 _H	U, PV	U, PV	Page 22-160
CC81C0V	Capture Register 0 Value	0274 _H	U, PV	BE	Page 22-161
CC81C1V	Capture Register 1 Value	0278 _H	U, PV	BE	Page 22-162
CC81C2V	Capture Register 2 Value	027C _H	U, PV	BE	Page 22-163
CC81C3V	Capture Register 3 Value	0280 _H	U, PV	BE	Page 22-164
CC81INTS	Interrupt Status	02A0 _H	U, PV	BE	Page 22-165
CC81INTE	Interrupt Enable	02A4 _H	U, PV	U, PV	Page 22-167
CC81SRS	Interrupt Configuration	02A8 _H	U, PV	U, PV	Page 22-169
CC81SWS	Interrupt Status Set	02AC _H	U, PV	U, PV	Page 22-171
CC81SWR	Interrupt Status Clear	02B0 _H	U, PV	U, PV	Page 22-173
CC81STC	Shadow Transfer Control	02B4 _H	U, PV	U, PV	Page 22-175
CC81ECD0	Extended Read Back 0	02B8 _H	U, PV	BE	Page 22-176
CC81ECD1	Extended Read Back 1	02BC _H	U, PV	BE	Page 22-177

CC82 Registers

CC82INS	Input Selector Unit Configuration	0300 _H	U, PV	U, PV	Page 22-132
CC82CMC	Connection Matrix Configuration	0304 _H	U, PV	U, PV	Page 22-134
CC82TCST	Timer Run Status	0308 _H	U, PV	BE	Page 22-138
CC82TCSET	Timer Run Set	030C _H	U, PV	U,PV	Page 22-139
CC82TCCLR	Timer Run Clear	0310 _H	U, PV	U, PV	Page 22-140
CC82TC	General Timer Configuration	0314 _H	U, PV	U, PV	Page 22-141
CC82PSL	Output Passive Level Configuration	0318 _H	U, PV	U, PV	Page 22-146
CC82DIT	Dither Configuration	031C _H	U, PV	BE	Page 22-147
CC82DITS	Dither Shadow Register	0320 _H	U, PV	U, PV	Page 22-148

Capture/Compare Unit 8 (CCU8)
Table 22-13 Register Overview of CCU8 (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
CC82PSC	Prescaler Configuration	0324 _H	U, PV	U, PV	Page 22-149
CC82FPC	Prescaler Compare Value	0328 _H	U, PV	U, PV	Page 22-149
CC82FPCS	Prescaler Shadow Compare Value	032C _H	U, PV	U, PV	Page 22-150
CC82PR	Timer Period Value	0330 _H	U, PV	BE	Page 22-151
CC82PRS	Timer Period Shadow Value	0334 _H	U, PV	U, PV	Page 22-152
CC82CR1	Timer Compare Value for Channel 1	0338 _H	U, PV	BE	Page 22-152
CC82CR1S	Timer Compare Shadow Value for Channel 1	033C _H	U, PV	U, PV	Page 22-153
CC82CR2	Timer Compare Value for Channel 2	0340 _H	U, PV	BE	Page 22-154
CC82CR2S	Timer Compare Shadow Value for Channel 2	0344 _H	U, PV	U, PV	Page 22-155
CC82CHC	Channel Control	0348 _H	U, PV	U, PV	Page 22-156
CC82DTC	Dead Time Control	034C _H	U, PV	U, PV	Page 22-158
CC82DC1R	Channel 1 Dead Time Counter Values	0350 _H	U, PV	U, PV	Page 22-159
CC82DC2R	Channel 2 Dead Time Counter Values	0354 _H	U, PV	U, PV	Page 22-160
CC82TIMER	Timer Current Value	0370 _H	U, PV	U, PV	Page 22-160
CC82C0V	Capture Register 0 Value	0374 _H	U, PV	BE	Page 22-161
CC82C1V	Capture Register 1 Value	0378 _H	U, PV	BE	Page 22-162
CC82C2V	Capture Register 2 Value	037C _H	U, PV	BE	Page 22-163
CC82C3V	Capture Register 3 Value	0380 _H	U, PV	BE	Page 22-164
CC82INTS	Interrupt Status	03A0 _H	U, PV	BE	Page 22-165
CC82INTE	Interrupt Enable	03A4 _H	U, PV	U, PV	Page 22-167
CC82SRS	Interrupt Configuration	03A8 _H	U, PV	U, PV	Page 22-169
CC82SWS	Interrupt Status Set	03AC _H	U, PV	U, PV	Page 22-171
CC82SWR	Interrupt Status Clear	03B0 _H	U, PV	U, PV	Page 22-173
CC82STC	Shadow Transfer Control	03B4 _H	U, PV	U, PV	Page 22-175

Capture/Compare Unit 8 (CCU8)
Table 22-13 Register Overview of CCU8 (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
CC82ECD0	Extended Read Back 0	03B8 _H	U, PV	BE	Page 22-176
CC82ECD1	Extended Read Back 1	03BC _H	U, PV	BE	Page 22-177

CC83 Registers

CC83INS	Input Selector Unit Configuration	0400 _H	U, PV	U, PV	Page 22-132
CC83CMC	Connection Matrix Configuration	0404 _H	U, PV	U, PV	Page 22-134
CC83TCST	Timer Run Status	0408 _H	U, PV	BE	Page 22-138
CC83TCSET	Timer Run Set	040C _H	U, PV	U, PV	Page 22-139
CC83TCCLR	Timer Run Clear	0410 _H	U, PV	U, PV	Page 22-140
CC83TC	General Timer Configuration	0414 _H	U, PV	U, PV	Page 22-141
CC83PSL	Output Passive Level Configuration	0418 _H	U, PV	U, PV	Page 22-146
CC83DIT	Dither Configuration	041C _H	U, PV	BE	Page 22-147
CC83DITS	Dither Shadow Register	0420 _H	U, PV	U, PV	Page 22-148
CC83PSC	Prescaler Configuration	0424 _H	U, PV	U, PV	Page 22-149
CC83FPC	Prescaler Compare Value	0428 _H	U, PV	U, PV	Page 22-149
CC83FPCS	Prescaler Shadow Compare Value	042C _H	U, PV	U, PV	Page 22-150
CC83PR	Timer Period Value	0430 _H	U, PV	BE	Page 22-151
CC83PRS	Timer Period Shadow Value	0434 _H	U, PV	U, PV	Page 22-152
CC83CR1	Timer Compare Value for Channel 1	0438 _H	U, PV	BE	Page 22-152
CC83CR1S	Timer Compare Shadow Value for Channel 1	043C _H	U, PV	U, PV	Page 22-153
CC83CR2	Timer Compare Value for Channel 2	0440 _H	U, PV	BE	Page 22-154
CC83CR2S	Timer Compare Shadow Value for Channel 2	0444 _H	U, PV	U, PV	Page 22-155
CC83CHC	Channel Control	0448 _H	U, PV	U, PV	Page 22-156
CC83DTC	Dead Time Control	044C _H	U, PV	U, PV	Page 22-158

Table 22-13 Register Overview of CCU8 (cont'd)

Short Name	Description	Offset Addr. ¹⁾	Access Mode		Description See
			Read	Write	
CC83DC1R	Channel 1 Dead Time Counter Values	0450 _H	U, PV	U, PV	Page 22-159
CC83DC2R	Channel 2 Dead Time Counter Values	0454 _H	U, PV	U, PV	Page 22-160
CC83TIMER	Timer Current Value	0470 _H	U, PV	U, PV	Page 22-160
CC83C0V	Capture Register 0 Value	0474 _H	U, PV	BE	Page 22-161
CC83C1V	Capture Register 1 Value	0478 _H	U, PV	BE	Page 22-162
CC83C2V	Capture Register 2 Value	047C _H	U, PV	BE	Page 22-163
CC83C3V	Capture Register 3 Value	0480 _H	U, PV	BE	Page 22-164
CC83INTS	Interrupt Status	04A0 _H	U, PV	BE	Page 22-165
CC83INTE	Interrupt Enable	04A4 _H	U, PV	U, PV	Page 22-167
CC83SRS	Interrupt Configuration	04A8 _H	U, PV	U, PV	Page 22-169
CC83SWS	Interrupt Status Set	04AC _H	U, PV	U, PV	Page 22-171
CC83SWR	Interrupt Status Clear	04B0 _H	U, PV	U, PV	Page 22-173
CC83STC	Shadow Transfer Control	04B4 _H	U, PV	U, PV	Page 22-175
CC83ECD0	Extended Read Back 0	04B8 _H	U, PV	BE	Page 22-176
CC83ECD1	Extended Read Back 1	04BC _H	U, PV	BE	Page 22-177

1) The absolute register address is calculated as follows:
Module Base Address + Offset Address (shown in this column)

22.7.1 Global Registers

GCTRL

The register contains the global configuration fields that affect all the timer slices inside CCU8.

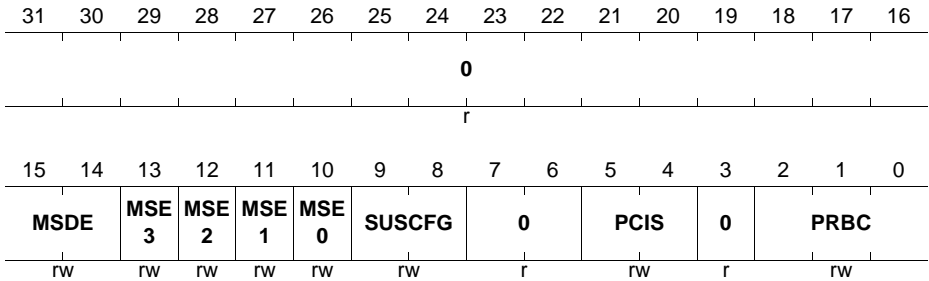
Capture/Compare Unit 8 (CCU8)

GCTRL

Global Control Register

(0000_H)

Reset Value: 00000000_H



Field	Bits	Type	Description
PRBC	[2:0]	rw	<p>Prescaler Clear Configuration</p> <p>This register controls how the prescaler Run Bit and internal registers are cleared.</p> <p>000_B SW only</p> <p>001_B GSTAT.PRB and prescaler registers are cleared when the Run Bit of CC80 is cleared.</p> <p>010_B GSTAT.PRB and prescaler registers are cleared when the Run Bit of CC81 is cleared.</p> <p>011_B GSTAT.PRB and prescaler registers are cleared when the Run Bit of CC82 is cleared.</p> <p>100_B GSTAT.PRB and prescaler registers are cleared when the Run Bit of CC83 is cleared.</p>
PCIS	[5:4]	rw	<p>Prescaler Input Clock Selection</p> <p>00_B Module clock</p> <p>01_B CCU8x.ECLKA</p> <p>10_B CCU8x.ECLKB</p> <p>11_B CCU8x.ECLKC</p>

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
SUSCFG	[9:8]	rw	<p>Suspend Mode Configuration This field controls the entering in suspend mode for all the CCU8 slices.</p> <p>00_B Suspend request ignored. The module never enters in suspend</p> <p>01_B Stops all the running slices immediately. Safe stop is not applied.</p> <p>10_B Stops the block immediately and clamps all the outputs to PASSIVE state. Safe stop is applied.</p> <p>11_B Waits for the roll over of each slice to stop and clamp the slices outputs. Safe stop is applied.</p>
MSE0	10	rw	<p>Slice 0 Multi Channel shadow transfer enable When this field is set, a shadow transfer of slice 0 can be requested not only by SW but also via the CCU8x.MCSS input.</p> <p>0_B Shadow transfer can only be requested by SW</p> <p>1_B Shadow transfer can be requested via SW and via the CCU8x.MCSS input.</p>
MSE1	11	rw	<p>Slice 1 Multi Channel shadow transfer enable When this field is set, a shadow transfer of slice 1 can be requested not only by SW but also via the CCU8x.MCSS input.</p> <p>0_B Shadow transfer can only be requested by SW</p> <p>1_B Shadow transfer can be requested via SW and via the CCU8x.MCSS input.</p>
MSE2	12	rw	<p>Slice 2 Multi Channel shadow transfer enable When this field is set, a shadow transfer of slice 2 can be requested not only by SW but also via the CCU8x.MCSS input.</p> <p>0_B Shadow transfer can only be requested by SW</p> <p>1_B Shadow transfer can be requested via SW and via the CCU8x.MCSS input.</p>

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
MSE3	13	rw	<p>Slice 3 Multi Channel shadow transfer enable When this field is set, a shadow transfer of slice 3 can be requested not only by SW but also via the CCU8x.MCSS input.</p> <p>0_B Shadow transfer can only be requested by SW 1_B Shadow transfer can be requested via SW and via the CCU8x.MCSS input.</p>
MSDE	[15:14]	rw	<p>Multi Channel shadow transfer request configuration This field configures the type of shadow transfer requested via the CCU8x.MCSS input. The field CC8yTC.MSEx needs to be set in order for this configuration to have any effect.</p> <p>00_B Only the shadow transfer for period and compare values is requested 01_B Shadow transfer for the compare, period and prescaler compare values is requested 10_B Reserved 11_B Shadow transfer for the compare, period, prescaler and dither compare values is requested</p>
0	3, [7:6], [31:16]	r	<p>Reserved A read always returns 0.</p>

GSTAT

The register contains the status of the prescaler and each timer slice (idle mode or running).

Capture/Compare Unit 8 (CCU8)

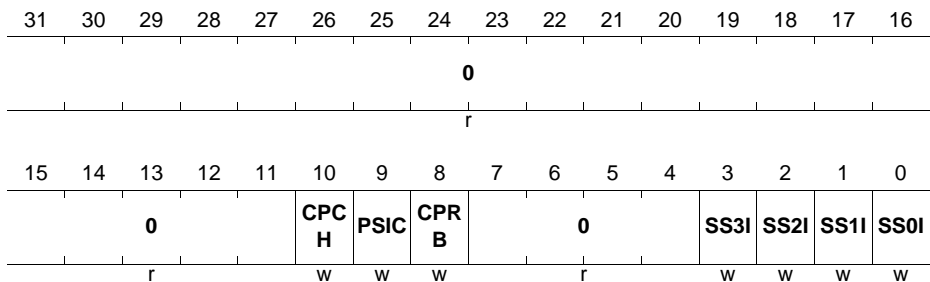
Field	Bits	Type	Description
PRB	8	rh	Prescaler Run Bit 0 _B Prescaler is stopped 1 _B Prescaler is running
PCRB	10	rh	Parity Checker Run Bit 0 _B Parity Checker is stopped 1 _B Parity Checker is running
0	[7:4], 9, [31:11]	r	Reserved Read always returns 0.

GIDLS

Through this register one can set the prescaler and the specific timer slices into idle mode.

GIDLS

Global Idle Set (0008_H) Reset Value: 00000000_H



Field	Bits	Type	Description
SS0I	0	w	CC80 IDLE mode set Writing a 1 _B to this bit sets the CC80 slice in IDLE mode. The clocks for the slice are stopped when in IDLE mode. When entering IDLE, the internal slice registers are not cleared. A read access always returns 0.

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
SS1I	1	w	CC81 IDLE mode set Writing a 1 _B to this bit sets the CC81 slice in IDLE mode. The clocks for the slice are stopped when in IDLE mode. When entering IDLE, the internal slice registers are not cleared. A read access always returns 0.
SS2I	2	w	CC82 IDLE mode set Writing a 1 _B to this bit sets the CC82 slice in IDLE mode. The clocks for the slice are stopped when in IDLE mode. When entering IDLE, the internal slice registers are not cleared. A read access always returns 0.
SS3I	3	w	CC83 IDLE mode set Writing a 1 _B to this bit sets the CC83 slice in IDLE mode. The clocks for the slice are stopped when in IDLE mode. When entering IDLE, the internal slice registers are not cleared. A read access always returns 0.
CPRB	8	w	Prescaler_B Run Bit Clear Writing a 1 into this register clears the Run Bit of the prescaler. Prescaler internal registers are not cleared. A read always returns 0.
PSIC	9	w	Prescaler clear Writing a 1 _B to this register clears the prescaler counter. It also loads the PSIV into the PVAL field for all Timer Slices. This performs a re alignment of the timer clock for all Slices. The Run Bit of the prescaler is not cleared. A read always returns 0.
CPCH	10	w	Parity Checker Run bit clear Writing a 1 _B to this register clears the run bit of the parity checker. All the internal registers are cleared. The status bit value is kept, GPCHK.PCST . A read always returns 0.
0	[7:4], [31:11]	r	Reserved Read always returns 0.

Capture/Compare Unit 8 (CCU8)

GIDLC

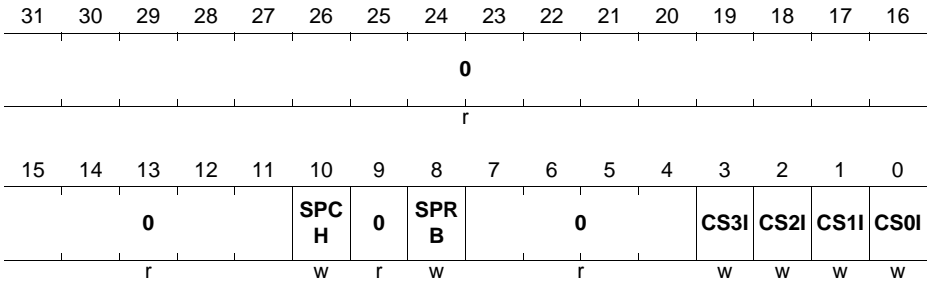
Through this register one can remove the prescaler and the specific timer slices from idle mode.

GIDLC

Global Idle Clear

(000C_H)

Reset Value: 00000000_H



Field	Bits	Type	Description
CS0I	0	w	CC80 IDLE mode clear Writing a 1 _B to this bit removes the CC80 from IDLE mode. No clear to the internal slice register is done. A read access always returns 0.
CS1I	1	w	CC81 IDLE mode clear Writing a 1 _B to this bit removes the CC81 from IDLE mode. No clear to the internal slice register is done. A read access always returns 0.
CS2I	2	w	CC82 IDLE mode clear Writing a 1 _B to this bit removes the CC82 from IDLE mode. No clear to the internal slice register is done. A read access always returns 0.
CS3I	3	w	CC83 IDLE mode clear Writing a 1 _B to this bit removes the CC83 from IDLE mode. No clear to the internal slice register is done. A read access always returns 0.
SPRB	8	w	Prescaler Run Bit Set Writing a 1 _B into this register sets the Run Bit of the prescaler. Prescaler internal registers are not cleared. A read always returns 0.

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
SPCH	10	w	Parity Checker run bit set Writing a 1 _B into this register sets the Run Bit of the parity checker. A read always returns 0.
0	[7:4], 9, [31:11]	r	Reserved Read always returns 0.

GCSS

Through this register one can request a shadow transfer for the specific timer slice(s) and set the status bit for each of the compare channels.

GCSS

Global Channel Set (0010_H) Reset Value: 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			0					S3S T2S	S2S T2S	S1S T2S	S0S T2S	S3S T1S	S2S T1S	S1S T1S	S0S T1S
			r					w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3P SE	S3D SE	S3S E	0	S2P SE	S2D SE	S2S E	0	S1P SE	S1D SE	S1S E	0	S0P SE	S0D SE	S0S E
r	w	w	w	r	w	w	w	r	w	w	w	r	w	w	w

Field	Bits	Type	Description
S0SE	0	w	Slice 0 shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S0SS field, enabling then a shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S0DSE	1	w	Slice 0 Dither shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S0DSS field, enabling then a shadow transfer for the Dither compare value. A read always returns 0.

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
S0PSE	2	w	Slice 0 Prescaler shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S0PSS field, enabling then a shadow transfer for the prescaler compare value. A read always returns 0.
S1SE	4	w	Slice 1 shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S1SS field, enabling then a shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S1DSE	5	w	Slice 1 Dither shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S1DSS field, enabling then a shadow transfer for the Dither compare value. A read always returns 0.
S1PSE	6	w	Slice 1 Prescaler shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S1PSS field, enabling then a shadow transfer for the prescaler compare value. A read always returns 0.
S2SE	8	w	Slice 2 shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S2SS field, enabling then a shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S2DSE	9	w	Slice 2 Dither shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S2DSS field, enabling then a shadow transfer for the Dither compare value. A read always returns 0.
S2PSE	10	w	Slice 2 Prescaler shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S2PSS field, enabling then a shadow transfer for the prescaler compare value. A read always returns 0.

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
S3SE	12	w	Slice 3 shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S3SS field, enabling then a shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S3DSE	13	w	Slice 3 Dither shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S3DSS field, enabling then a shadow transfer for the Dither compare value. A read always returns 0.
S3PSE	14	w	Slice 3 Prescaler shadow transfer set enable Writing a 1 _B to this bit will set the GCST.S3PSS field, enabling then a shadow transfer for the prescaler compare value. A read always returns 0.
S0ST1S	16	w	Slice 0 status bit 1 set Writing a 1 _B into this register sets the compare channel 1 status bit of slice 0 (GCST.CC80ST1). A read always returns 0.
S1ST1S	17	w	Slice 1 status bit 1 set Writing a 1 _B into this register sets the compare channel 1 status bit of slice 1 (GCST.CC81ST1). A read always returns 0.
S2ST1S	18	w	Slice 2 status bit 1 set Writing a 1 _B into this register sets the compare channel 1 status bit of slice 2 (GCST.CC82ST1). A read always returns 0.
S3ST1S	19	w	Slice 3 status bit 1 set Writing a 1 _B into this register sets the compare channel 1 status bit of slice 3 (GCST.CC83ST1). A read always returns 0.
S0ST2S	20	w	Slice 0 status bit 2 set Writing a 1 _B into this register sets the compare channel 2 status bit of slice 0 (GCST.CC80ST2). A read always returns 0.

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
S1ST2S	21	w	Slice 1 status bit 2 set Writing a 1 _B into this register sets the compare channel 2 status bit of slice 1 (GCST.CC81ST2). A read always returns 0.
S2ST2S	22	w	Slice 2 status bit 2 set Writing a 1 _B into this register sets the compare channel 2 status bit of slice 2 (GCST.CC82ST2). A read always returns 0.
S3ST2S	23	w	Slice 3 status bit 2 set Writing a 1 _B into this register sets the compare channel 2 status bit of slice 3 (GCST.CC83ST2). A read always returns 0.
0	3, 7, 11, 15, [31:24]	r	Reserved Read always returns 0.

GCSC

Through this register one can reset a shadow transfer request for the specific timer slice and clear the status bit for each the compare channels.

GCSC

Global Channel Clear

(0014_H)

Reset Value: 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
				0					S3S T2C	S2S T2C	S1S T2C	S0S T2C	S3S T1C	S2S T1C	S1S T1C	S0S T1C
				r					w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	S3P SC	S3D SC	S3S C	0	S2P SC	S2D SC	S2S C	0	S1P SC	S1D SC	S1S C	0	S0P SC	S0D SC	S0S C	
r	w	w	w	r	w	w	w	r	w	w	w	r	w	w	w	

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
S0SC	0	w	Slice 0 shadow transfer request clear Writing a 1 _B to this bit will clear the GCST.S0SS field, canceling any pending shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S0DSC	1	w	Slice 0 Dither shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S0DSS field, canceling any pending shadow transfer for the Dither compare value. A read always returns 0.
S0PSC	2	w	Slice 0 Prescaler shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S0PSS field, canceling any pending shadow transfer for the prescaler compare value. A read always returns 0.
S1SC	4	w	Slice 1 shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S1SS field, canceling any pending shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S1DSC	5	w	Slice 1 Dither shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S1DSS field, canceling any pending shadow transfer for the Dither compare value. A read always returns 0.
S1PSC	6	w	Slice 1 Prescaler shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S1PSS field, canceling any pending shadow transfer for the prescaler compare value. A read always returns 0.
S2SC	8	w	Slice 2 shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S2SS field, canceling any pending shadow transfer for the Period, Compare and Passive level values. A read always returns 0.

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
S2DSC	9	w	Slice 2 Dither shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S2DSS field, canceling any pending shadow transfer for the Dither compare value. A read always returns 0.
S2PSC	10	w	Slice 2 Prescaler shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S2PSS field, canceling any pending shadow transfer for the prescaler compare value. A read always returns 0.
S3SC	12	w	Slice 3 shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S3SS field, canceling any pending shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S3DSC	13	w	Slice 3 Dither shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S3DSS field, canceling any pending shadow transfer for the Dither compare value. A read always returns 0.
S3PSC	14	w	Slice 3 Prescaler shadow transfer clear Writing a 1 _B to this bit will clear the GCST.S3PSS field, canceling any pending shadow transfer for the prescaler compare value. A read always returns 0.
S0ST1C	16	w	Slice 0 status bit 1 clear Writing a 1 _B into this register clears the compare channel 1 status bit of slice 0 (GCST.CC80ST1). A read always returns 0.
S1ST1C	17	w	Slice 1 status bit 1 clear Writing a 1 _B into this register clears the compare channel 1 status bit of slice 1 (GCST.CC81ST1). A read always returns 0.
S2ST1C	18	w	Slice 2 status bit 1 clear Writing a 1 _B into this register clears the compare channel 1 status bit of slice 2 (GCST.CC82ST1). A read always returns 0.

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
S3ST1C	19	w	Slice 3 status bit 1 clear Writing a 1 _B into this register clears the compare channel 1 status bit of slice 3 (GCST.CC83ST1). A read always returns 0.
S0ST2C	20	w	Slice 0 status bit 2 clear Writing a 1 _B into this register clears the compare channel 2 status bit of slice 0 (GCST.CC80ST2). A read always returns 0.
S1ST2C	21	w	Slice 1 status bit 2 clear Writing a 1 _B into this register clears the compare channel 2 status bit of slice 1 (GCST.CC81ST2). A read always returns 0.
S2ST2C	22	w	Slice 2 status bit 2 clear Writing a 1 _B into this register clears the compare channel 2 status bit of slice 2 (GCST.CC82ST2). A read always returns 0.
S3ST2C	23	w	Slice 3 status bit 2 clear Writing a 1 _B into this register clears the compare channel 2 status bit of slice 3 (GCST.CC83ST2). A read always returns 0.
0	3, 7, 11, 15, [31:24]	r	Reserved Read always returns 0.

GCST

This register holds the information of the shadow transfer requests and of each timer slice status bit.

Capture/Compare Unit 8 (CCU8)

GCST

Global Channel status (0018_H) Reset Value: 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								CC8 3ST2	CC8 2ST2	CC8 1ST2	CC8 0ST2	CC8 3ST1	CC8 2ST1	CC8 1ST1	CC8 0ST1
r								rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3P SS	S3D SS	S3S S	0	S2P SS	S2D SS	S2S S	0	S1P SS	S1D SS	S1S S	0	S0P SS	S0D SS	S0S S
r	rh	rh	rh	r	rh	rh	rh	r	rh	rh	rh	r	rh	rh	rh

Field	Bits	Type	Description
S0SS	0	rh	Slice 0 shadow transfer status 0 _B Shadow transfer has not been requested 1 _B Shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S0DSS	1	rh	Slice 0 Dither shadow transfer status 0 _B Dither shadow transfer has not been requested 1 _B Dither shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S0PSS	2	rh	Slice 0 Prescaler shadow transfer status 0 _B Prescaler shadow transfer has not been requested 1 _B Prescaler shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S1SS	4	rh	Slice 1 shadow transfer status 0 _B Shadow transfer has not been requested 1 _B Shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
S1DSS	5	rh	<p>Slice 1 Dither shadow transfer status</p> <p>0_B Dither shadow transfer has not been requested</p> <p>1_B Dither shadow transfer has been requested</p> <p>This field is cleared by HW after the requested shadow transfer has been executed.</p>
S1PSS	6	rh	<p>Slice 1 Prescaler shadow transfer status</p> <p>0_B Prescaler shadow transfer has not been requested</p> <p>1_B Prescaler shadow transfer has been requested</p> <p>This field is cleared by HW after the requested shadow transfer has been executed.</p>
S2SS	8	rh	<p>Slice 2 shadow transfer status</p> <p>0_B Shadow transfer has not been requested</p> <p>1_B Shadow transfer has been requested</p> <p>This field is cleared by HW after the requested shadow transfer has been executed.</p>
S2DSS	9	rh	<p>Slice 2 Dither shadow transfer status</p> <p>0_B Dither shadow transfer has not been requested</p> <p>1_B Dither shadow transfer has been requested</p> <p>This field is cleared by HW after the requested shadow transfer has been executed.</p>
S2PSS	10	rh	<p>Slice 2 Prescaler shadow transfer status</p> <p>0_B Prescaler shadow transfer has not been requested</p> <p>1_B Prescaler shadow transfer has been requested</p> <p>This field is cleared by HW after the requested shadow transfer has been executed.</p>
S3SS	12	rh	<p>Slice 3 shadow transfer status</p> <p>0_B Shadow transfer has not been requested</p> <p>1_B Shadow transfer has been requested</p> <p>This field is cleared by HW after the requested shadow transfer has been executed.</p>

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
S3DSS	13	rh	Slice 3 Dither shadow transfer status 0_B Dither shadow transfer has not been requested 1_B Dither shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S3PSS	14	rh	Slice 3 Prescaler shadow transfer status 0_B Prescaler shadow transfer has not been requested 1_B Prescaler shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
CC80ST1	16	rh	Slice 0 compare channel 1 status bit
CC81ST1	17	rh	Slice 1 compare channel 1 status bit
CC82ST1	18	rh	Slice 2 compare channel 1 status bit
CC83ST1	19	rh	Slice 3 compare channel 1 status bit
CC80ST2	20	rh	Slice 0 compare channel 2 status bit
CC81ST2	21	rh	Slice 1 compare channel 2 status bit
CC82ST2	22	rh	Slice 2 compare channel 2 status bit
CC83ST2	23	rh	Slice 3 compare channel 2 status bit
0	3, 7, 11, 15, [31:24]	r	Reserved Read always returns 0.

GPCHK

This register contains the configuration for the Parity Check function of CCU8.

GPCHK

Parity Checker Configuration

(001C_H)

Reset Value: 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCSEL3				PCSEL2				PCSEL1				PCSEL0			
rw				rw				rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS T	0						PCT S	PCDS	PISEL	PACS	PAS E				
rh	r						rw	rw	rw	rw	rw				

Field	Bits	Type	Description
PASE	0	rw	Parity Checker Automatic start/stop If this field is set, the parity checker run bit is automatically set, when the run bit of the selected slice is set and it is cleared when the run bit of the slice is cleared. The field PACS needs to be programmed accordingly.
PACS	[2:1]	rw	Parity Checker Automatic start/stop selector This fields selects to which slice the automatic start/stop of the parity checker is associated: 00 _B CC80 01 _B CC81 10 _B CC82 11 _B CC83
PISEL	[4:3]	rw	Driver Input signal selector This fields selects which signal contains the driver parity information: 00 _B CC8x.GP01 - driver output is connected to event 1 of slice 0 01 _B CC8x.GP11 - drive output is connected to event 1 of slice 1 10 _B CC8x.GP21 - driver output is connected to event 1 of slice 2 11 _B CC8x.GP31 - driver output is connected to event 1 of slice 3

Capture/Compare Unit 8 (CCU8)

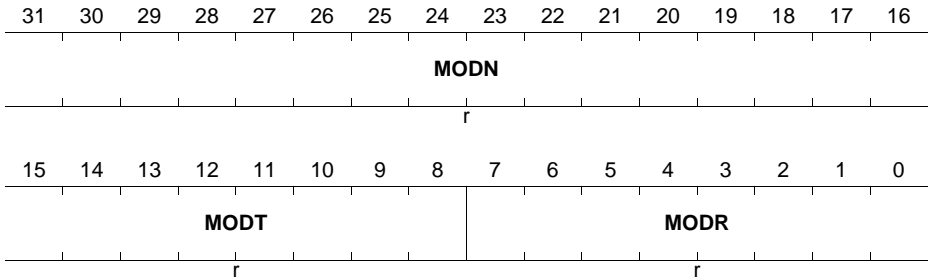
Field	Bits	Type	Description
PCDS	[6:5]	rw	Parity Checker Delay Input Selector This fields selects which signal is controlling the delay between the change at the CCU8 outputs and effective change at the driver parity output: 00 _B CCU8x.IGBTA 01 _B CCU8x.IGBTB 10 _B CCU8x.IGBTC 11 _B CCU8x.IGBTD
PCTS	7	rw	Parity Checker type selector This fields selects if we have an odd or even parity: 0 _B Even parity enabled 1 _B Odd parity enabled
PCST	15	rh	Parity Checker XOR status This field contains the current value of the XOR chain.
PCSELO	[19:16]	rw	Parity Checker Slice 0 output selection This fields selects which slice 0 outputs are going to be used to perform the parity check. The respective bit field needs to be set to 1 _B to enable the output in the parity function. PCSELO[0] - CCU8x.OUT00 PCSELO[1] - CCU8x.OUT01 PCSELO[2] - CCU8x.OUT02 PCSELO[3] - CCU8x.OUT03
PCSEL1	[23:20]	rw	Parity Checker Slice 1 output selection Same description as PCSELO.
PCSEL2	[27:24]	rw	Parity Checker Slice 2 output selection Same description as PCSELO.
PCSEL3	[31:28]	rw	Parity Checker Slice 3 output selection Same description as PCSELO.
0	[14:8]	r	Reserved Read always returns 0.

MIDR

This register contains the module identification number.

MIDR

Module Identification (0080_H) Reset Value: 00A7C0XX_H



Field	Bits	Type	Description
MODR	[7:0]	r	Module Revision This bit field indicates the revision number of the module implementation (depending on the design step). The given value of 00 _H is a placeholder for the actual number.
MODT	[15:8]	r	Module Type
MODN	[31:16]	r	Module Number

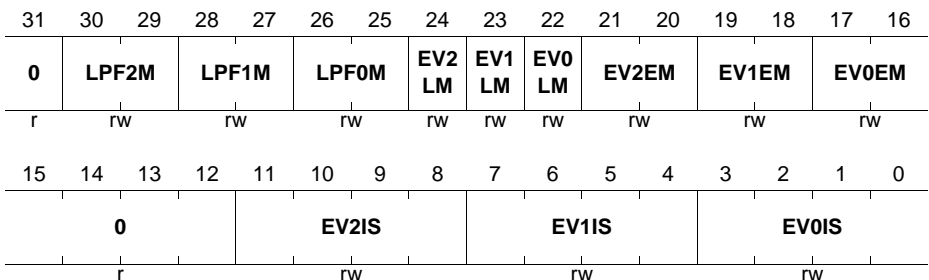
22.7.2 Slice (CC8y) Registers

CC8yINS

The register contains the configuration for the input selector.

CC8yINS (y = 0 - 3)

Input Selector Configuration (0100_H + 0100_H * y) Reset Value: 00000000_H



Field	Bits	Type	Description
EV0IS	[3:0]	rw	Event 0 signal selection This field selects which pins is used for the event 0. 0000 _B CCU8x.INyA 0001 _B CCU8x.INyB 0010 _B CCU8x.INyC 0011 _B CCU8x.INyD 0100 _B CCU8x.INyE 0101 _B CCU8x.INyF 0110 _B CCU8x.INyG 0111 _B CCU8x.INyH 1000 _B CCU8x.INyI 1001 _B CCU8x.INyJ 1010 _B CCU8x.INyK 1011 _B CCU8x.INyL 1100 _B CCU8x.INyM 1101 _B CCU8x.INyN 1110 _B CCU8x.INyO 1111 _B CCU8x.INyP
EV1IS	[7:4]	rw	Event 1 signal selection Same as EV0IS description
EV2IS	[11:8]	rw	Event 2 signal selection Same as EV0IS description
EV0EM	[17:16]	rw	Event 0 Edge Selection 00 _B No action 01 _B Signal active on rising edge 10 _B Signal active on falling edge 11 _B Signal active on both edges
EV1EM	[19:18]	rw	Event 1 Edge Selection Same as EV0EM description
EV2EM	[21:20]	rw	Event 2 Edge Selection Same as EV0EM description
EV0LM	22	rw	Event 0 Level Selection 0 _B Active on HIGH level 1 _B Active on LOW level
EV1LM	23	rw	Event 1 Level Selection Same as EV0LM description

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
EV2LM	24	rw	Event 2 Level Selection Same as EV0LM description
LPF0M	[26:25]	rw	Event 0 Low Pass Filter Configuration This field sets the number of consecutive counts for the Low Pass Filter of Event 0. The input signal value needs to remain stable for this number of counts (f_{CCU8}), so that a level/transition is accepted. 00 _B LPF is disabled 01 _B 3 clock cycles of f_{CCU8} 10 _B 5 clock cycles of f_{CCU8} 11 _B 7 clock cycles of f_{CCU8}
LPF1M	[28:27]	rw	Event 1 Low Pass Filter Configuration Same description as LPF0M
LPF2M	[30:29]	rw	Event 2 Low Pass Filter Configuration Same description as LPF0M
0	[15:12] , 31	r	Reserved Read always returns 0.

CC8yCMC

The register contains the configuration for the connection matrix.

CC8yCMC (y = 0 - 3)

Connection Matrix Control (0104_H + 0100_H * y) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											TCE	MOS	TS	OFS	
r											rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTS		LDS		UDS		GATES		CAP1S		CAP0S		ENDS		STRTS	
rw		rw		rw		rw		rw		rw		rw		rw	

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
STRTS	[1:0]	rw	<p>External Start Functionality Selector Selects the Event that is going to be linked with the external start functionality.</p> <p>00_B External Start Function deactivated 01_B External Start Function triggered by Event 0 10_B External Start Function triggered by Event 1 11_B External Start Function triggered by Event 2</p>
ENDS	[3:2]	rw	<p>External Stop Functionality Selector Selects the Event that is going to be linked with the external stop functionality.</p> <p>00_B External Stop Function deactivated 01_B External Stop Function triggered by Event 0 10_B External Stop Function triggered by Event 1 11_B External Stop Function triggered by Event 2</p>
CAP0S	[5:4]	rw	<p>External Capture 0 Functionality Selector Selects the Event that is going to be linked with the external capture for capture registers number 1 and 0. This function is used to capture the value of the timer into the capture registers 1 and 0.</p> <p>00_B External Capture 0 Function deactivated 01_B External Capture 0 Function triggered by Event 0 10_B External Capture 0 Function triggered by Event 1 11_B External Capture 0 Function triggered by Event 2</p> <p><i>Note: If the field SCE is set, this functionality is deactivated.</i></p>

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
CAP1S	[7:6]	rw	<p>External Capture 1 Functionality Selector Selects the Event that is going to be linked with the external capture for capture registers number 3 and 2. This function is used to capture the value of the timer into the capture registers 3 and 2.</p> <p>00_B External Capture 1 Function deactivated 01_B External Capture 1 Function triggered by Event 0 10_B External Capture 1 Function triggered by Event 1 11_B External Capture 1 Function triggered by Event 2</p>
GATES	[9:8]	rw	<p>External Gate Functionality Selector Selects the Event that is going to be linked with the counter gating function. This function is used to gate the timer increment/decrement procedure.</p> <p>00_B External Gating Function deactivated 01_B External Gating Function triggered by Event 0 10_B External Gating Function triggered by Event 1 11_B External Gating Function triggered by Event 2</p>
UDS	[11:10]	rw	<p>External Up/Down Functionality Selector Selects the Event that is going to be linked with the Up/Down counting direction control. This function is used to control externally the timer increment/decrement operation.</p> <p>00_B External Up/Down Function deactivated 01_B External Up/Down Function triggered by Event 0 10_B External Up/Down Function triggered by Event 1 11_B External Up/Down Function triggered by Event 2</p>

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
LDS	[13:12]	rw	<p>External Timer Load Functionality Selector Selects the Event that is going to be linked with the timer load function. The value present in the CC8yCR1/CC8yCR2 (depending on the value of CC8yTC.TLS) is loaded into the specific slice timer. 00_B - External Load Function deactivated 01_B - External Load Function triggered by Event 0 10_B - External Load Function triggered by Event 1 11_B - External Load Function triggered by Event 2</p>
CNTS	[15:14]	rw	<p>External Count Selector Selects the Event that is going to be linked with the count function. The counter is going to be increment/decremented each time that a specific transition on the event is detected. 00_B External Count Function deactivated 01_B External Count Function triggered by Event 0 10_B External Count Function triggered by Event 1 11_B External Count Function triggered by Event 2 <i>Note: In CC40 this field doesn't exist. This is a read only reserved field. Read access always returns 0.</i></p>
OFS	16	rw	<p>Override Function Selector This field enables the ST bit override functionality. 0_B Override functionality disabled 1_B Status bit trigger override connected to Event 1; Status bit value override connected to Event 2</p>
TS	17	rw	<p>Trap Function Selector This field enables the trap functionality. 0_B Trap function disabled 1_B TRAP function connected to Event 2</p>
MOS	[19:18]	rw	<p>External Modulation Functionality Selector Selects the Event that is going to be linked with the external modulation function. 00_B - Modulation Function deactivated 01_B - Modulation Function triggered by Event 0 10_B - Modulation Function triggered by Event 1 11_B - Modulation Function triggered by Event 2</p>

Capture/Compare Unit 8 (CCU8)

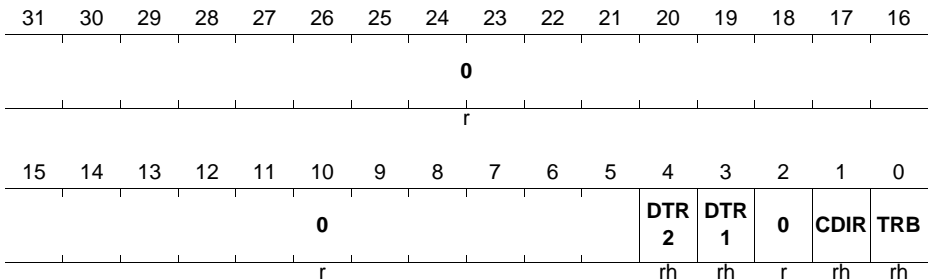
Field	Bits	Type	Description
TCE	20	rw	Timer Concatenation Enable This bit enables the timer concatenation with the previous slice. 0_B Timer concatenation is disabled 1_B Timer concatenation is enabled <i>Note: In CC80 this field doesn't exist. This is a read only reserved field. Read access always returns 0.</i>
0	[31:21]	r	Reserved A read always returns 0

CC8yTCST

The register holds the status of the timer (running/stopped) and the information about the counting direction (up/down).

CC8yTCST (y = 0 - 3)

Slice Timer Status (0108_H + 0100_H * y) **Reset Value: 00000000_H**



Field	Bits	Type	Description
TRB	0	rh	Timer Run Bit This field indicates if the timer is running. 0_B Timer is stopped 1_B Timer is running
CDIR	1	rh	Timer Counting Direction This field indicates if the timer is being increment or decremented 0_B Timer is counting up 1_B Timer is counting down

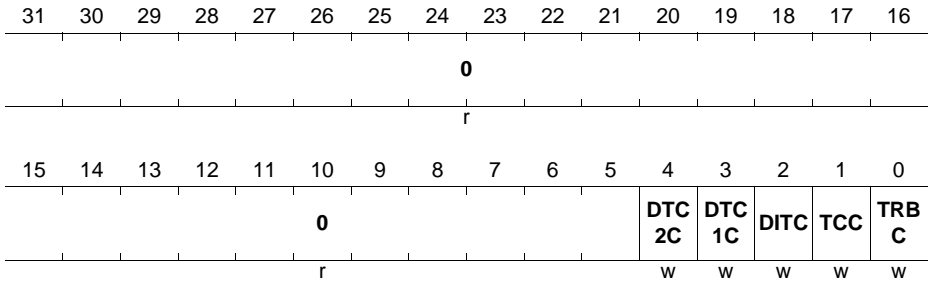
Capture/Compare Unit 8 (CCU8)

CC8yTCCLR

Through this register it is possible to stop and clear the timer, and clearing also the dither counter

CC8yTCCLR (y = 0 - 3)

Slice Timer Clear (0110_H + 0100_H * y) **Reset Value: 00000000_H**



Field	Bits	Type	Description
TRBC	0	w	Timer Run Bit Clear Writing a 1 _B into this field clears the run bit of the timer. The timer is not cleared. Read always returns 0.
TCC	1	w	Timer Clear Writing a 1 _B into this field clears the timer value. Read always returns 0.
DITC	2	w	Dither Counter Clear Writing a 1 _B into this field clears the dither counter. Read always returns 0.
DTC1C	3	w	Dead Time Counter 1 Clear Writing a 1 _B into this field clears the channel 1 dead time counter. The counter is stopped until a new start trigger is detected. Read always returns 0.
DTC2C	4	w	Dead Time Counter 2 Clear Writing a 1 _B into this field clears the channel 2 dead time counter. The counter is stopped until a new start trigger is detected. Read always returns 0.

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
0	[31:5]	r	Reserved Read always returns 0

CC8yTC

This register holds the several possible configurations for the timer operation.

CC8yTC (y = 0 - 3)

Slice Timer Control (0114_H + 0100_H * y) **Reset Value: 18000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	STOS	EME	MCM E2	MCM E1	EMT	EMS	TRP SW	TRP SE	TRA PE3	TRA PE2	TRA PE1	TRA PE0	FPE		
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIM	DITHE	CCS	SCE	STR M	ENDM	TLS	CAPC	ECM	CMO D	CLS T	TSS M	TCM			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TCM	0	rw	Timer Counting Mode This field controls the actual counting scheme of the timer. 0 _B Edge aligned mode 1 _B Center aligned mode <i>Note: When using an external signal to control the counting direction, the counting scheme is always edge aligned.</i>
TSSM	1	rw	Timer Single Shot Mode This field controls the single shot mode. This is applicable in edge and center aligned modes. 0 _B Single shot mode is disabled 1 _B Single shot mode is enabled

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
CLST	2	rw	<p>Shadow Transfer on Clear</p> <p>Setting this bit to 1_B enables a shadow transfer when a timer clearing action is done (by SW or by an external event). Notice that the shadow transfer enable bitfields on the GCST register still need to be set to 1_B via software.</p>
CMOD	3	rh	<p>Capture Compare Mode</p> <p>This field indicates in which mode the slice is operating. The default value is compare mode. The capture mode is automatically set by the HW when an external signal is mapped to a capture trigger.</p> <p>0_B Compare Mode 1_B Capture Mode</p>
ECM	4	rw	<p>Extended Capture Mode</p> <p>This field control the Capture mode of the specific slice. It only has effect if the CMOD bit is 1_B.</p> <p>0_B Normal Capture Mode. Clear of the Full Flag of each capture register is done by accessing the registers individually only.</p> <p>1_B Extended Capture Mode. Clear of the Full Flag of each capture register is done not only by accessing the individual registers but also by accessing the ECRD register. When reading the ECRD register, only the capture register register full flag pointed by the VPTR is cleared</p>
CAPC	[6:5]	rw	<p>Clear on Capture Control</p> <p>00_B Timer is never cleared on a capture event 01_B Timer is cleared on a capture event into capture registers 2 and 3. (When SCE = 1_B, Timer is always cleared in a capture event) 10_B Timer is cleared on a capture event into capture registers 0 and 1. (When SCE = 1_B, Timer is always cleared in a capture event) 11_B Timer is always cleared in a capture event.</p>
TLS	7	rw	<p>Timer Load selector</p> <p>0_B Timer is loaded with the value of CR1 1_B Timer is loaded with the value of CR2</p>

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
ENDM	[9:8]	rw	<p>Extended Stop Function Control</p> <p>This field controls the extended functions of the external Stop signal.</p> <p>00_B Clears the timer run bit only (default stop) 01_B Clears the timer only (flush) 10_B Clears the timer and run bit (flush/stop) 11_B Reserved</p> <p><i>Note: When using an external up/down signal the flush operation sets the timer with zero if the counter is counting up and with the Period value if the counter is being decremented.</i></p>
STRM	10	rw	<p>Extended Start Function Control</p> <p>This field controls the extended functions of the external Start signal.</p> <p>0_B Sets run bit only (default start) 1_B Clears the timer and sets run bit, if not set (flush/start)</p> <p><i>Note: When using an external up/down signal the flush operation sets the timer with zero if the counter is being incremented and with the Period value if the counter is being decremented.</i></p>
SCE	11	rw	<p>Equal Capture Event enable</p> <p>0_B Capture into CC8yC0V/CC8yC1V registers control by CCycapt0 and capture into CC8yC3V/CC8yC2V control by CCycapt1 1_B Capture into CC8yC0V/CC8yC1V and CC8yC3V/CC8yC2V control by CCycapt1</p>
CCS	12	rw	<p>Continuous Capture Enable</p> <p>0_B The capture into a specific capture register is done with the rules linked with the full flags, described at Section 22.2.8.6. 1_B The capture into the capture registers is always done regardless of the full flag status (even if the register has not been read back).</p>

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
DITHE	[14:13]	rw	<p>Dither Enable This field controls the dither mode for the slice. See Section 22.2.12.</p> <p>00_B Dither is disabled 01_B Dither is applied to the Period 10_B Dither is applied to the Compare 11_B Dither is applied to the Period and Compare</p>
DIM	15	rw	<p>Dither input selector This fields selects if the dither control signal is connected to the dither logic of the specific slice of is connected to the dither logic of slice 0. Notice that even if this field is set to 1_B, the field DITHE still needs to be programmed.</p> <p>0_B Slice is using it own dither unit 1_B Slice is connected to the dither unit of slice 0.</p>
FPE	16	rw	<p>Floating Prescaler enable Setting this bit to 1_B enables the floating prescaler mode.</p> <p>0_B Floating prescaler mode is disabled 1_B Floating prescaler mode is enabled</p>
TRAPE0	17	rw	<p>TRAP enable for CCU8x.OUTy0 Setting this bit to 1 enables the TRAP action at the CCU8x.OUTy0 output pin. After mapping an external signal to the TRAP functionality, the user must set this field to 1 to activate the effect of the TRAP on the specific output pin. Writing a 0 into this field disables the effect of the TRAP function regardless of the state of the input signal.</p> <p>0_B TRAP functionality has no effect on the CCU8x.OUTy0 output 1_B TRAP functionality affects the CCU8x.OUTy0 output</p>
TRAPE1	18	rw	<p>TRAP enable for CCU8x.OUTy1 TRAP enable for the CCU8x.OUTy1. Same description as for the TRAPE0 field.</p>
TRAPE2	19	rw	<p>TRAP enable for CCU8x.OUTy2 TRAP enable for the CCU8x.OUTy2. Same description as for the TRAPE0 field.</p>

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
TRAPE3	20	rw	TRAP enable for CCU8x.OUTy3 TRAP enable for the CCU8x.OUTy3. Same description as for the TRAPE0 field.
TRPSE	21	rw	TRAP Synchronization Enable Writing a 1 into this bit enables a synchronous exiting with the PWM period of the trap state. 0 _B Exiting from TRAP state isn't synchronized with the PWM signal 1 _B Exiting from TRAP state is synchronized with the PWM signal
TRPSW	22	rw	TRAP State Clear Control 0 _B The slice exits the TRAP state automatically when the TRAP condition is not present (Trap state cleared by HW and SW) 1 _B The TRAP state can only be exited by a SW request.
EMS	23	rw	External Modulation Synchronization Setting this bit to 1 enables the synchronization of the external modulation functionality with the PWM period. 0 _B External Modulation functionality is not synchronized with the PWM signal 1 _B External Modulation functionality is synchronized with the PWM signal
EMT	24	rw	External Modulation Type This field selects if the external modulation event is clearing the CC8ySTx bits or if is gating the outputs. 0 _B External Modulation functionality is clearing the CC8ySTx bits. 1 _B External Modulation functionality is gating the outputs.
MCME1	25	rw	Multi Channel Mode Enable for Channel 1 0 _B Multi Channel Mode in Channel 1 is disabled 1 _B Multi Channel Mode in Channel 1 is enabled
MCME2	26	rw	Multi Channel Mode Enable for Channel 2 0 _B Multi Channel Mode in Channel 2 is disabled 1 _B Multi Channel Mode in Channel 2 is enabled

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
EME	[28:27]	rw	External Modulation Channel enable This field controls in which channel, the modulation functionality has effect. The modulations functionality needs to be previously enabled by setting the CC8yCMC.MOS accordingly. 00 _B External Modulation functionality doesn't affect any channel 01 _B External Modulation only applied on channel 1 10 _B External Modulation only applied on channel 2 11 _B External Modulation applied on both channels
STOS	[30:29]	rw	Status bit output selector This field selects to which channel the output CC8ySTy is mapped. 00 _B CC8yST1 forward to CCU8x.STy 01 _B CC8yST2 forward to CCU8x.STy 10 _B CC8yST1 AND CC8yST2 forward to CCU8x.STy 11 _B CC8yST1 OR CC8yST2 forward to CCU8x.STy
0	31	r	Reserved Read always returns 0.

CC8yPSL

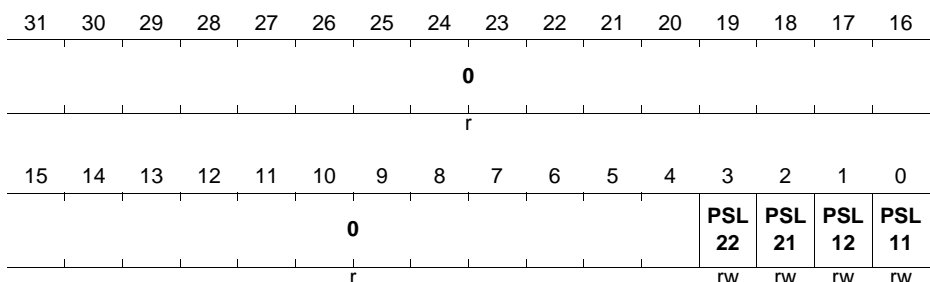
This register holds the configuration for the output passive level control.

CC8yPSL (y = 0 - 3)

Passive Level Config

$$(0118_H + 0100_H * y)$$

Reset Value: 00000000_H



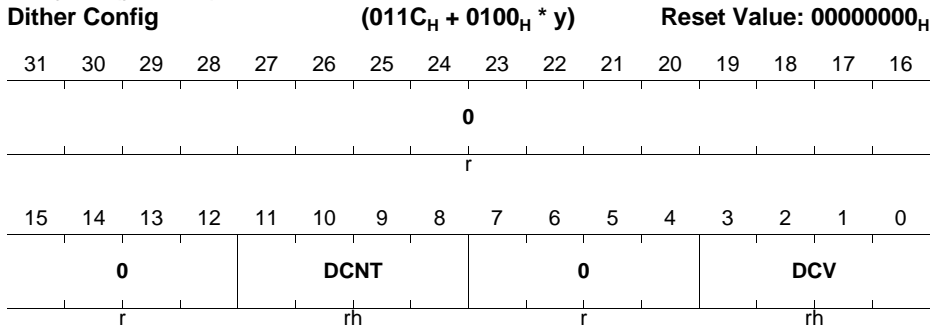
Field	Bits	Type	Description
PSL11	0	rw	Output Passive Level for CCU8x.OUTy0 This field controls the passive level of the CCU8x.OUTy0. 0 _B Passive Level is LOW 1 _B Passive Level is HIGH A write always addresses the shadow register, while a read always returns the current used value.
PSL12	1	rw	Output Passive Level for CCU8x.OUTy1 This field controls the passive level of the CCU8x.OUTy1. 0 _B Passive Level is LOW 1 _B Passive Level is HIGH A write always addresses the shadow register, while a read always returns the current used value.
PSL21	2	rw	Output Passive Level for CCU8x.OUTy2 This field controls the passive level of the CCU8x.OUTy2. 0 _B Passive Level is LOW 1 _B Passive Level is HIGH A write always addresses the shadow register, while a read always returns the current used value.
PSL22	3	rw	Output Passive Level for CCU8x.OUTy3 This field controls the passive level of the CCU8x.OUTy3. 0 _B Passive Level is LOW 1 _B Passive Level is HIGH A write always addresses the shadow register, while a read always returns the current used value.
0	[31:4]	r	Reserved A read access always returns 0

CC8yDIT

This register holds the current dither compare and dither counter values.

Capture/Compare Unit 8 (CCU8)

CC8yDIT (y = 0 - 3)

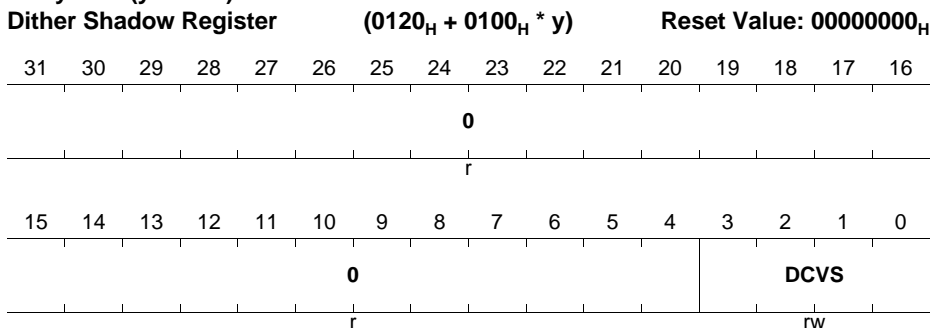


Field	Bits	Type	Description
DCV	[3:0]	rh	Dither compare Value This field contains the value used for the dither comparison. This value is updated when a shadow transfer occurs with the CC8yDITS.DCVS .
DCNT	[11:8]	rh	Dither counter actual value
0	[7:4], [31:12]	r	Reserved Read always returns 0.

CC8yDITS

This register contains the value that is going to be loaded into the **CC8yDIT.DCV** when the next shadow transfer occurs.

CC8yDITS (y = 0 - 3)



Capture/Compare Unit 8 (CCU8)

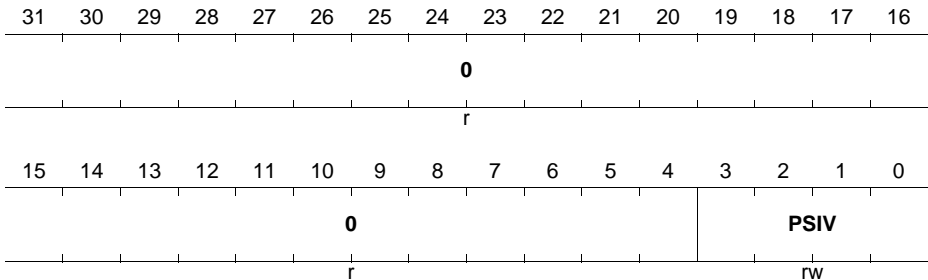
Field	Bits	Type	Description
DCVS	[3:0]	rw	Dither Shadow Compare Value This field contains the value that is going to be set on the dither compare value, CC8yDIT.DCV , within the next shadow transfer.
0	[31:4]	r	Reserved Read always returns 0.

CC8yPSC

This register contains the value that is loaded into the prescaler during restart.

CC8yPSC (y = 0 - 3)

Prescaler Control (0124_H + 0100_H * y) **Reset Value: 00000000_H**



Field	Bits	Type	Description
PSIV	[3:0]	rw	Prescaler Initial Value This field contains the value that is applied to the Prescaler at startup. When floating prescaler mode is used, this value is applied when a timer compare match AND prescaler compare match occurs or when a capture event is triggered.
0	[31:4]	r	Reserved Read always returns 0.

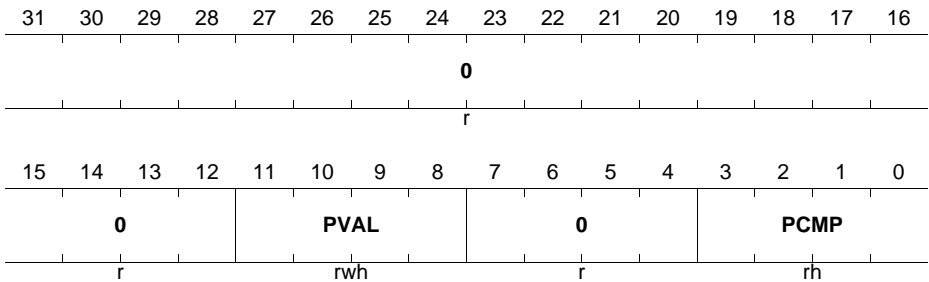
CC8yFPC

This register contains the value used for the floating prescaler compare and the actual prescaler division value.

Capture/Compare Unit 8 (CCU8)

CC8yFPC (y = 0 - 3)

Floating Prescaler Control $(0128_H + 0100_H * y)$ **Reset Value: 00000000_H**



Field	Bits	Type	Description
PCMP	[3:0]	rh	Floating Prescaler Compare Value This field contains the value used to compare the actual prescaler value. The comparison is triggered by the Timer Compare match event. See Section 22.2.13.2 .
PVAL	[11:8]	rwh	Actual Prescaler Value See Table 22-8 . Writing into this register is only possible when the prescaler is stopped. When the floating prescaler mode is not used, this value is equal to the CC8yPSC.PSIV .
0	[7:4], [15:12], [31:16]	r	Reserved Read always returns 0.

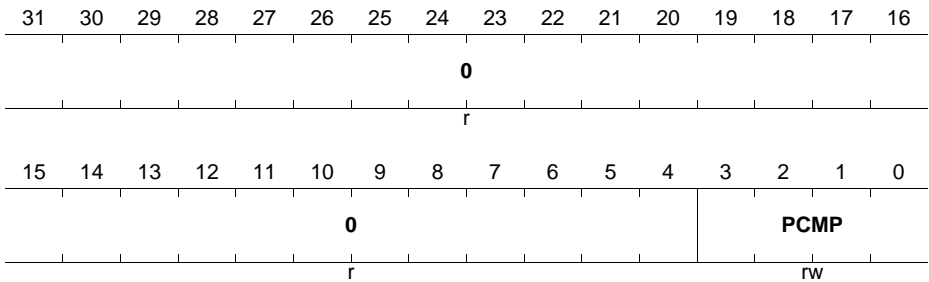
CC8yFPCS

This register contains the value that is going to be transferred to the [CC8yFPC.PCMP](#) field within the next shadow transfer update.

Capture/Compare Unit 8 (CCU8)

CC8yFPCS (y = 0 - 3)

Floating Prescaler Shadow $(012C_H + 0100_H * y)$ **Reset Value: 00000000_H**



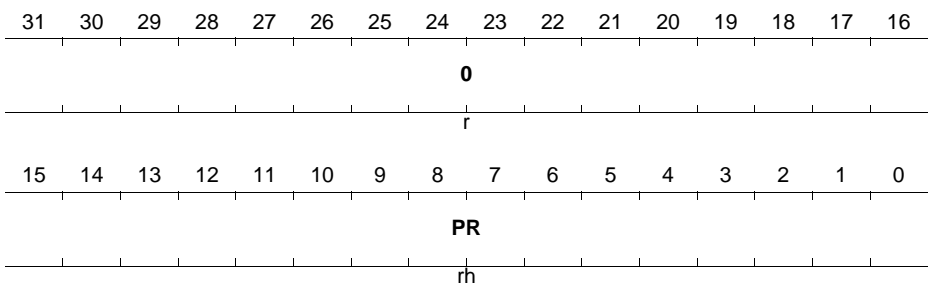
Field	Bits	Type	Description
PCMP	[3:0]	rw	Floating Prescaler Shadow Compare Value This field contains the value that is going to be set on the CC8yFPC.PCMP within the next shadow transfer. See Table 22-8 .
0	[31:4]	r	Reserved Read always returns 0.

CC8yPR

This register contains the actual value for the timer period.

CC8yPR (y = 0 - 3)

Timer Period Value $(0130_H + 0100_H * y)$ **Reset Value: 00000000_H**



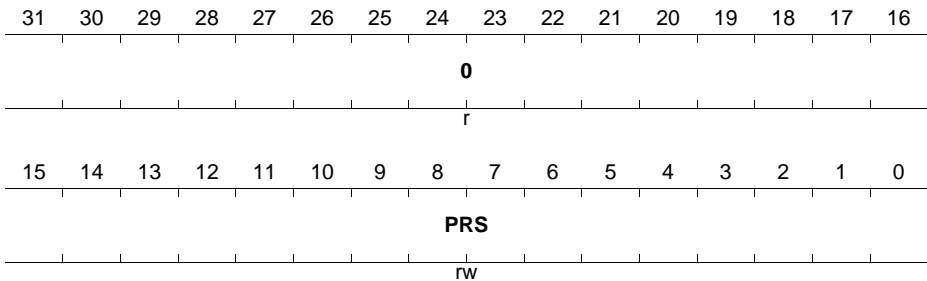
Field	Bits	Type	Description
PR	[15:0]	rh	Period Register Contains the value of the timer period.
0	[31:16]	r	Reserved A read always returns 0.

CC8yPRS

This register contains the value for the timer period that is going to be transferred into the **CC8yPR.PR** field when the next shadow transfer occurs.

CC8yPRS (y = 0 - 3)

Timer Shadow Period Value $(0134_H + 0100_H * y)$ **Reset Value: 00000000_H**



Field	Bits	Type	Description
PRS	[15:0]	rw	Period Register Contains the value of the timer period, that is going to be passed into the CC8yPR.PR field when the next shadow transfer occurs.
0	[31:16]	r	Reserved A read always returns 0.

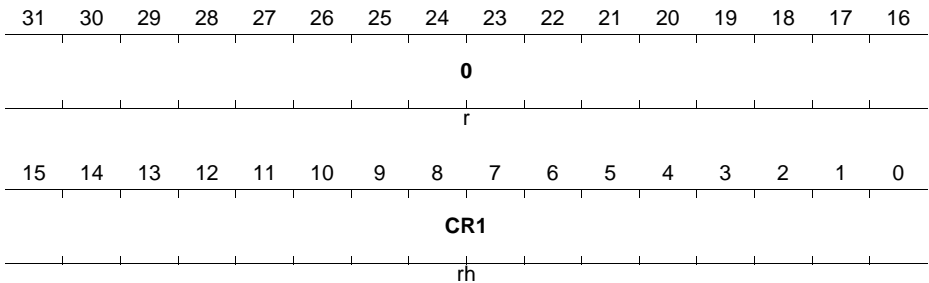
CC8yCR1

This register contains the value for the timer comparison of channel 1.

Capture/Compare Unit 8 (CCU8)

CC8yCR1 (y = 0 - 3)

Channel 1 Compare Value **(0138_H + 0100_H * y)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
CR1	[15:0]	rh	<p>Compare Register for Channel 1 Contains the value for the timer comparison. A write always addresses the shadow register, while a read returns the actual value.</p> <p><i>Note: In Capture Mode when a external signal is selected for capturing the timer value into the capture registers 0 and 1, the CR is not accessible for writing. A read always returns 0.</i></p>
0	[31:16]	r	<p>Reserved A read always returns 0.</p>

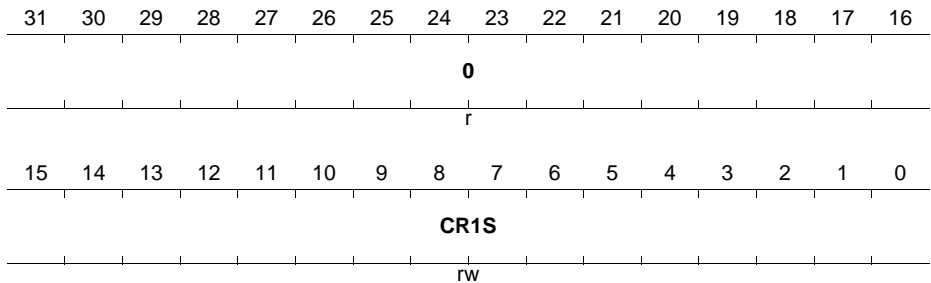
CC8yCR1S

This register contains the value that is going to be loaded into the **CC8yCR1.CR** field when the next shadow transfer occurs.

Capture/Compare Unit 8 (CCU8)

CC8yCR1S (y = 0 - 3)

Channel 1 Compare Shadow Value(013C_H + 0100_H * y) Reset Value: 00000000_H



Field	Bits	Type	Description
CR1S	[15:0]	rw	<p>Shadow Compare Register for Channel 1</p> <p>Contains the value for the timer comparison, that is going to be passed into the CC8yCR1.CR1 field when the next shadow transfer occurs.</p> <p><i>Note: In Capture Mode when a external signal is selected for capturing the timer value into the capture registers 0 and 1, the CR is not accessible for writing. A read always returns 0.</i></p>
0	[31:16]	r	<p>Reserved</p> <p>A read always returns 0.</p>

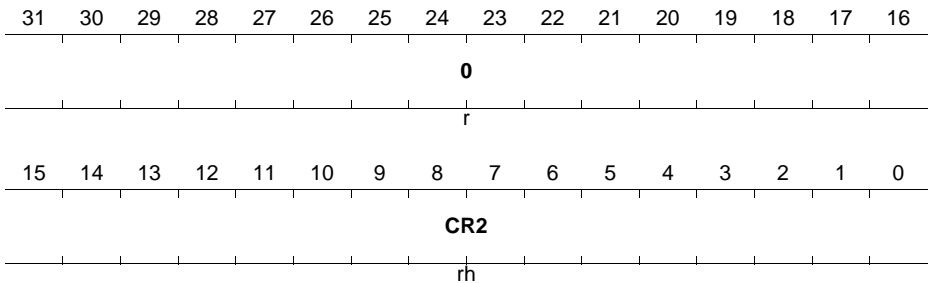
CC8yCR2

This register contains the value for the timer comparison of channel 2.

Capture/Compare Unit 8 (CCU8)

CC8yCR2 (y = 0 - 3)

Channel 2 Compare Value **(0140_H + 0100_H * y)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
CR2	[15:0]	rh	<p>Compare Register for Channel 2 Contains the value for the timer comparison. A write always addresses the shadow register, while a read returns the actual value.</p> <p><i>Note: In Capture Mode when a external signal is selected for capturing the timer value into the capture registers 2 and 3, the CR is not accessible for writing. A read always returns 0.</i></p>
0	[31:16]	r	<p>Reserved A read always returns 0.</p>

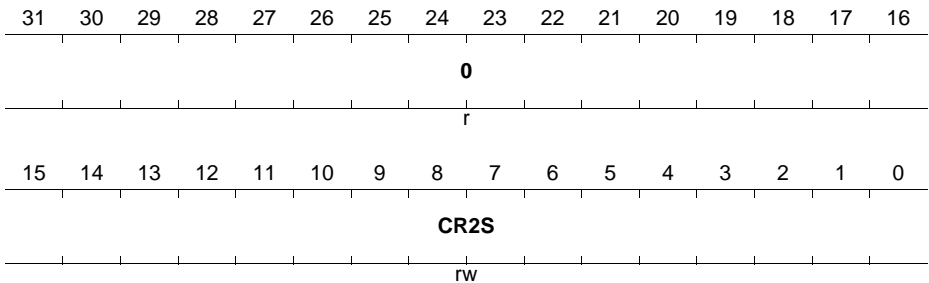
CC8yCR2S

This register contains the value that is going to be loaded into the **CC8yCR2.CR** field when the next shadow transfer occurs.

Capture/Compare Unit 8 (CCU8)

CC8yCR2S (y = 0 - 3)

Channel 2 Compare Shadow Value(0144_H + 0100_H * y) Reset Value: 00000000_H



Field	Bits	Type	Description
CR2S	[15:0]	rw	<p>Shadow Compare Register for Channel 2</p> <p>Contains the value for the timer comparison, that is going to be passed into the CC8yCR2.CR2 field when the next shadow transfer occurs.</p> <p><i>Note: In Capture Mode when a external signal is selected for capturing the timer value into the capture registers 2 and 3, the CR is not accessible for writing. A read always returns 0.</i></p>
0	[31:16]	r	<p>Reserved</p> <p>A read always returns 0.</p>

CC8yCHC

This register contains the configuration for the output connections from the two compare channels and the enable for the asymmetric mode.

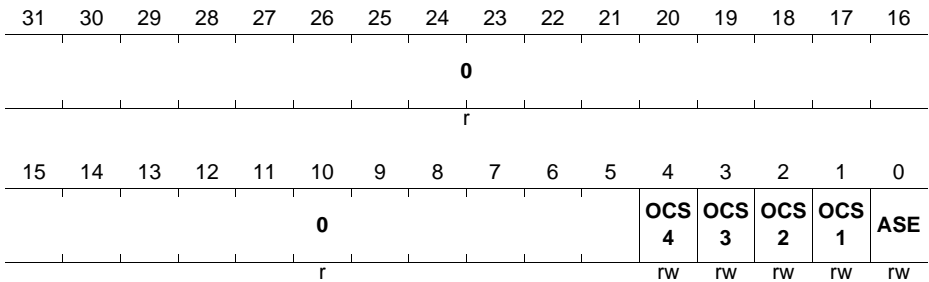
Capture/Compare Unit 8 (CCU8)

CC8yCHC (y = 0 - 3)

Channel Control

(0148_H + 0100_H * y)

Reset Value: 00000000_H



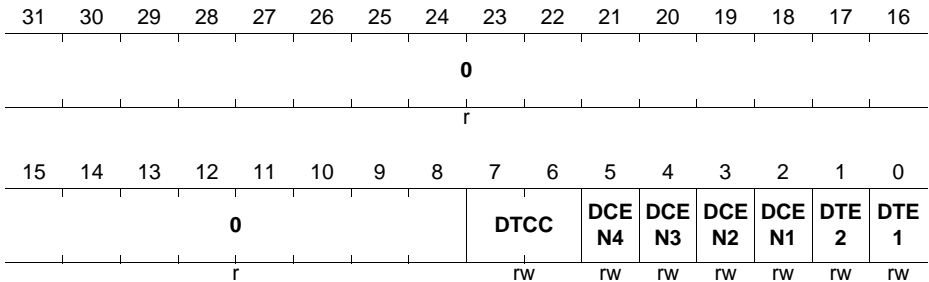
Field	Bits	Type	Description
ASE	0	rw	Asymmetric PWM mode Enable 0 _B Asymmetric PWM is disabled 1 _B Asymmetric PWM is enabled
OCS1	1	rw	Output selector for CCU8x.OUTy0 0 _B CC8yST1 signal path is connected to the CCU8x.OUTy0 1 _B Inverted CC8yST1 signal path is connected to the CCU8x.OUTy0
OCS2	2	rw	Output selector for CCU8x.OUTy1 0 _B Inverted CC8yST1 signal path is connected to the CCU8x.OUTy1 1 _B CC8yST1 signal path is connected to the CCU8x.OUTy1
OCS3	3	rw	Output selector for CCU8x.OUTy2 0 _B CC8yST2 signal path is connected to the CCU8x.OUTy2 1 _B Inverted CC8yST2 signal path is connected to the CCU8x.OUTy2
OCS4	4	rw	Output selector for CCU8x.OUTy3 0 _B Inverted CC8yST2 signal path is connected to the CCU8x.OUTy3 1 _B CC8yST2 signal path is connected to the CCU8x.OUTy3
0	[31:5]	r	Reserved A read access always returns 0

CC8yDTC

This register contains the configuration for the dead time generator.

CC8yDTC (y = 0 - 3)

Dead Time Control (014C_H + 0100_H * y) **Reset Value: 00000000_H**



Field	Bits	Type	Description
DTE1	0	rw	Dead Time Enable for Channel 1 This field enables the dead time counter for the compare channel 1. 0 _B Dead Time for channel 1 is disabled 1 _B Dead Time for channel 1 is enabled
DTE2	1	rw	Dead Time Enable for Channel 2 This field enables the dead time counter for the compare channel 2. 0 _B Dead Time for channel 2 is disabled 1 _B Dead Time for channel 2 is enabled
DCEN1	2	rw	Dead Time Enable for CC8yST1 0 _B Dead Time for CC8yST1 path is disabled 1 _B Dead Time for CC8yST1 path is enabled
DCEN2	3	rw	Dead Time Enable for inverted CC8yST1 0 _B Dead Time for inverted CC8yST1 path is disabled 1 _B Dead Time for inverted CC8yST1 path is enabled
DCEN3	4	rw	Dead Time Enable for CC8yST2 0 _B Dead Time for CC8yST2 path is disabled 1 _B Dead Time for CC8yST2 path is enabled

Capture/Compare Unit 8 (CCU8)

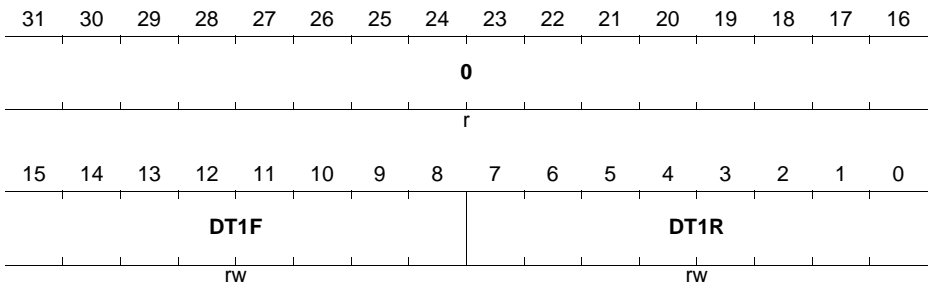
Field	Bits	Type	Description
DCEN4	5	rw	Dead Time Enable for inverted CC8yST2 0_B Dead Time for inverted CC8yST2 path is disabled 1_B Dead Time for inverted CC8yST2 path is enabled
DTCC	[7:6]	rw	Dead Time clock control This field controls the prescaler clock configuration for the dead time counters. 00_B f_{clk} 01_B $f_{\text{clk}}/2$ 10_B $f_{\text{clk}}/4$ 11_B $f_{\text{clk}}/8$
0	[15:8], [31:16]	r	Reserved A read always returns 0.

CC8yDC1R

This register contains the dead time value for the compare channel 1.

CC8yDC1R (y = 0 - 3)

Channel 1 Dead Time Values ($0150_H + 0100_H * y$) **Reset Value: 00000000_H**



Field	Bits	Type	Description
DT1R	[7:0]	rw	Rise Value for Dead Time of Channel 1 This field contains the delay value that is applied every time that a 0 to 1 transition occurs in the CC8yST1.

Capture/Compare Unit 8 (CCU8)

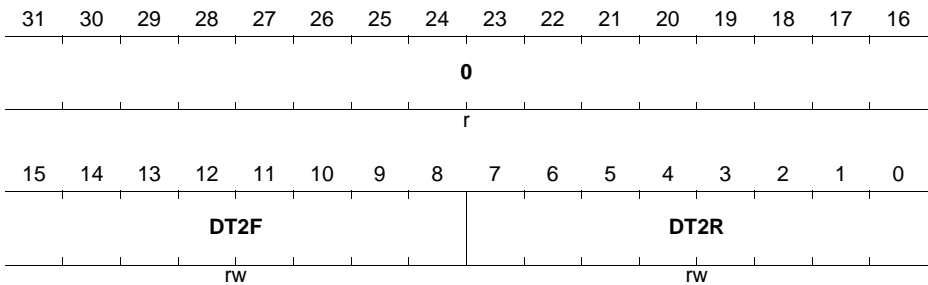
Field	Bits	Type	Description
DT1F	[15:8]	rw	Fall Value for Dead Time of Channel 1 This field contains the delay value that is applied every time that a 1 to 0 transition occurs in the CC8yST1.
0	[31:16]	r	Reserved A read access always returns 0

CC8yDC2R

This register contains the dead time value for the compare channel 2.

CC8yDC2R (y = 0 - 3)

Channel 2 Dead Time Values (0154_H + 0100_H * y) **Reset Value: 00000000_H**



Field	Bits	Type	Description
DT2R	[7:0]	rw	Rise Value for Dead Time of Channel 2 This field contains the delay value that is applied every time that a 0 to 1 transition occurs in the CC8yST2.
DT2F	[15:8]	rw	Fall Value for Dead Time of Channel 2 This field contains the delay value that is applied every time that a 1 to 0 transition occurs in the CC8yST2.
0	[31:16]	r	Reserved A read access always returns 0

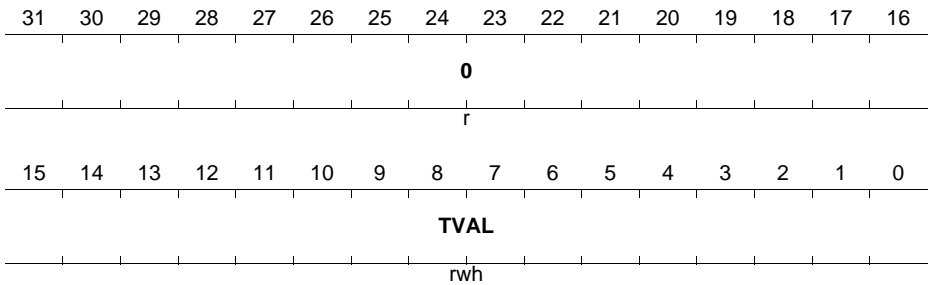
CC8yTIMER

This register contains the current value of the timer.

Capture/Compare Unit 8 (CCU8)

CC8yTIMER (y = 0 - 3)

Timer Value $(0170_H + 0100_H * y)$ **Reset Value: 00000000_H**



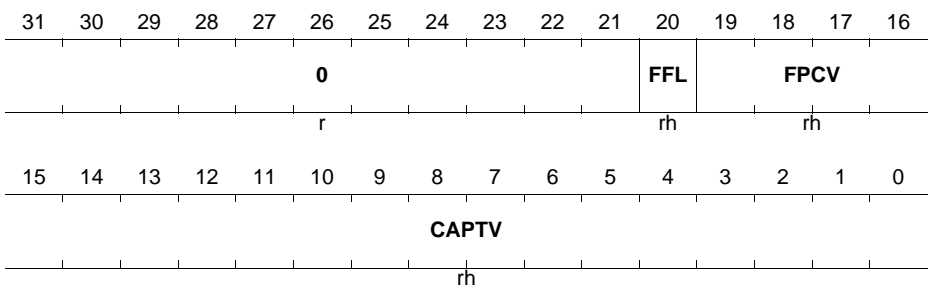
Field	Bits	Type	Description
TVALL	[15:0]	rwh	Timer Value This field contains the actual value of the timer. A write access is only possible when the timer is stopped.
0	[31:16]	r	Reserved A read access always returns 0

CC8yC0V

This register contains the values associated with the Capture 0 field.

CC8yC0V (y = 0 - 3)

Capture Register 0 $(0174_H + 0100_H * y)$ **Reset Value: 00000000_H**



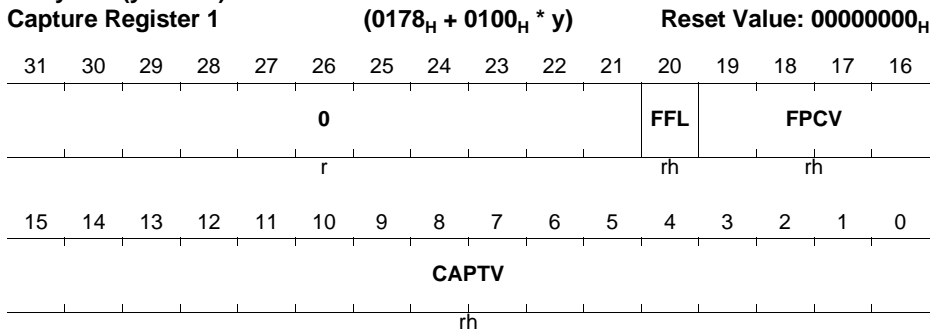
Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
CAPT	[15:0]	rh	Capture Value This field contains the capture register 0 value. See Figure 22-44 . In compare mode a read access always returns 0.
FPCV	[19:16]	rh	Prescaler Value This field contains the prescaler value when the time of the capture event into the capture register 0. In compare mode a read access always returns 0.
FFL	20	rh	Full Flag This bit indicates if a new value was capture into the capture register 0 after the last read access. See Figure 22-44 . In compare mode a read access always returns 0. 0_B No new value was captured into the specific capture register 1_B A new value was captured into the specific register
0	[31:21]	r	Reserved A read always returns 0

CC8yC1V

This register contains the values associated with the Capture 1 field.

CC8yC1V (y = 0 - 3)



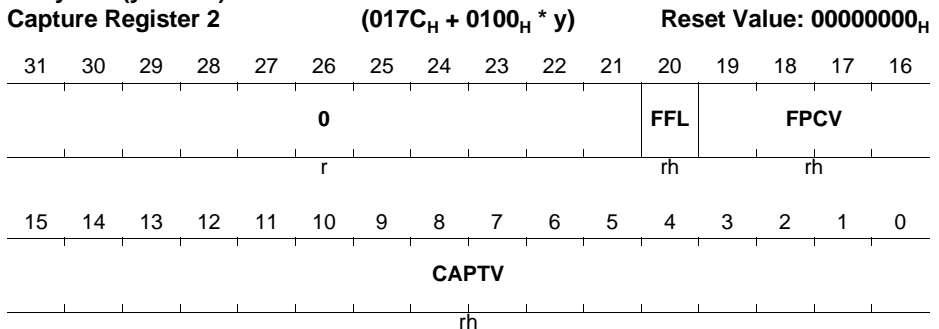
Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
CAPT	[15:0]	rh	Capture Value This field contains the capture register 1 value. See Figure 22-44 . In compare mode a read access always returns 0.
FPCV	[19:16]	rh	Prescaler Value This field contains the prescaler value when the time of the capture event into the capture register 1. In compare mode a read access always returns 0.
FFL	20	rh	Full Flag This bit indicates if a new value was capture into the capture register 1 after the last read access. See Figure 22-44 . In compare mode a read access always returns 0. 0_B No new value was captured into the specific capture register 1_B A new value was captured into the specific register
0	[31:21]	r	Reserved A read always returns 0

CC8yC2V

This register contains the values associated with the Capture 2 field.

CC8yC2V (y = 0 - 3)



Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
CAPTV	[15:0]	rh	Capture Value This field contains the capture register 3 value. See Figure 22-44 . In compare mode a read access always returns 0.
FPCV	[19:16]	rh	Prescaler Value This field contains the prescaler value when the time of the capture event into the capture register 3. In compare mode a read access always returns 0.
FFL	20	rh	Full Flag This bit indicates if a new value was capture into the capture register 3 after the last read access. See Figure 22-44 . In compare mode a read access always returns 0. 0_B No new value was captured into the specific capture register 1_B A new value was captured into the specific register
0	[31:21]	r	Reserved A read always returns 0

CC8yINTS

This register contains the status of all interrupt sources.

CC8yINTS (y = 0 - 3)

Interrupt Status										(01A0_H + 0100_H * y)					Reset Value: 00000000_H				
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
0																			
r																			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0				TRP	E2A	E1A	E0A	0			CMD	CMU	CMD	CMU	OMD	PMU			
				F	S	S	S				2S	2S	1S	1S	S	S			
r				rh	rh	rh	rh	r			rh	rh	rh	rh	rh	rh			

Field	Bits	Type	Description
PMUS	0	rh	Period Match while Counting Up 0_B Period match while counting up not detected 1_B Period match while counting up detected
OMDS	1	rh	One Match while Counting Down 0_B One match while counting down not detected 1_B One match while counting down detected
CMU1S	2	rh	Channel 1 Compare Match while Counting Up 0_B Compare match while counting up not detected 1_B Compare match while counting up detected
CMD1S	3	rh	Channel 1 Compare Match while Counting Down 0_B Compare match while counting down not detected 1_B Compare match while counting down detected
CMU2S	4	rh	Channel 2 Compare Match while Counting Up 0_B Compare match while counting up not detected 1_B Compare match while counting up detected
CMD2S	5	rh	Channel 2 Compare Match while Counting Down 0_B Compare match while counting down not detected 1_B Compare match while counting down detected
E0AS	8	rh	Event 0 Detection Status Depending on the user selection on the CC8yINS.EV0EM , this bit can be set when a rising, falling or both transitions are detected. 0_B Event 0 not detected 1_B Event 0 detected
E1AS	9	rh	Event 1 Detection Status Depending on the user selection on the CC8yINS.EV1EM , this bit can be set when a rising, falling or both transitions are detected. 0_B Event 1 not detected 1_B Event 1 detected

Capture/Compare Unit 8 (CCU8)

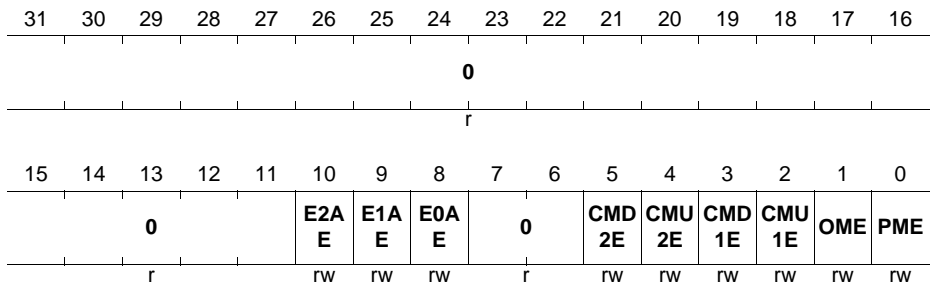
Field	Bits	Type	Description
E2AS	10	rh	Event 2 Detection Status Depending on the user selection on the CC8yINS.EV1EM , this bit can be set when a rising, falling or both transitions are detected. 0_B Event 2 not detected 1_B Event 2 detected <i>Note: If this event is linked with the TRAP function, this field is automatically cleared when the slice exits the Trap State.</i>
TRPF	11	rh	Trap Flag Status This field contains the status of the Trap Flag.
0	[7:6], [31:12]	r	Reserved A read always returns 0.

CC8yINTE

Through this register it is possible to enable or disable the specific interrupt source(s).

CC8yINTE (y = 0 - 3)

Interrupt Enable Control (**01A4_H + 0100_H * y**) **Reset Value: 00000000_H**



Field	Bits	Type	Description
PME	0	rw	Period match while counting up enable Setting this bit to 1_B enables the generation of an interrupt pulse every time a period match while counting up occurs. 0_B Period Match interrupt is disabled 1_B Period Match interrupt is enabled

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
OME	1	rw	<p>One match while counting down enable Setting this bit to 1_B enables the generation of an interrupt pulse every time an one match while counting down occurs.</p> <p>0_B One Match interrupt is disabled 1_B One Match interrupt is enabled</p>
CMU1E	2	rw	<p>Channel 1 Compare match while counting up enable Setting this bit to 1_B enables the generation of an interrupt pulse every time a compare match while counting up occurs.</p> <p>0_B Compare Match while counting up interrupt is disabled 1_B Compare Match while counting up interrupt is enabled</p>
CMD1E	3	rw	<p>Channel 1 Compare match while counting down enable Setting this bit to 1_B enables the generation of an interrupt pulse every time a compare match while counting down occurs.</p> <p>0_B Compare Match while counting down interrupt is disabled 1_B Compare Match while counting down interrupt is enabled</p>
CMU2E	4	rw	<p>Channel 2 Compare match while counting up enable Setting this bit to 1_B enables the generation of an interrupt pulse every time a compare match while counting up occurs.</p> <p>0_B Compare Match while counting up interrupt is disabled 1_B Compare Match while counting up interrupt is enabled</p>

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
CMD2E	5	rw	Channel 2 Compare match while counting down enable Setting this bit to 1 _B enables the generation of an interrupt pulse every time a compare match while counting down occurs. 0 _B Compare Match while counting down interrupt is disabled 1 _B Compare Match while counting down interrupt is enabled
E0AE	8	rw	Event 0 interrupt enable Setting this bit to 1 _B enables the generation of an interrupt pulse every time that Event 0 is detected. 0 _B Event 0 detection interrupt is disabled 1 _B Event 0 detection interrupt is enabled
E1AE	9	rw	Event 1 interrupt enable Setting this bit to 1 _B enables the generation of an interrupt pulse every time that Event 1 is detected. 0 _B Event 1 detection interrupt is disabled 1 _B Event 1 detection interrupt is enabled
E2AE	10	rw	Event 2 interrupt enable Setting this bit to 1 _B enables the generation of an interrupt pulse every time that Event 2 is detected. 0 _B Event 2 detection interrupt is disabled 1 _B Event 2 detection interrupt is enabled
0	[7:6], [31:11]	r	Reserved A read always returns 0

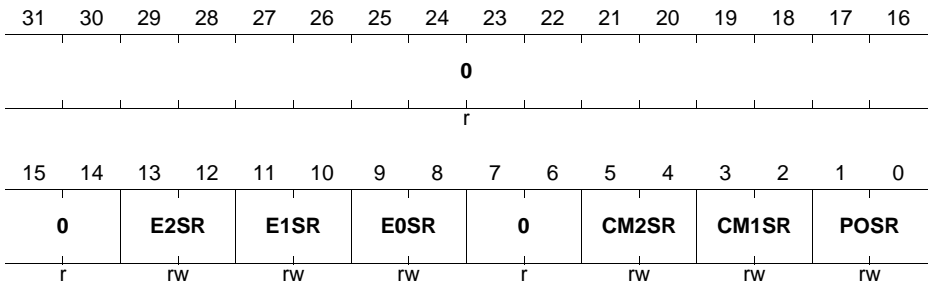
CC8ySRS

Through this register it is possible to select to which service request line each interrupt source is forwarded.

Capture/Compare Unit 8 (CCU8)

CC8ySRS (y = 0 - 3)

Service Request Selector (01A8_H + 0100_H * y) Reset Value: 00000000_H



Field	Bits	Type	Description
POSR	[1:0]	rw	<p>Period/One match Service request selector</p> <p>This field selects to which slice Service request line, the interrupt(s) generated by the Period match while counting up and One match while counting down are going to be forward.</p> <p>00_B Forward to CC8ySR0 01_B Forward to CC8ySR1 10_B Forward to CC8ySR2 11_B Forward to CC8ySR3</p>
CM1SR	[3:2]	rw	<p>Channel 1 Compare match Service request selector</p> <p>This field selects to which slice Service request line, the interrupt(s) generated by the Compare match, of channel 1, while counting up and Compare match while counting down are going to be forward.</p> <p>00_B Forward to CC8ySR0 01_B Forward to CC8ySR1 10_B Forward to CC8ySR2 11_B Forward to CC8ySR3</p>

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
CM2SR	[5:4]	rw	<p>Channel 2 Compare match Service request selector</p> <p>This field selects to which slice Service request line, the interrupt(s) generated by the Compare match, of channel 2, while counting up and Compare match while counting down are going to be forward.</p> <p>00_B Forward to CC8ySR0 01_B Forward to CC8ySR1 10_B Forward to CC8ySR2 11_B Forward to CC8ySR3</p>
E0SR	[9:8]	rw	<p>Event 0 Service request selector</p> <p>This field selects to which slice Service request line, the interrupt generated by the Event 0 detection are going to be forward.</p> <p>00_B Forward to CCvySR0 01_B Forward to CC8ySR1 10_B Forward to CC8ySR2 11_B Forward to CC8ySR3</p>
E1SR	[11:10]	rw	<p>Event 1 Service request selector</p> <p>This field selects to which slice Service request line, the interrupt generated by the Event 1 detection are going to be forward.</p> <p>00_B Forward to CC8ySR0 01_B Forward to CC8ySR1 10_B Forward to CC8ySR2 11_B Forward to CC8ySR3</p>
E2SR	[13:12]	rw	<p>Event 2 Service request selector</p> <p>This field selects to which slice Service request line, the interrupt generated by the Event 2 detection are going to be forward.</p> <p>00_B Forward to CC8ySR0 01_B Forward to CCvySR1 10_B Forward to CC8ySR2 11_B Forward to CC8ySR3</p>
0	[7:6], [31:14]	r	<p>Reserved</p> <p>Read always returns 0.</p>

CC8ySWS

Through this register it is possible for the SW to set a specific interrupt status flag.

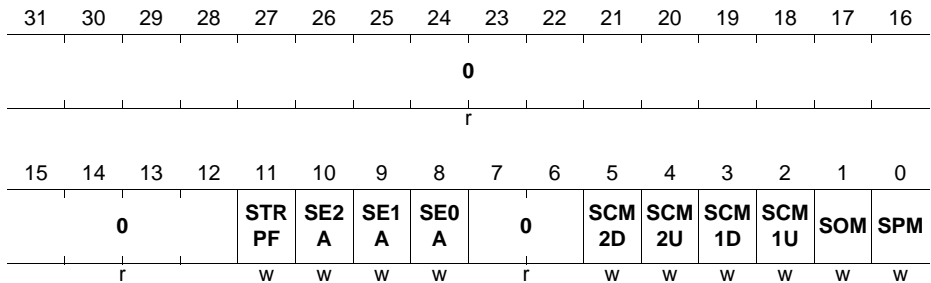
Capture/Compare Unit 8 (CCU8)

CC8ySWS (y = 0 - 3)

Interrupt Status Set

(01AC_H + 0100_H * y)

Reset Value: 00000000_H



Field	Bits	Type	Description
SPM	0	w	Period match while counting up set Writing a 1 _B into this field sets the CC8yINTS.PMUS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SOM	1	w	One match while counting down set Writing a 1 _B into this bit sets the CC8yINTS.OMDS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SCM1U	2	w	Channel 1 Compare match while counting up set Writing a 1 _B into this field sets the CC8yINTS.CMU1S bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SCM1D	3	w	Channel 1 Compare match while counting down set Writing a 1 _B into this bit sets the CC8yINTS.CMD1S bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SCM2U	4	w	Compare match while counting up set Writing a 1 _B into this field sets the CC8yINTS.CMU2S bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
SCM2D	5	w	Compare match while counting down set Writing a 1 _B into this bit sets the CC8yINTS.CMD2S bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SE0A	8	w	Event 0 detection set Writing a 1 _B into this bit sets the CC8yINTS.E0AS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SE1A	9	w	Event 1 detection set Writing a 1 _B into this bit sets the CC8yINTS.E1AS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SE2A	10	w	Event 2 detection set Writing a 1 _B into this bit sets the CC8yINTS.E2AS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
STRPF	11	w	Trap Flag status set Writing a 1 _B into this bit sets the CC8yINTS.TRPF bit. A read always returns 0.
0	[7:6], [31:12]	r	Reserved Read always returns 0

CC8ySWR

Through this register it is possible for the SW to clear a specific interrupt status flag.

CC8ySWR (y = 0 - 3)

Interrupt Status Clear (01B0_H + 0100_H * y) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			RTR	RE2	RE1	RE0	0			RCM	RCM	RCM	RCM	ROM	RPM
			PF	A	A	A				2D	2U	1D	1U		
			w	w	w	w				w	w	w	w	w	w
			r												

Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
RPM	0	w	Period match while counting up clear Writing a 1 _B into this field clears the CC8yINTS .PMUS bit. A read always returns 0.
ROM	1	w	One match while counting down clear Writing a 1 into this bit clears the CC8yINTS .OMDS bit. A read always returns 0.
RCM1U	2	w	Channel 1 Compare match while counting up clear Writing a 1 _B into this field clears the CC8yINTS .CMU1S bit. A read always returns 0.
RCM1D	3	w	Channel 1 Compare match while counting down clear Writing a 1 _B into this bit clears the CC8yINTS .CMD1S bit. A read always returns 0.
RCM2U	4	w	Channel 2 Compare match while counting up clear Writing a 1 _B into this field clears the CC8yINTS .CMU2S bit. A read always returns 0.
RCM2D	5	w	Channel 2 Compare match while counting down clear Writing a 1 _B into this bit clears the CC8yINTS .CMD2S bit. A read always returns 0.
RE0A	8	w	Event 0 detection clear Writing a 1 _B into this bit clears the CC8yINTS .E0AS bit. A read always returns 0.
RE1A	9	w	Event 1 detection clear Writing a 1 _B into this bit clears the CC8yINTS .E1AS bit. A read always returns 0.
RE2A	10	w	Event 2 detection clear Writing a 1 _B into this bit clears the CC8yINTS .E2AS bit. A read always returns 0.
RTRPF	11	w	Trap Flag status clear Writing a 1 _B into this bit clears the CC8yINTS .TRPF bit. Not valid if CC8yTC .TRPEN = 1 _B and the Trap State is still active. A read always returns 0.

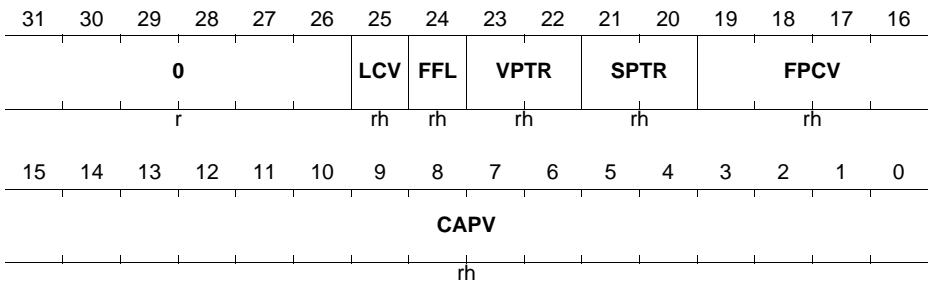
Capture/Compare Unit 8 (CCU8)

CC8yECRD0

Through this register it is possible to read back the FIFO structure of the capture function that is linked with the capture trigger 0. The read back is only valid if the **CC8yTC.ECM** = 1_B.

CC8yECRD0 (y = 0 - 3)

Extended Read Back 0 (01B_H + 0100_H * y) **Reset Value: 00000000_H**



Field	Bits	Type	Description
CAPV	[15:0]	rh	Timer Capture Value This field contains the timer captured value
FPCV	[19:16]	rh	Prescaler Capture value This field contains the value of the prescaler clock division associated with the specific CAPV field
SPTR	[21:20]	rh	Slice pointer This field indicates the slice index in which the value was captured. 00 _B CC80 01 _B CC81 10 _B CC82 11 _B CC83
VPTR	[23:22]	rh	Capture register pointer This field indicates the capture register index in which the value was captured. 00 _B Capture register 0 01 _B Capture register 1 10 _B Capture register 2 11 _B Capture register 3

Capture/Compare Unit 8 (CCU8)

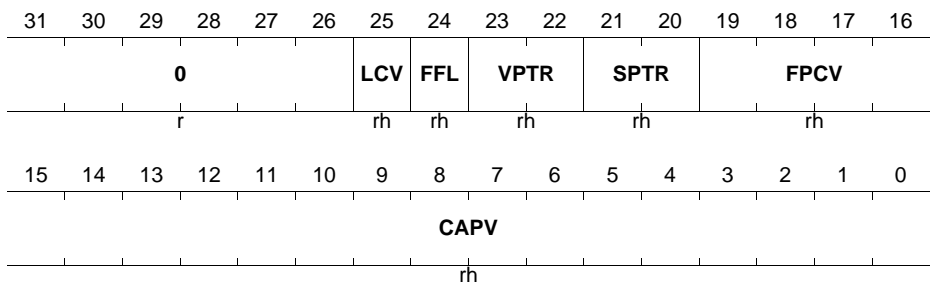
Field	Bits	Type	Description
FFL	24	rh	Full Flag This bit indicates if the associated capture register contains a new value. 0_B No new value was captured into this register 1_B A new value has been captured into this register
LCV	25	rh	Lost Capture Value This field indicates if between two reads of the ECRD0 a capture trigger occurred while the FIFO structure was full. If a capture trigger occurred between two reads than a capture value was lost. This field is automatically cleared by the HW whenever a read to the ECRD occurs. 0_B No capture was lost 1_B A capture was lost
0	[31:26]	r	Reserved Read always returns 0

CC8yECRD1

Through this register it is possible to read back the FIFO structure of the capture function that is linked with the capture trigger 1. The read back is only valid if the **CC8yTC.ECM** = 1_B .

CC8yECRD1 (y = 0 - 3)

Extended Read Back 1 ($01BC_H + 0100_H * y$) **Reset Value: 00000000_H**



Capture/Compare Unit 8 (CCU8)

Field	Bits	Type	Description
CAPV	[15:0]	rh	Timer Capture Value This field contains the timer captured value
FPCV	[19:16]	rh	Prescaler Capture value This field contains the value of the prescaler clock division associated with the specific CAPV field
SPTR	[21:20]	rh	Slice pointer This field indicates the slice index in which the value was captured. 00 _B CC80 01 _B CC81 10 _B CC82 11 _B CC83
VPTR	[23:22]	rh	Capture register pointer This field indicates the capture register index in which the value was captured. 00 _B Capture register 0 01 _B Capture register 1 10 _B Capture register 2 11 _B Capture register 3
FFL	24	rh	Full Flag This bit indicates if the associated capture register contains a new value. 0 _B No new value was captured into this register 1 _B A new value has been captured into this register
LCV	25	rh	Lost Capture Value This field indicates if between two reads of the ECRD0 a capture trigger occurred while the FIFO structure was full. If a capture trigger occurred between two reads than a capture value was lost. This field is automatically cleared by the HW whenever a read to the ECRD occurs. 0 _B No capture was lost 1 _B A capture was lost
0	[31:26]	r	Reserved Read always returns 0

22.8 Interconnects

The tables that refer to the “global pins” are the ones that contain the inputs/outputs of each module that are common to all slices.

The GPIO mapping is available at the Ports unit.

22.8.1 CCU80 Pins

Table 22-14 CCU80 Pin Connections

Global Inputs/Outputs	I/O	Connected To	Description
CCU80.MCLK	I	SCU.CCUCLK	Kernel clock
CCU80.CLKA	I (s)	ERU1.IOUT0	another count source for the prescaler
CCU80.CLKB	I (s)	ERU1.IOUT1	another count source for the prescaler
CCU80.CLKC	I (s)	0	another count source for the prescaler
CCU80.MCSS	I	NOT CONNECTED	Multi pattern sync with shadow transfer trigger
CCU80.IGBTA	I	CCU40.ST3;	Parity Checker delay finish trigger
CCU80.IGBTB	I	CCU40.SR3;	Parity Checker delay finish trigger
CCU80.IGBTC	I	NOT CONNECTED	Parity Checker delay finish trigger
CCU80.IGBTD	I	NOT CONNECTED	Parity Checker delay finish trigger
CCU80.IGBTO	O	CCU40.IN3H; CCU42.IN0H;	Parity Checker delay start trigger
CCU80.SR0	O	NVIC; DMA; VADC.G0REQGTM; VADC.G1REQGTM; VADC.BGREQGTM;	Service request line

Table 22-14 CCU80 Pin Connections

Global Inputs/Outputs	I/O	Connected To	Description
CCU80.SR1	O	NVIC; DMA; DAC.TRIGGER[0]; VADC.G0REQGTN; VADC.G1REQGTN; U0C0.DX2F; U0C1.DX2F; VADC.BGREQGTN	Service request line
CCU80.SR2	O	NVIC; VADC.G0REQTRI; VADC.G1REQTRI; VADC.BGREQTRI;	Service request line
CCU80.SR3	O	NVIC; VADC.G0REQTRJ; VADC.G1REQTRJ; VADC.BGREQTRJ;	Service request line

Table 22-15 CCU80 - CC80 Pin Connections

Input/Output	I/O	Connected To	Description
CCU80.IN0A	I (s)	PORTS	General purpose function
CCU80.IN0B	I (s)	PORTS	General purpose function
CCU80.IN0C	I (s)	PORTS	General purpose function
CCU80.IN0D	I (s)	NOT CONNECTED	General purpose function
CCU80.IN0E	I (s)	NOT CONNECTED	General purpose function
CCU80.IN0F	I (s)	VADC.G0SR3	General purpose function
CCU80.IN0G	I (s)	ERU1.IOUT0	General purpose function
CCU80.IN0H	I (s)	SCU.GSC80	General purpose function
CCU80.IN0I	I (s)	VADC.G0BFL0	General purpose function
CCU80.IN0J	I (s)	ERU1.PDOUT0	General purpose function
CCU80.IN0K	I (s)	CCU40.SR3	General purpose function
CCU80.IN0L	I (s)	NOT CONNECTED	General purpose function
CCU80.IN0M	I (s)	CCU80.ST0	General purpose function
CCU80.IN0N	I (s)	CCU80.ST1	General purpose function

Capture/Compare Unit 8 (CCU8)

Table 22-15 CCU80 - CC80 Pin Connections

Input/Output	I/O	Connected To	Description
CCU80.IN00	I (s)	CCU80.ST2	General purpose function
CCU80.IN0P	I (s)	CCU80.ST3	General purpose function
CCU80.MCI00	I	NOT CONNECTED	Multi Channel pattern input for CCST1
CCU80.MCI01	I	NOT CONNECTED	Multi Channel pattern input for NOT(CCST1)
CCU80.MCI02	I	NOT CONNECTED	Multi Channel pattern input for CCST2
CCU80.MCI03	I	NOT CONNECTED	Multi Channel pattern input for NOT(CCST2)
CCU80.OUT00	O	PORTS	Slice compare output from channel 1. Can be the CCST1 or NOT(CCST1) path
CCU80.OUT01	O	PORTS	Slice compare output from channel 1. Can be the CCST1 or NOT(CCST1) path
CCU80.OUT02	O	PORTS	Slice compare output from channel 2. Can be the CCST2 or NOT(CCST2) path
CCU80.OUT03	O	PORTS	Slice compare output from channel 2. Can be the CCST2 or NOT(CCST2) path
CCU80.GP00	O	NOT CONNECTED	Selected signal for event 0
CCU80.GP01	O	NOT CONNECTED	Selected signal for event 1
CCU80.GP02	O	NOT CONNECTED	Selected signal for event 2
CCU80.ST0	O	ERU1.0B1	Output of the status bit multiplexer. It can be CCST1 or CCST2
CCU80.ST0A	O	NOT CONNECTED	Channel 1 status bit: CCST1
CCU80.ST0B	O	NOT CONNECTED	Channel 2 status bit: CCST2
CCU80.PS0	O	NOT CONNECTED	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

Table 22-16 CCU80 - CC81 Pin Connections

Input/Output	I/O	Connected To	Description
CCU80.IN1A	I (s)	PORTS	General purpose function
CCU80.IN1B	I (s)	PORTS	General purpose function
CCU80.IN1C	I (s)	PORTS	General purpose function
CCU80.IN1D	I (s)	NOT CONNECTED	General purpose function
CCU80.IN1E	I (s)	NOT CONNECTED	General purpose function
CCU80.IN1F	I (s)	ERU1.PDOUT1	General purpose function
CCU80.IN1G	I (s)	ERU1.IOOUT1	General purpose function
CCU80.IN1H	I (s)	SCU.GSC80	General purpose function
CCU80.IN1I	I (s)	VADC.G0BFL1	General purpose function
CCU80.IN1J	I (s)	ERU1.PDOUT0	General purpose function
CCU80.IN1K	I (s)	CCU41.SR3	General purpose function
CCU80.IN1L	I (s)	CCU81.SR3	General purpose function
CCU80.IN1M	I (s)	CCU80.ST0	General purpose function
CCU80.IN1N	I (s)	CCU80.ST1	General purpose function
CCU80.IN1O	I (s)	CCU80.ST2	General purpose function
CCU80.IN1P	I (s)	CCU80.ST3	General purpose function
CCU80.MCI10	I	NOT CONNECTED	Multi Channel pattern input for CCST1
CCU80.MCI11	I	NOT CONNECTED	Multi Channel pattern input for NOT(CCST1)
CCU80.MCI12	I	NOT CONNECTED	Multi Channel pattern input for CCST2
CCU80.MCI13	I	NOT CONNECTED	Multi Channel pattern input for NOT(CCST2)
CCU80.OUT10	O	PORTS	Slice compare output from channel 1. Can be the CCST1 or NOT(CCST1) path
CCU80.OUT11	O	PORTS	Slice compare output from channel 1. Can be the CCST1 or NOT(CCST1) path

Capture/Compare Unit 8 (CCU8)

Table 22-16 CCU80 - CC81 Pin Connections

Input/Output	I/O	Connected To	Description
CCU80.OUT12	O	PORTS	Slice compare output from channel 2. Can be the CCST2 or NOT(CCST2) path
CCU80.OUT13	O	PORTS	Slice compare output from channel 2. Can be the CCST2 or NOT(CCST2) path
CCU80.GP10	O	NOT CONNECTED	Selected signal for event 0
CCU80.GP11	O	NOT CONNECTED	Selected signal for event 1
CCU80.GP12	O	NOT CONNECTED	Selected signal for event 2
CCU80.ST1	O	ERU1.1B1	Output of the status bit multiplexer. It can be CCST1 or CCST2
CCU80.ST1A	O	NOT CONNECTED	Channel 1 status bit: CCST1
CCU80.ST1B	O	NOT CONNECTED	Channel 2 status bit: CCST2
CCU80.PS1	O	NOT CONNECTED	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

Table 22-17 CCU80 - CC82 Pin Connections

Input/Output	I/O	Connected To	Description
CCU80.IN2A	I (s)	PORTS	General purpose function
CCU80.IN2B	I (s)	PORTS	General purpose function
CCU80.IN2C	I (s)	PORTS	General purpose function
CCU80.IN2D	I (s)	NOT CONNECTED	General purpose function
CCU80.IN2E	I (s)	NOT CONNECTED	General purpose function
CCU80.IN2F	I (s)	ERU1.PDOUT2	General purpose function
CCU80.IN2G	I (s)	ERU1.IOOUT2	General purpose function
CCU80.IN2H	I (s)	SCU.GSC80	General purpose function
CCU80.IN2I	I (s)	VADC.G0BFL2	General purpose function
CCU80.IN2J	I (s)	ERU1.PDOUT0	General purpose function
CCU80.IN2K	I (s)	NOT CONNECTED	General purpose function

Capture/Compare Unit 8 (CCU8)

Table 22-17 CCU80 - CC82 Pin Connections

Input/Output	I/O	Connected To	Description
CCU80.IN2L	I (s)	NOT CONNECTED	General purpose function
CCU80.IN2M	I (s)	CCU80.ST0	General purpose function
CCU80.IN2N	I (s)	CCU80.ST1	General purpose function
CCU80.IN2O	I (s)	CCU80.ST2	General purpose function
CCU80.IN2P	I (s)	CCU80.ST3	General purpose function
CCU80.MCI20	I	NOT CONNECTED	Multi Channel pattern input for CCST1
CCU80.MCI21	I	NOT CONNECTED	Multi Channel pattern input for NOT(CCST1)
CCU80.MCI22	I	NOT CONNECTED	Multi Channel pattern input for CCST2
CCU80.MCI23	I	NOT CONNECTED	Multi Channel pattern input for NOT(CCST2)
CCU80.OUT20	O	PORTS	Slice compare output from channel 1. Can be the CCST1 or NOT(CCST1) path
CCU80.OUT21	O	PORTS	Slice compare output from channel 1. Can be the CCST1 or NOT(CCST1) path
CCU80.OUT22	O	PORTS	Slice compare output from channel 2. Can be the CCST2 or NOT(CCST2) path
CCU80.OUT23	O	PORTS	Slice compare output from channel 2. Can be the CCST2 or NOT(CCST2) path
CCU80.GP20	O	NOT CONNECTED	Selected signal for event 0
CCU80.GP21	O	NOT CONNECTED	Selected signal for event 1
CCU80.GP22	O	NOT CONNECTED	Selected signal for event 2
CCU80.ST2	O	ERU1.2B1	Output of the status bit multiplexer. It can be CCST1 or CCST2
CCU80.ST2A	O	NOT CONNECTED	Channel 1 status bit: CCST1

Capture/Compare Unit 8 (CCU8)

Table 22-17 CCU80 - CC82 Pin Connections

Input/Output	I/O	Connected To	Description
CCU80.ST2B	O	NOT CONNECTED	Channel 2 status bit: CCST2
CCU80.PS2	O	NOT CONNECTED	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

Table 22-18 CCU80 - CC83 Pin Connections

Input/Output	I/O	Connected To	Description
CCU80.IN3A	I (s)	PORTS	General purpose function
CCU80.IN3B	I (s)	PORTS	General purpose function
CCU80.IN3C	I (s)	PORTS	General purpose function
CCU80.IN3D	I (s)	NOT CONNECTED	General purpose function
CCU80.IN3E	I (s)	NOT CONNECTED	General purpose function
CCU80.IN3F	I (s)	ERU1.PDOUT3	General purpose function
CCU80.IN3G	I (s)	ERU1.IOOUT3	General purpose function
CCU80.IN3H	I (s)	SCU.GSC80	General purpose function
CCU80.IN3I	I (s)	VADC.G0BFL3	General purpose function
CCU80.IN3J	I (s)	ERU1.PDOUT0	General purpose function
CCU80.IN3K	I (s)	NOT CONNECTED	General purpose function
CCU80.IN3L	I (s)	NOT CONNECTED	General purpose function
CCU80.IN3M	I (s)	CCU80.ST0	General purpose function
CCU80.IN3N	I (s)	CCU80.ST1	General purpose function
CCU80.IN3O	I (s)	CCU80.ST2	General purpose function
CCU80.IN3P	I (s)	CCU80.ST3	General purpose function
CCU80.MCI30	I	NOT CONNECTED	Multi Channel pattern input for CCST1
CCU80.MCI31	I	NOT CONNECTED	Multi Channel pattern input for NOT(CCST1)
CCU80.MCI32	I	NOT CONNECTED	Multi Channel pattern input for CCST2
CCU80.MCI33	I	NOT CONNECTED	Multi Channel pattern input for NOT(CCST2)

Capture/Compare Unit 8 (CCU8)

Table 22-18 CCU80 - CC83 Pin Connections

Input/Output	I/O	Connected To	Description
CCU80.OUT30	O	PORTS	Slice compare output from channel 1. Can be the CCST1 or NOT(CCST1) path
CCU80.OUT31	O	PORTS	Slice compare output from channel 1. Can be the CCST1 or NOT(CCST1) path
CCU80.OUT32	O	PORTS	Slice compare output from channel 2. Can be the CCST2 or NOT(CCST2) path
CCU80.OUT33	O	PORTS	Slice compare output from channel 2. Can be the CCST2 or NOT(CCST2) path
CCU80.GP30	O	NOT CONNECTED	Selected signal for event 0
CCU80.GP31	O	NOT CONNECTED	Selected signal for event 1
CCU80.GP32	O	NOT CONNECTED	Selected signal for event 2
CCU80.ST3	O	ERU1.3B1	Output of the status bit multiplexer. It can be CCST1 or CCST2
CCU80.ST3A	O	VADC.G0REQGTE; VADC.G1REQGTE; VADC.BGREQGTE;	Channel 1 status bit: CCST1
CCU80.ST3B	O	VADC.G0REQGTF; VADC.G1REQGTF; VADC.BGREQGTF;	Channel 2 status bit: CCST2
CCU80.PS3	O	NOT CONNECTED	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

General Purpose I/O Ports

23 General Purpose I/O Ports (PORTS)

The XMC4300 has many digital port pins which can be used as General Purpose I/Os (GPIO) and are connected to the on-chip peripheral units.

23.1 Overview

The PORTS provide a generic and flexible software and hardware interface for all standard digital I/Os. Each port slice has the same software interfaces for the operation as General Purpose I/O and it further provides the connectivity to the on-chip periphery and the control for the pad characteristics. [Table 23-1](#) gives an overview of the available PORTS and other pins in the different packages of the XMC4300:

Table 23-1 Port/Pin Overview

Function	LQFP-100	Note
P0	13	
P1	16	
P2	13	
P3	7	
P4	2	
P5	4	
P14	14	Analog/Digital Input only
P15	4	Analog/Digital Input only
Dedicated I/Os	12	HIB_IO, TMS, TCK, USB, VBUS, XTAL, RTC_XTAL, PORST
Analog Supply, Reference and Ground	4	VDDA, VSSA, VAREF, VAGND
Digital Supply	9	VDDP, VDDC, VBAT
Digital Ground	2	VSS, VSSO, must be connected to common V_{SS}
Exposed Die Pad	1	Must be connected to common V_{SS}

23.1.1 Features

This is a list of the main features of the PORTS:

- same generic register interface for each port pin, [Section 23.8](#)
- simple and robust software access for General Purpose I/O functionality, [Section 23.2](#)

General Purpose I/O Ports (PORTS)

- separate set and clear output control to avoid read-modify-write operations, [Section 23.8.5](#)
- direct input connections to on-chip peripherals, [Section 23.2.1](#)
- parallel input of the same pin to different peripherals possible, for example triggering a capture event in a CAPCOM unit and a Service Request via the ERU
- up to four alternate output connections from peripherals selectable, [Section 23.2.2](#)
- separate input and output path, which allows to evaluate the input while the output is active (feedback, plausability check)
- dedicated hardware-controlled interface for EtherCAT, SDMMC, LEDTS and QSPI with select option, [Section 23.3](#)
- programmable open-drain or push-pull output driver stage, [Section 23.8.1](#)
- programmable driver strength and slew rate, [Page 23-6](#)
- programmable weak pull-up and pull-down devices, [Section 23.8.1](#)
- programmable input inverter, [Section 23.2.1](#)
- programmable power-save behavior in Deep Sleep mode, [Section 23.4](#)
- defined power-up/power-fail behavior, [Section 23.6](#)
- Privilege Mode restricted access to configuration registers to avoid accidental modification
- disabling of digital input stage on shared analog input ports, [Section 23.5](#)

23.1.2 Block Diagram

Below is a figure with the generic structure of a digital port pin, split into the port slice with the control logic and the pad with the pull devices and the input and output stages, [Figure 23-1](#).

General Purpose I/O Ports (PORTS)

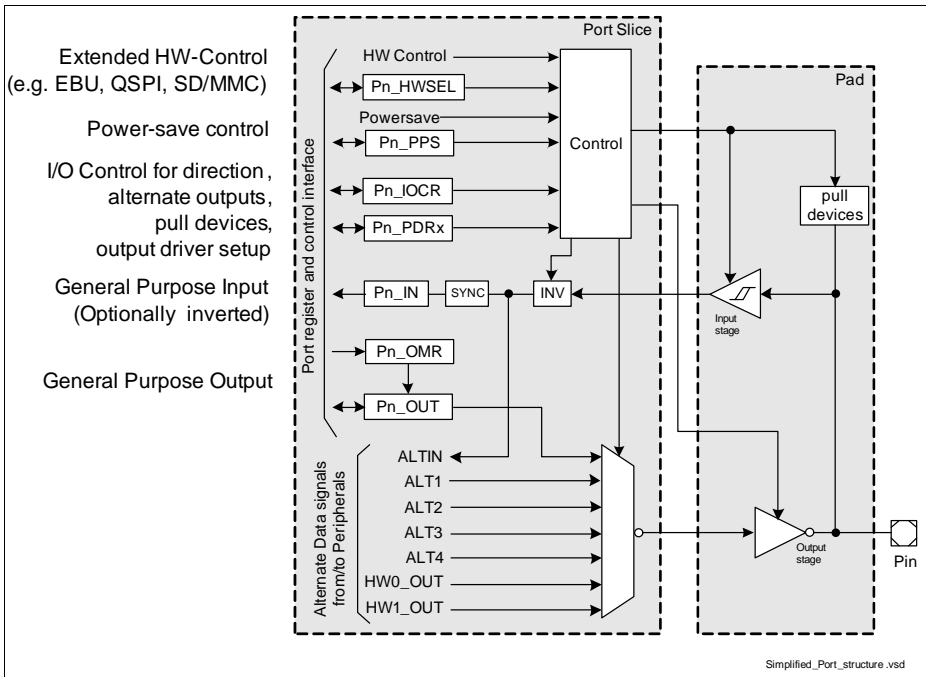


Figure 23-1 General Structure of a digital Port Pin

23.1.3 Definition of Terms

Some specific terms are used throughout this chapter:

- **Pin/Ball:** External connection of the device to the PCB.
- **Dedicated Pin:** A Pin with a dedicated function that is not under the control of the port logic (i.e. supply pins, $\overline{\text{PORST}}$).
- **Port Pin:** A pin under the control of the port logic (P0.1).
- **Port:** A group of up to 16 Port Pins sharing the same generic register set (P0).
- **Port Slice:** The “sum” of register bits and control logic used to control a port pin.
- **Pad:** Analog component containing the output driver, pull devices and input Schmitt-Trigger. Also interfaces the internal logic operating on V_{DDC} to the pad supply domain V_{DDP} .
- **GPIO:** General Purpose Input/Output. A port pin with the input and/or output function controlled by the application software.
- **Alternate Function:** Direct connection of a port pin with an on-chip peripheral.

23.2 GPIO and Alternate Function

The Ports can be operated as General Purpose Input/Outputs (GPIO) and with Alternate Functions of the on-chip peripheral, configured by the Port Input/Output Control Register (Pn_IOCR, [Section 23.8.1](#)). It selects between

- Direct or Inverted Input
 - with or without pull device
- Push-pull or Open-Drain Output driven by
 - Pn_OUT (GPIO)
 - selected peripheral output connections.

As GPIO the port pin is controlled by the application software, reading the input value by the Port Input register Pn_IN ([Section 23.8.6](#)) and/or defining the output value by the Output Modification Register Pn_OMR ([Section 23.8.5](#)). Output modification by Pn_OMR is preferred over the direct change of the output value with the Output register Pn_OUT ([Section 23.8.4](#)), as Pn_OMR allows the manipulation of individual port pins in a single access without “disturbing” other pins controlled by the same Pn_OUT register. If an application uses a GPIO as a bi-directional I/O line, register Pn_IOCR has to be written to switch between input and output functionality.

For the operation with Alternate Functions, the port pins are directly connected to input or output functions of the on-chip peripheral. This allows the peripheral to directly evaluate the input value or drive the output value of the port pin without further application software interaction after the initial configuration. The connection of alternate functions is used for control and communication interfaces, like a PWM from a CAPCOM unit or a SPI communication of a USIC channel. A detailed connectivity list of the peripherals to the port pins is given in the [Port I/O Functions](#) chapter. For specific functions, certain peripherals may also take direct control of “their” port pins, see [Hardware Controlled I/Os](#).

23.2.1 Input Operation

As an input, the actual voltage level at the port pin is translated into a logical 0_B or 1_B via a Schmitt-Trigger device within the pad. The resulting input value can be optionally inverted. As general purpose input the signal is synchronized and can be read with the Input register (Pn_IN, [Section 23.8.6](#)). Alternatively, the input can be connected to multiple on-chip peripherals via the ALTIN signal. Where necessary, these peripherals have internal controls to select the appropriate port pin with an input multiplexer stage, and will take care of synchronization and the further processing of the input signals (for more details on the input selection and handling see the respective peripheral chapters). With the Pn_IOCR register ([Section 23.8.1](#)) it is also possible to activate an internal weak pull-up or pull-down device in the pad.

The input register Pn_IN and the ALTIN signal always represent the state of the input, independent whether the port pin is configured as input or output. So, even if the port is

General Purpose I/O Ports (PORTS)

in output mode, the level of the pin can be read by software via Pn_IN and/or a peripheral can use the pin level as an input.

The ALTIN input signal of a port pin can be evaluated by multiple on-chip peripherals at the same time. For example, a pin used as slave select input of a USIC channel configured as SPI slave can also be used as trigger input of the ERU to trigger a service request or a wake-up event when the connected SPI master starts a communication.

23.2.2 Output Operation

In output mode, the output driver is activated and drives the value supplied through the output multiplexer to the port pin. Switching between input and output mode is accomplished through the Pn_IOCR register ([Section 23.8.1](#)), which

- enables or disables the output driver,
- selects between open-drain and push-pull mode,
- selects the general purpose or alternate function outputs.

The output multiplexer selects the signal source of the output with

- Pn_IOCR
 - general purpose output (Pn_OUT, [Section 23.8.4](#))
 - alternate peripheral functions, ALT1..ALT4
- hardware control, Pn_HWSEL
 - HWO0
 - HWO1

Note: It is recommended to complete the Port and peripheral configuration with respect to driver strength, operating mode and initial values before the port pin is switched to output mode.

The output function is exclusive, meaning that always only exactly one peripheral has control of the output path.

Used as general purpose output, software can directly modify the content of Pn_OUT to define the output value on the pin. A write operation to Pn_OUT updates all port pins of that port (e.g. P0) that are configured as general purpose output. Updating just one or a selected few general purpose output pins via Pn_OUT requires a masked read-modify-write operation to avoid disturbing pins that shall not be changed. Direct writes to Pn_OUT will also affect Pn_OUT bits configured for use with the Pin Power-save function, [Section 23.4](#).

Because of that, it is preferred to modify Pn_OUT bits by the Output Modification Register Pn_OMR ([Section 23.8.5](#)). The bits in Pn_OMR allow to individually set, clear or toggle the bits in the Pn_OUT register and only update the “addressed” Pn_OUT bits.

The data written by software into the output register Pn_OUT can also be used as input data to an on-chip peripheral. This enables, for example, peripheral tests and simulation via software without external circuitry.

General Purpose I/O Ports (PORTS)

Output lines of on-chip peripherals can directly control the output value of the output driver if selected via ALT1 to ALT4 as well as HW0_OUT and HW1_OUT. After initialization, this allows the connected peripherals to directly drive complex control and communication patterns without further software interaction with the ports.

The actual logic level at the pin can be examined through reading Pn_IN and compared against the applied output level (either applied by the output register Pn_OUT, or via an alternate output function of a peripheral unit). This can be used to detect some electrical failures at the pin caused through external circuitry. In addition, software-supported arbitration schemes between different “masters” can be implemented in this way, using the open-drain configuration and an external wired-AND circuitry. Collisions on the external communication lines can be detected when a high level (1_B) is output, but a low level (0_B) is seen when reading the pin value via the input register Pn_IN or directly by a peripheral (via ALTIN, for example a USIC channel in IIC mode).

Driver Mode

Before activating the push-pull driver, it is recommended to configure its driver strength and slew rate according to its pad class and the application need by the Pad Driver Mode register Pn_PDR ([Section 23.8.2](#)). Selecting the appropriate driver strength allows to optimize the outputs for the needed interface performance, can help to reduce power consumption, and limits noise, crosstalk and electromagnetic emissions.

There are three classes of GPIO output drivers:

- Class A1 pads (low speed 3.3V LVTTTL outputs)
- Class A1+ pads (medium speed 3.3V LVTTTL outputs)
- Class A2 pads (high speed 3.3V LVTTTL outputs, e.g. for EBU or fast serial interfaces)

Class A1 pins provide the choice between medium and weak output drivers.

Class A1+ pins provide the choice between strong/medium/weak output drivers. For the strong driver, the signal transition edge can be additionally selected as soft or slow.

Class A2 pins provide the choice between strong/medium/weak output drivers. For the strong driver, the signal transition edge can be additionally selected as sharp/medium/soft.

The assignment of each port pin to one of these pad classes is listed in the [Package Pin Summary](#) table. Further details about pad properties in the XMC4300 are summarized in the Data Sheet.

23.3 Hardware Controlled I/Os

Some ports pins are overlaid with peripheral functions for which the connected peripheral needs direct hardware control, e.g. for the direction of a bi-directional data bus. There is a dedicated hardware control interface for these functions. As multiple peripherals need access to this interface, the Pn_HWSEL register ([Section 23.8.8](#))

General Purpose I/O Ports (PORTS)

allows to select between the hardware “masters”. The assigned functions are listed in the columns HWO0/HWIO in the **Port I/O Functions** table.

Depending on the operating mode, the peripheral can take control of various functions:

- Pin direction, input or output, e.g. for bi-directional signals
- Driver type, open-drain or push-pull
- Pull devices under peripheral control or under standard control via Pn_IOCR

Some configurations remain under control by the standard configuration interface, the output driver strength by Pn_PDR and the direct or inverted input path by Pn_IOCR.

Pn_HWSEL.HWx just pre-assigns the hardware-control of the pin to a certain peripheral, but the peripheral itself decides to actually take control over it. As long as the peripheral does not take control of a given pin via HWx_EN, the configuration of this pin is still defined by the configuration registers and it is available as GPIO or for other alternate functions. This might be because the selected peripheral has controls to just activate a subset of its pins, or because the peripheral is not active at all. E.g. unused address lines of the EBU are free for use as GPIO.

This mechanism can also be used to prohibit the hardware control of certain pins to a peripheral, in case the application does not need the respective functionality and the peripheral has no controls to disable the hardware control selectively.

If not specified differently, hardware outputs activate the push-pull output driver and the strength is defined by Pn_PDR. Similarly, the default hardware input configuration and the pull devices are controlled by Pn_IOCR.

If the JTAG interface is selected by Pn_HWSEL of the respective port pins, pull device and output driver configuration is overridden by hardware.

If configured accordingly, the LEDTS module can also control the internal pull devices and change between push-pull and open-drain output drivers.

Note: Do not enable the Pin Power Save function for pins configured for Hardware Control (Pn_HWSEL.HWx != 00_B). Doing so may result in an undefined behavior of the pin when the device enters the Deep Sleep state.

23.4 Power Saving Mode Operation

In Deep Sleep mode, the behavior of a pin depends on the setting of the Pin Power Save register Pn_PPS (**Section 23.8.7**). Basically, each pin can be configured to react to the Power Save Mode Request or to ignore it. In case a pin is configured to react to a Power Save Mode Request, the output driver is switched to tri-state, the input Schmitt-Trigger and the pull devices are switched off (see **Figure 23-2**). The input signal to the on-chip peripherals is optionally driven statically high or low, software-defined by a value stored in Pn_OUT or by the last input value sampled to the Pn_OUT register during normal operation. The actual reaction is configured with the Pn_IOCR register under power save conditions, see **Table 23-8**.

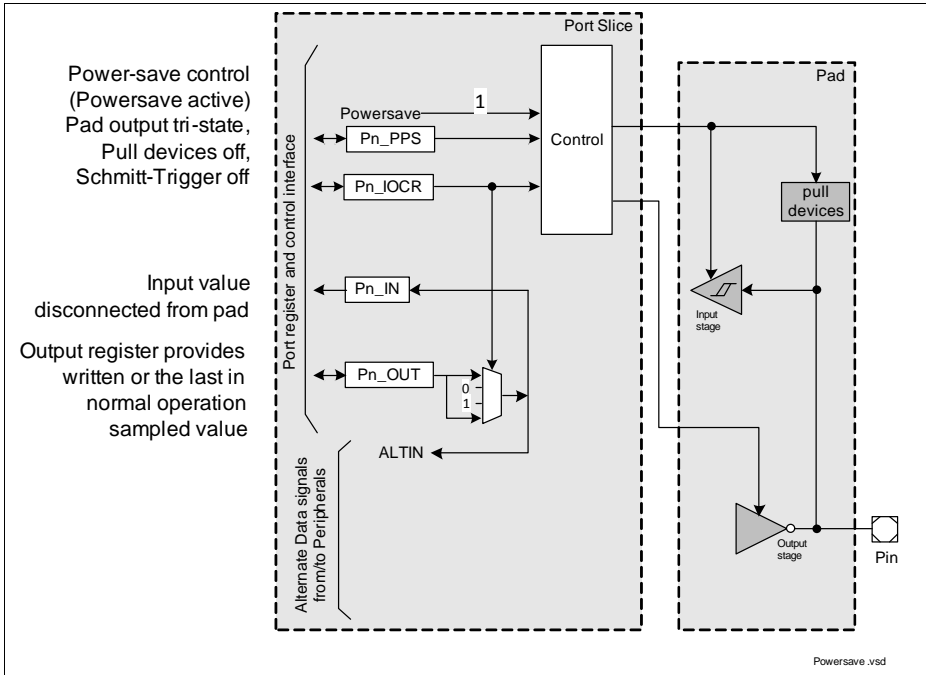


Figure 23-2 Port Pin in Power Save State

Note: Do not enable the Pin Power Save function for pins configured for Hardware Control (Pn_HWSEL.HWx != 00_B). Doing so may result in an undefined behavior of the pin when the device enters the Deep Sleep state.

23.5 Analog Ports

P14 and P15 are analog and digital input ports with a simplified port and pad structure, see [Figure 23-3](#). The analog pads have no output drivers and the digital input Schmitt-Trigger can be controlled by the Pn_PDISC ([Section 23.8.3](#)) register. Accordingly, the port control interface is reduced in its functionality. The Pn_IOCR register controls the pull devices, the optional input inversion and the input source in power-save mode. The Pn_OUT has only its power-save functionality, as described in [Section 23.4](#).

General Purpose I/O Ports (PORTS)

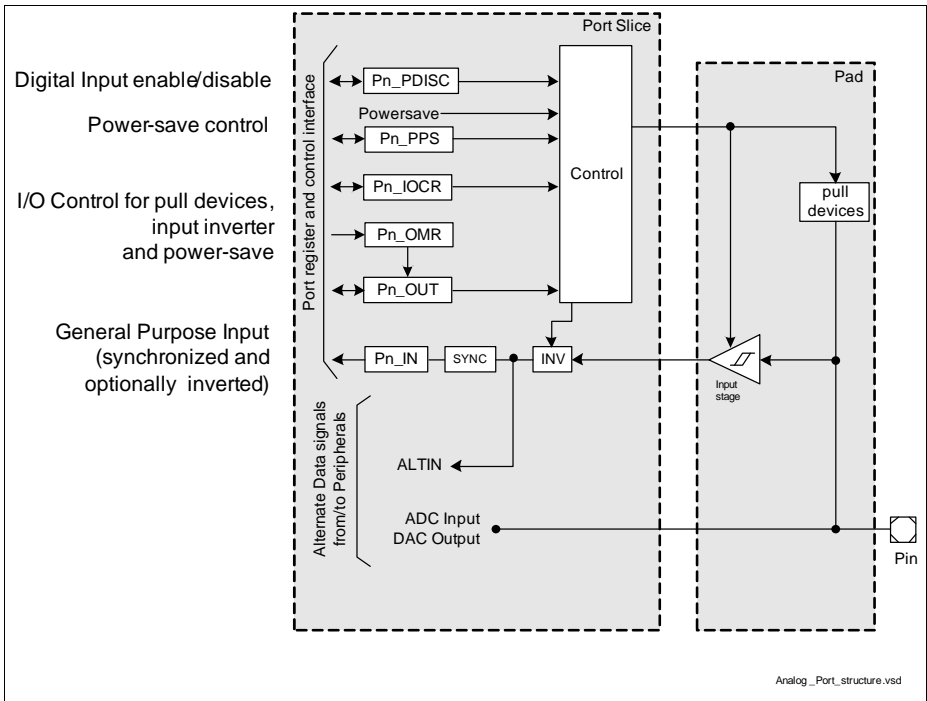


Figure 23-3 Analog Port Structure

23.6 Power, Reset and Clock

During power-up, until V_{DDC} and V_{DDP} voltage levels are stable and within defined limits, or during power-fail, while one/some voltage levels are outside the defined limits, the digital I/O pads are held in a defined state, which is tri-state input, output driver disabled and no pull devices active. The JTAG interface activates pull devices as default configuration.

The pins of the battery-buffered Hibernate Domain are independent to the supply monitoring of V_{DDC} and V_{DDP} , but are reset with the Standby Reset (see Reset Control chapter in the System Control Unit).

See the SCU Power Management chapter and the Data Sheet for details on the power-up, supply monitoring and voltage limits.

All Port registers are reset with the System Reset (see Reset Control chapter in the System Control Unit). The standard reset values are defined such that the port pins are configured as tri-state inputs, output driver disabled and no pull devices active.

General Purpose I/O Ports (PORTS)

Exceptions from these standard values are related to special interfaces (for example JTAG) or the analog input channels.

The Ports register interface is connected to Peripheral Bridge 1 (PBA1) and all registers of the Ports are clocked with f_{PERIPH} .

23.7 Initialization and System Dependencies

It is recommended to follow pre-defined routines for the initialization of the port pins.

Input

When a peripheral shall use a port pin as input, the actual pin levels may immediately trigger an unexpected peripheral event (e.g. clock edge at SPI). This can be avoided by forcing the "passive" level via pull-up/down programming.

The following steps are required to configure a port pin as an input:

- Pn_IOCR
input configuration with pull device and/or power-save mode configuration
- Hardware Control (if applicable)
 - Pn_HWSEL
switch hardware control to peripheral
- Pin Power Save (if applicable)
 - Pn_OMR/Pn_OUT
default value in power save mode (if applicable)
 - Pn_PPS
enable power save control

Output

When a port pin is configured as output for an on-chip peripheral, it is important that the peripheral is configured before the port switches the control to the peripheral in order to avoid spikes on the output.

The following steps are required to configure a port pin as an output:

- Pn_OMR/Pn_OUT
Initial output value (as general purpose output)
- Pn_PDR
Pad Driver Strength configuration
- GPIO or Alternate Output
 - Pn_IOCR
Output multiplexer select
Push-pull or open-drain output driver mode
Activates the output driver!
- Hardware Control

General Purpose I/O Ports (PORTS)

- Pn_IOCR
depending on the hardware function Pn_IOCR can enable the internal pull devices
- Pn_HWSEL
Switch hardware control to peripheral

Transitions

If a port pin is used for different functions that require a reconfiguration of the port registers, it is recommended to do this transition via an intermediate “neutral” tri-state input configuration.

- Pn_HWSEL
disable hardware selection; can be omitted if no hardware control is used on the port pin
- Pn_PPS
disable power save mode control of the pin; can be omitted if no power save configuration is used on the port pin
- Pn_IOCR
tri-state input and no pull device active

23.8 Registers

Registers Overview

The absolute register address is calculated by adding:
Module Base Address + Offset Address

Table 23-2 Registers Address Space

Module	Base Address	End Address	Note
P0	4802 8000 _H	4802 80FF _H	
P1	4802 8100 _H	4802 81FF _H	
P2	4802 8200 _H	4802 82FF _H	
P3	4802 8300 _H	4802 83FF _H	
P4	4802 8400 _H	4802 84FF _H	
P5	4802 8500 _H	4802 85FF _H	
P14	4802 8E00 _H	4802 8EFF _H	Analog/Digital Input only
P15	4802 8F00 _H	4802 8FFF _H	Analog/Digital Input only

General Purpose I/O Ports (PORTS)

Table 23-3 Register Overview

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
Pn_OUT	Port n Output Register	0000 _H	U, PV	U, PV	Page 23-24
Pn_OMR	Port n Output Modification Register	0004 _H	U, PV	U, PV	Page 23-25
–	Reserved	0008 _H - 000C _H	BE	BE	–
Pn_IOCR0	Port n Input/Output Control Register 0	0010 _H	U, PV	PV	Page 23-14
Pn_IOCR4	Port n Input/Output Control Register 4	0014 _H	U, PV	PV	Page 23-15
Pn_IOCR8	Port n Input/Output Control Register 8	0018 _H	U, PV	PV	Page 23-15
Pn_IOCR12	Port n Input/Output Control Register 12	001C _H	U, PV	PV	Page 23-16
–	Reserved	0020 _H	BE	BE	–
Pn_IN	Port n Input Register	0024 _H	U, PV	R	Page 23-26
–	Reserved	0028 _H - 003C _H	BE	BE	–
Pn_PDR0	Port n Pad Driver Mode 0 Register	0040 _H	U, PV	PV	Page 23-20
Pn_PDR1	Port n Pad Driver Mode 1 Register	0044 _H	U, PV	PV	Page 23-21
–	Reserved	0048 _H - 005C _H	BE	BE	–
Pn_PDISC	Port n Pin Function Decision Control Register (non-ADC ports)	0060 _H	U, PV	BE	Page 23-22
P14_PDISC P15_PDISC	Port n Pin Function Decision Control Register (ADC ports)	0060 _H	U, PV	PV	Page 23-23
–	Reserved	0064 _H - 006C _H	BE	BE	–
Pn_PPS	Port n Pin Power Save Register	0070 _H	U, PV	PV	Page 23-27

General Purpose I/O Ports (PORTS)

Table 23-3 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
Pn_HWSEL	Port n Hardware Select Register	0074 _H	U, PV	PV	Page 23-28
–	Reserved	0078 _H -00FC _H	BE	BE	–

Table 23-4 Registers Access Rights and Reset Classes

Register Short Name	Access Rights		Reset Class
	Read	Write	
Pn_IN	U, PV	R	System Reset
Pn_OUT		U, PV	
Pn_OMR		PV	
Pn_IOCR0			
Pn_IOCR4			
Pn_IOCR8			
Pn_IOCR12			
Pn_PDISC (ADC ports)			
Pn_PDR0			
Pn_PDR1			
Pn_PPS			
Pn_HWSEL			
Pn_PDISC (non-ADC ports)		BE	

23.8.1 Port Input/Output Control Registers

The port input/output control registers select the digital output and input driver functionality and characteristics of a GPIO port pin. Port direction (input or output), pull-up or pull-down devices for inputs, and push-pull or open-drain functionality for outputs can be selected by the corresponding bit fields PCx (x = 0-15). Each 32-bit wide port input/output control register controls four GPIO port lines:

- Register Pn_IOCR0 controls the Pn.[3:0] port lines
- Register Pn_IOCR4 controls the Pn.[7:4] port lines
- Register Pn_IOCR8 controls the Pn.[11:8] port lines
- Register Pn_IOCR12 controls the Pn.[15:12] port lines

The diagrams below show the register layouts of the port input/output control registers with the PCx bit fields. One PCx bit field controls exactly one port line Pn.x.

Pn_IOCR0 (n=0-5)

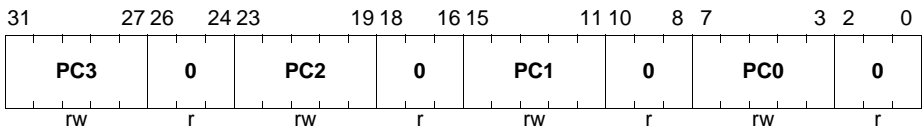
Port n Input/Output Control Register 0

(4802 8010_H + n*100_H) Reset Value: 0000 0000_H

Pn_IOCR0 (n=14-15)

Port n Input/Output Control Register 0

(4802 8010_H + n*100_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
PC0, PC1, PC2, PC3	[7:3], [15:11], [23:19], [31:27]	rw	Port Control for Port n Pin 0 to 3 This bit field determines the Port n line x functionality (x = 0-3) according to the coding table (see Table 23-5).
0	[2:0], [10:8], [18:16], [26:24]	r	Reserved Read as 0; should be written with 0.

General Purpose I/O Ports (PORTS)

Pn_IOCRA (n=0-3)

Port n Input/Output Control Register 4

(4802 8014_H + n*100_H)

Reset Value: 0000 0000_H

P5_IOCRA

Port 5 Input/Output Control Register 4

(0014_H)

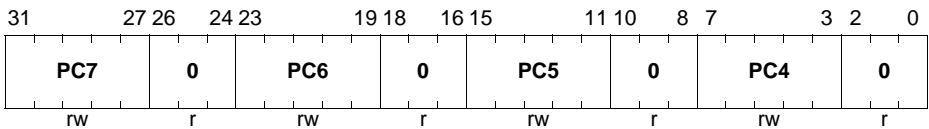
Reset Value: 0000 0000_H

Pn_IOCRA (n=14-15)

Port n Input/Output Control Register 4

(4802 8014_H + n*100_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:3], [15:11], [23:19], [31:27]	rw	Port Control for Port n Pin 4 to 7 This bit field determines the Port n line x functionality (x = 4-7) according to the coding table (see Table 23-5).
0	[2:0], [10:8], [18:16], [26:24]	r	Reserved Read as 0; should be written with 0.

Pn_IOCRA8 (n=0-2)

Port n Input/Output Control Register 8

(4802 8018_H + n*100_H)

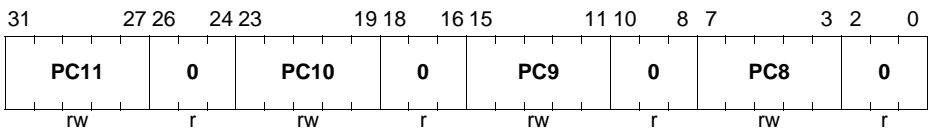
Reset Value: 0000 0000_H

Pn_IOCRA8 (n=14-15)

Port n Input/Output Control Register 8

(4802 8018_H + n*100_H)

Reset Value: 0000 0000_H



General Purpose I/O Ports (PORTS)

Field	Bits	Type	Description
PC8, PC9, PC10, PC11	[7:3], [15:11], [23:19], [31:27]	rw	Port Control for Port n Pin 8 to 11 This bit field determines the Port n line x functionality (x = 8-11) according to the coding table (see Table 23-5).
0	[2:0], [10:8], [18:16], [26:24]	r	Reserved Read as 0; should be written with 0.

Pn_IOCR12 (n=0-2)

Port n Input/Output Control Register 12

(4802 801C_H + n*100_H)

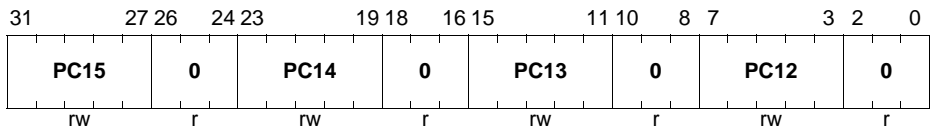
Reset Value: 0000 0000_H

P14_IOCR12

Port 14 Input/Output Control Register 12

(001C_H)

Reset Value: 0000 0000_H



Field	Bits	Type	Description
PC12, PC13, PC14, PC15	[7:3], [15:11], [23:19], [31:27]	rw	Port Control for Port n Pin 12 to 15 This bit field determines the Port n line x functionality (x = 12-15) according to the coding table (see Table 23-5).
0	[2:0], [10:8], [18:16], [26:24]	r	Reserved Read as 0; should be written with 0.

Depending on the GPIO port functionality (number of GPIO lines of a port), not all of the port input/output control registers are implemented.

The structure with one control bit field for each port pin located in different register bytes offers the possibility to configure the port pin functionality of a single pin with byte-oriented accesses without accessing the other PCx bit fields.

General Purpose I/O Ports (PORTS)

Port Control Coding

Table 23-5 describes the coding of the PCx bit fields that determine the port line functionality.

The Pn_IOCry PCx bit field is also used to control the pin behavior in Deep Sleep mode if the Pin Power Save option is enabled, see **Section 23.8.7**.

Table 23-5 Standard PCx Coding¹⁾

PCx[4:0]	I/O	Output Characteristics	Selected Pull-up / Pull-down / Selected Output Function
0X000 _B	Direct Input	–	No internal pull device active
0X001 _B			Internal pull-down device active
0X010 _B			Internal pull-up device active
0X011 _B			No internal pull device active; Pn_OUTx continuously samples the input value
0X100 _B	Inverted Input	–	No internal pull device active
0X101 _B			Internal pull-down device active
0X110 _B			Internal pull-up device active
0X111 _B			No internal pull device active; Pn_OUTx continuously samples the input value

General Purpose I/O Ports (PORTS)

Table 23-5 Standard PCx Coding¹⁾ (cont'd)

PCx[4:0]	I/O	Output Characteristics	Selected Pull-up / Pull-down / Selected Output Function
10000 _B	Output (Direct Input)	Push-pull	General-purpose output
10001 _B			Alternate output function 1
10010 _B			Alternate output function 2
10011 _B			Alternate output function 3
10100 _B			Alternate output function 4
10101 _B			Reserved.
10110 _B			Reserved.
10111 _B			Reserved.
11000 _B		Open-drain	General-purpose output
11001 _B			Alternate output function 1
11010 _B			Alternate output function 2
11011 _B			Alternate output function 3
11100 _B			Alternate output function 4
11101 _B			Reserved.
11110 _B			Reserved.
11111 _B			Reserved.

1) For the analog and digital input ports P14 and P15 the combinations with PCx[4]=1_B are reserved.

23.8.2 Pad Driver Mode Register

The pad structure of the XMC4300 GPIO lines offers the possibility to select the output driver strength and the slew rate. These two parameters are controlled by the bit fields in the pad driver mode registers Pn_PDR0/1, independently from input/output and pull-up/pull-down control functionality as programmed in the Pn_IOCR register. Pn_PDR0 and Pn_PDR1 registers are assigned to each port.

Depending on the assigned pad class, the 3-bit wide pad driver mode selection bit fields PDx in the pad driver mode registers Pn_PDR make it possible to select the port line functionality as shown in [Table 23-6](#). Note that the pad driver mode registers are specific for each port.

General Purpose I/O Ports (PORTS)

Table 23-6 Pad Driver Mode Selection

Pad Class	PDx.2	PDx.1	PDx.0	Functionality
A1	X	X	0	Medium driver
			1	Weak driver
A1+	0	X	0	Strong driver soft edge
	0	X	1	Strong driver slow edge
	1	X	0	Medium driver
	1	X	1	Weak driver
A2	0	0	0	Strong driver, sharp edge
	0	0	1	Strong driver, medium edge
	0	1	0	Strong driver, soft edge
	0	1	1	Reserved
	1	0	0	Medium driver
	1	0	1	
	1	1	0	Reserved
	1	1	1	Weak driver

Note: The XMC4300 Data Sheet describes the DC characteristics of all pad classes.

Pad Driver Mode Registers

This is the general description of the PDR registers. Each port contains its own specific PDR registers, described additionally at each port, that can contain between one and eight PDx fields for PDR0 and PDR1 registers, respectively. Each field controls 1 pin. For coding of PDx, see [Table 23-6](#).

The analog and digital input ports P14 and P15 don't have Pn_PDR registers.

General Purpose I/O Ports (PORTS)

Pn_PDR0 (n=0-5)

Port n Pad Driver Mode 0 Register(4802 8040_H + n*100_H) Reset Value: 2222 2222_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	PD7			0	PD6			0	PD5			0	PD4		
r	rw			r	rw			r	rw			r	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PD3			0	PD2			0	PD1			0	PD0		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
PD0	[2:0]	rw	Pad Driver Mode for Pn.0
PD1	[6:4]	rw	Pad Driver Mode for Pn.1
PD2	[10:8]	rw	Pad Driver Mode for Pn.2
PD3	[14:12]	rw	Pad Driver Mode for Pn.3
PD4	[18:16]	rw	Pad Driver Mode for Pn.4
PD5	[22:20]	rw	Pad Driver Mode for Pn.5
PD6	[26:24]	rw	Pad Driver Mode for Pn.6
PD7	[30:28]	rw	Pad Driver Mode for Pn.7
0	3, 7, 11, 15, 19, 23, 27, 31	r	Reserved Read as 0; should be written with 0.

General Purpose I/O Ports (PORTS)

Pn_PDR1 (n=0-2)

Port n Pad Driver Mode 1 Register

(4802 8044_H + n*100_H)

Reset Value: 2222 2222_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	PD15			0	PD14			0	PD13			0	PD12		
r	rw			r	rw			r	rw			r	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PD11			0	PD10			0	PD9			0	PD8		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
PD8	[2:0]	rw	Pad Driver Mode for Pn.8
PD9	[6:4]	rw	Pad Driver Mode for Pn.9
PD10	[10:8]	rw	Pad Driver Mode for Pn.10
PD11	[14:12]	rw	Pad Driver Mode for Pn.11
PD12	[18:16]	rw	Pad Driver Mode for Pn.12
PD13	[22:20]	rw	Pad Driver Mode for Pn.13
PD14	[26:24]	rw	Pad Driver Mode for Pn.14
PD15	[30:28]	rw	Pad Driver Mode for Pn.15
0	3, 7, 11, 15, 19, 23, 27, 31	r	Reserved Read as 0; should be written with 0.

23.8.3 Pin Function Decision Control Register

Pin Function Decision Control Register

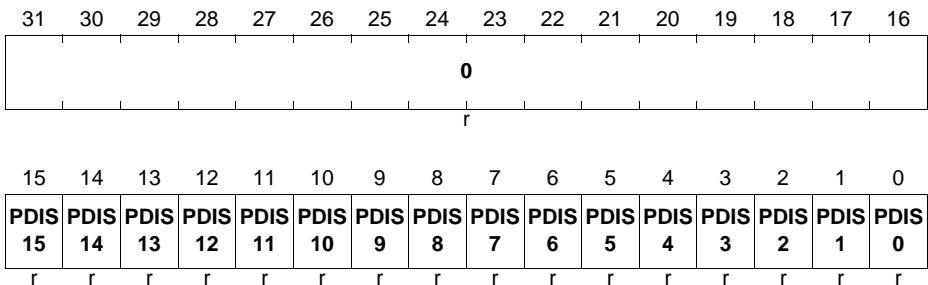
The primary use for this register is to disable/enable the digital pad structure in shared analog and digital ports, see the dedicated description of the **Pn_PDISC (n=14-15)** register of the analog ports.

For “normal” digital I/O ports (P0-P6) this register is read-only and the read value corresponds to the available pins in the given package.

Pn_PDISC (n=0-5)

Port n Pin Function Decision Control Register

(4802 8060_H + n*100_H) Reset Value: 0000 XXXX_H¹⁾



1) The reset value is package dependent.

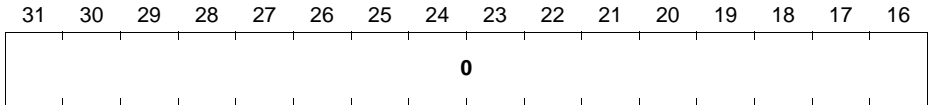
Field	Bits	Type	Description
PDISx (x = 0-15)	x	r	Pad Disable for Port n Pin x 0 _B Pad Pn.x is enabled. 1 _B Pad Pn.x is disabled.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

General Purpose I/O Ports (PORTS)

Pn_PDISC (n=14-15)

Port n Pin Function Decision Control Register

(4802 8060_H + n*100_H) Reset Value: 0000 XXXX_H¹⁾



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

1) The reset value is package dependent.

Field	Bits	Type	Description
PDISx (x = 0-15)	x	rW	Pad Disable for Port n Pin x This bit disables or enables the digital pad function. 0 _B Digital Pad input is enabled. Analog and digital input path active. 1 _B Digital Pad input is disabled. Analog input path active. (default)
0	[31:16]	r	Reserved Read as 0; should be written with 0.

23.8.4 Port Output Register

The port output register determines the value of a GPIO pin when it is selected by Pn_IOC Rx as output. Writing a 0 to a Pn_OUT.Px (x = 0-15) bit position delivers a low level at the corresponding output pin. A high level is output when the corresponding bit is written with a 1. Note that the bits of Pn_OUT.Px can be individually set/reset by writing appropriate values into the port output modification register Pn_OMR, avoiding read-modify-write operations on the Pn_OUT, which might affect other pins of the port.

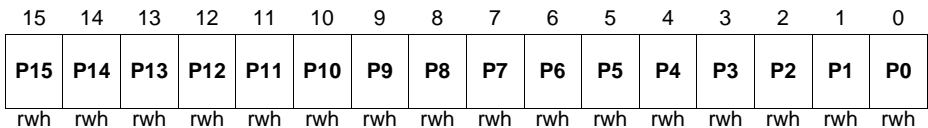
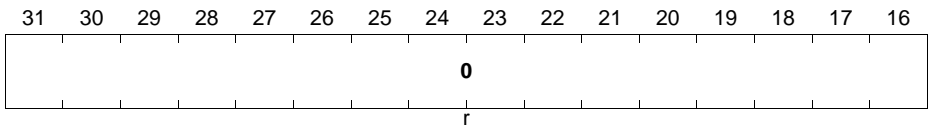
The Pn_OUT is also used to store/drive a defined value for the input in Deep Sleep mode. For details on this see the [Port Pin Power Save Register](#). That is also the only use of the Pn_OUT register in the analog and digital input ports P14 and P15.

Pn_OUT (n=0-5)

Port n Output Register **(4802 8000_H + n*100_H)** **Reset Value: 0000 0000_H**

Pn_OUT (n=14-15)

Port n Output Register **(4802 8000_H + n*100_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
Px (x = 0-15)	x	rwh	Port n Output Bit x This bit determines the level at the output pin Pn.x if the output is selected as GPIO output. 0 _B The output level of Pn.x is 0. 1 _B The output level of Pn.x is 1. Pn.x can also be set/reset by control bits of the Pn_OMR register.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

General Purpose I/O Ports (PORTS)

23.8.5 Port Output Modification Register

The port output modification register contains control bits that make it possible to individually set, reset, or toggle the logic state of a single port line by manipulating the output register.

Pn_OMR (n=0-5)

Port n Output Modification Register

$$(4802\ 8004_H + n \cdot 100_H)$$

Reset Value: 0000 0000_H

Pn_OMR (n=14-15)

Port n Output Modification Register

$$(4802\ 8004_H + n \cdot 100_H)$$

Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PR	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR	PR
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS	PS
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
PSx (x = 0-15)	x	w	Port n Set Bit x Setting this bit will set or toggle the corresponding bit in the port output register Pn_OUT. The function of this bit is shown in Table 23-7 .
PRx (x = 0-15)	x + 16	w	Port n Reset Bit x Setting this bit will reset or toggle the corresponding bit in the port output register Pn_OUT. The function of this bit is shown in Table 23-7 .

Note: Register Pn_OMR is virtual and does not contain any flip-flop. A read action delivers the value of 0. A 8 or 16-bits write behaves like a 32-bit write padded with zeros.

Table 23-7 Function of the Bits PRx and PSx

PRx	PSx	Function
0	0	Bit Pn_OUT.Px is not changed.
0	1	Bit Pn_OUT.Px is set.
1	0	Bit Pn_OUT.Px is reset.
1	1	Bit Pn_OUT.Px is toggled.

23.8.6 Port Input Register

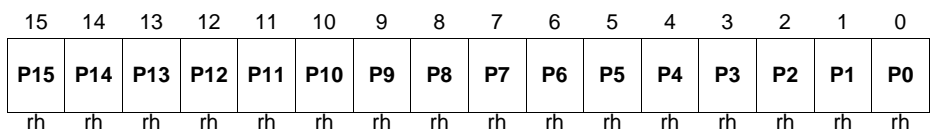
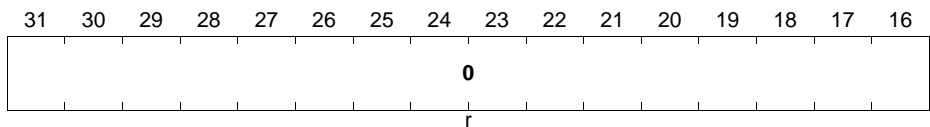
The logic level of a GPIO pin can be read via the read-only port input register Pn_IN. Reading the Pn_IN register always returns the current logical value at the GPIO pin, synchronized to avoid meta-stabilities, independently whether the pin is selected as input or output.

Pn_IN (n=0-5)

Port n Input Register (4802 8024_H + n*100_H) **Reset Value: 0000 XXXX_H**

Pn_IN (n=14-15)

Port n Input Register (4802 8024_H + n*100_H) **Reset Value: 0000 XXXX_H**



Field	Bits	Type	Description
Px (x = 0-15)	x	rh	Port n Input Bit x This bit indicates the level at the input pin Pn.x. 0 _B The input level of Pn.x is 0. 1 _B The input level of Pn.x is 1.
0	[31:16]	r	Reserved Read as 0.

23.8.7 Port Pin Power Save Register

When the XMC4300 enters Deep Sleep mode, pins with enabled Pin Power Save option are set to a defined state and the input Schmitt-Trigger as well as the output driver stage are switched off.

Note: Do not enable the Pin Power Save function for pins configured for Hardware Control ($Pn_HWSEL.HWx \neq 00_B$). Doing so may result in an undefined behavior of the pin when the device enters the Deep Sleep state.

Pn_PPS (n=0-5)

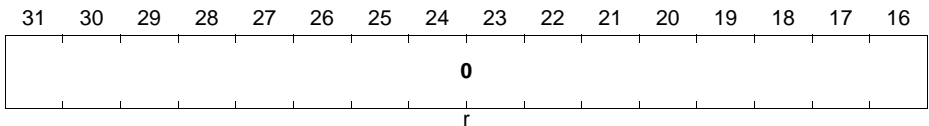
Port n Pin Power Save Register

$(4802\ 8070_H + n*100_H)$ Reset Value: 0000 0000_H

Pn_PPS (n=14-15)

Port n Pin Power Save Register

$(4802\ 8070_H + n*100_H)$ Reset Value: 0000 0000_H



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

Field	Bits	Type	Description
PPSx (x = 0-15)	x	rW	Port n Pin Power Save Bit x 0 _B Pin Power Save of Pn.x is disabled. 1 _B Pin Power Save of Pn.x is enabled.
0	[31:16]	r	Reserved Read as 0.

Deep Sleep Pin Power Save behavior

The actual behavior in Deep Sleep mode with enabled Pin Power Save is controlled by the Pn_IOCRy.PCx bit field ([Page 23-14](#)) of the respective pin. [Table 23-8](#) shows the coding.

Table 23-8 PCx Coding in Deep Sleep mode

PCx[4:0]	I/O	Normal Operation or PPSx=0 _B	Deep Sleep mode and PPSx=1 _B
0X000 _B	Direct Input	See Table 23-5	Input value=Pn_OUTx
0X001 _B			Input value=0 _B ; pull-down deactivated
0X010 _B			Input value=1 _B ; pull-up deactivated
0X011 _B			Input value=Pn_OUTx, storing the last sampled input value
0X100 _B	Inverted Input	See Table 23-5	Input value= $\overline{\text{Pn_OUTx}}$
0X101 _B			Input value=1 _B ; pull-down deactivated
0X110 _B			Input value=0 _B ; pull-up deactivated
0X111 _B			Input value= $\overline{\text{Pn_OUTx}}$, storing the last sampled input value
1XXXX _B	Output	See Table 23-5	Output driver off, Input Schmitt-Trigger off, no pull device active, Input value=Pn_OUTx

23.8.8 Port Pin Hardware Select Register

Some peripherals require direct hardware control of their I/Os. As on some pins multiple such peripheral I/Os are mapped, the register Pn_HWSEL is used to select which peripheral has the control over the pin.

Note: Pn_HWSEL.HWx just pre-assigns the hardware-control of the pin to a certain peripheral, but the peripheral itself decides to actually take control over it. As long as the peripheral does not take control of a given pin via HWx_EN, the configuration of this pin is still defined by the configuration registers and it is available as GPIO or for other alternate functions. This might be because the selected peripheral has controls to just activate a subset of its pins, or because the peripheral is not active at all.

This mechanism can also be used to prohibit the hardware control of certain pins to a peripheral, in case the application does not need the respective functionality and the peripheral has no controls to disable the hardware control selectively.

The shared analog and digital input ports P14 and P15 do not support the hardware select feature.

General Purpose I/O Ports (PORTS)

P0_HWSEL

Port 0 Pin Hardware Select Register (0074_H) Reset Value: 0001 4000_H

P1_HWSEL

Port 1 Pin Hardware Select Register (0074_H) Reset Value: 0000 0000_H

P2_HWSEL

Port 2 Pin Hardware Select Register (0074_H) Reset Value: 0000 0004_H

Pn_HWSEL (n=3-5)

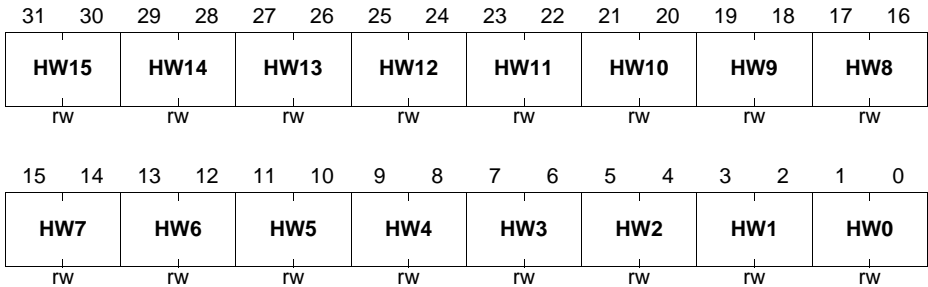
Port n Pin Hardware Select Register
(4802 8074_H + n*100_H) Reset Value: 0000 0000_H

P14_HWSEL

Port 14 Pin Hardware Select Register (0074_H) Reset Value: 0000 0000_H

P15_HWSEL

Port 15 Pin Hardware Select Register (0074_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
HWx (x = 0-15)	[2*x+1: 2*x]	rw	<p>Port n Pin Hardware Select Bit x</p> <p>00_B Software control only.</p> <p>01_B HWI0/HWO0 control path can override the software configuration.</p> <p>10_B HWI1/HWO1 control path can override the software configuration.</p> <p>11_B Reserved.</p>

23.9 Package Pin Summary

The following general building block is used to describe each pin:

Table 23-9 Package Pin Mapping Description

Function	Package A	Package B	...	Pad Type	Notes
Name	N	Ax	...	A2	

The table is sorted by the “Function” column, starting with the regular Port pins (Px.y), followed by the dedicated pins (i.e. PORST) and supply pins.

The following columns, titled with the supported package variants, lists the package pin number to which the respective function is mapped in that package.

The “Pad Type” indicates the employed pad type (A1, A1+, A2, special=special pad, In=input pad, AN/DIG_IN=analog and digital input, Power=power supply). Details about the pad properties are defined in the Data Sheet.

In the “Notes”, special information to the respective pin/function is given, i.e. deviations from the default configuration after reset. Per default the regular Port pins are configured as direct input with no internal pull device active.

Table 23-10 Package Pin Mapping

Function	LQFP-100	Pad Type	Notes
P0.0	2	A1+	
P0.1	1	A1+	
P0.2	100	A2	
P0.3	99	A2	
P0.4	98	A2	
P0.5	97	A2	
P0.6	96	A2	
P0.7	89	A2	After a system reset, via HWSEL this pin selects the DB.TDI function.
P0.8	88	A2	After a system reset, via HWSEL this pin selects the DB.TRST function, with a weak pull-down active.
P0.9	4	A2	
P0.10	3	A1+	
P0.11	95	A1+	

General Purpose I/O Ports (PORTS)

Table 23-10 Package Pin Mapping (cont'd)

Function	LQFP-100	Pad Type	Notes
P0.12	94	A1+	
P1.0	79	A1+	
P1.1	78	A1+	
P1.2	77	A2	
P1.3	76	A2	
P1.4	75	A1+	
P1.5	74	A1+	
P1.6	83	A2	
P1.7	82	A2	
P1.8	81	A2	
P1.9	80	A2	
P1.10	73	A1+	
P1.11	72	A1+	
P1.12	71	A2	
P1.13	70	A2	
P1.14	69	A2	
P1.15	68	A2	
P2.0	52	A2	
P2.1	51	A2	After a system reset, via HWSEL this pin selects the DB.TDO function.
P2.2	50	A2	
P2.3	49	A2	
P2.4	48	A2	
P2.5	47	A2	
P2.6	54	A1+	
P2.7	53	A1+	
P2.8	46	A2	
P2.9	45	A2	
P2.10	44	A2	
P2.14	41	A2	
P2.15	40	A2	
P3.0	7	A2	

General Purpose I/O Ports (PORTS)

Table 23-10 Package Pin Mapping (cont'd)

Function	LQFP-100	Pad Type	Notes
P3.1	6	A2	
P3.2	5	A2	
P3.3	93	A1+	
P3.4	92	A1+	
P3.5	91	A2	
P3.6	90	A2	
P4.0	85	A2	
P4.1	84	A2	
P5.0	58	A1+	
P5.1	57	A1+	
P5.2	56	A1+	
P5.7	55	A1+	
P14.0	31	AN/DIG_IN	
P14.1	30	AN/DIG_IN	
P14.2	29	AN/DIG_IN	
P14.3	28	AN/DIG_IN	
P14.4	27	AN/DIG_IN	
P14.5	26	AN/DIG_IN	
P14.6	25	AN/DIG_IN	
P14.7	24	AN/DIG_IN	
P14.8	37	AN/DAC/DIG_IN	
P14.9	36	AN/DAC/DIG_IN	
P14.12	23	AN/DIG_IN	
P14.13	22	AN/DIG_IN	
P14.14	21	AN/DIG_IN	
P14.15	20	AN/DIG_IN	
P15.2	19	AN/DIG_IN	
P15.3	18	AN/DIG_IN	
P15.8	39	AN/DIG_IN	
P15.9	38	AN/DIG_IN	

General Purpose I/O Ports (PORTS)

Table 23-10 Package Pin Mapping (cont'd)

Function	LQFP-100	Pad Type	Notes
HIB_IO_0	14	A1 special	At the first power-up and with every reset of the hibernate domain this pin is configured as open-drain output and drives "0". As output the medium driver mode is active.
HIB_IO_1	13	A1 special	At the first power-up and with every reset of the hibernate domain this pin is configured as input with no pull device active. As output the medium driver mode is active.
USB_DP	9	special	
USB_DM	8	special	
TCK	67	A1	Weak pull-down active.
TMS	66	A1+	Weak pull-up active. As output the strong-soft driver mode is active.
$\overline{\text{PORST}}$	65	special	Weak pull-up permanently active, strong pull-down controlled by EVR.
XTAL1	61	clock_IN	
XTAL2	62	clock_O	
RTC_XTAL1	16	clock_IN	
RTC_XTAL2	15	clock_O	
VBAT	17	Power	When VDDP is supplied VBAT has to be supplied as well.
VBUS	10	special	
VAREF	33	AN_Ref	
VAGND	32	AN_Ref	
VDDA	35	AN_Power	
VSSA	34	AN_Power	
VDDC	12	Power	
VDDC	42	Power	
VDDC	64	Power	
VDDC	86	Power	
VDDP	11	Power	

General Purpose I/O Ports (PORTS)

Table 23-10 Package Pin Mapping (cont'd)

Function	LQFP-100	Pad Type	Notes
VDDP	43	Power	
VDDP	60	Power	
VDDP	87	Power	
VSS	59	Power	
VSSO	63	Power	
VSS	Exp. Pad	Power	<p>Exposed Die Pad The exposed die pad is connected internally to VSS. For proper operation, it is mandatory to connect the exposed pad directly to the common ground on the board. For thermal aspects, please refer to the Data Sheet. Board layout examples are given in an application note.</p>

23.10 Port I/O Functions

The following general building block is used to describe each PORT pin:

Table 23-11 Port I/O Function Description

Function	Outputs			Inputs		
	ALT1	AL _{Tn}	HWO0	HWI0	Input	Input
P0.0		MODA.OUT	MODB.OUT	MODB.INA	MODC.INA	
P _n .y	MODA.OUT				MODA.INA	MODC.INB

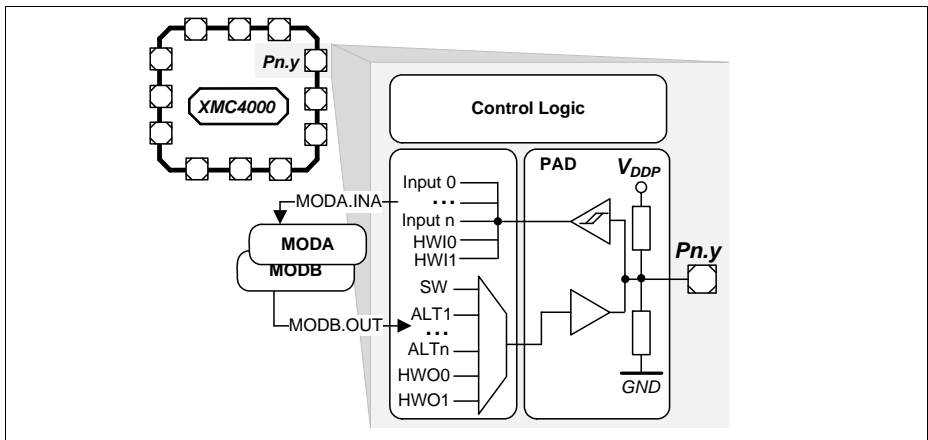


Figure 23-4 Simplified Port Structure

$Pn.y$ is the port pin name, defining the control and data bits/registers associated with it. As GPIO, the port is under software control. Its input value is read via $Pn_IN.y$, Pn_OUT defines the output value.

Up to four alternate output functions (ALT1/2/3/4) can be mapped to a single port pin, selected by $Pn_IOCR.PC$. The output value is directly driven by the respective module, with the pin characteristics controlled by the port registers (within the limits of the connected pad).

The port pin input can be connected to multiple peripherals. Most peripherals have an input multiplexer to select between different possible input sources.

The input path is also active while the pin is configured as output. This allows to feedback an output to on-chip resources without wasting an additional external pin.

By Pn_HWSEL (Section 23.8.8) it is possible to select between different hardware “masters” (HWO0/HWI0). The selected peripheral can take control of the pin(s). Hardware control overrules settings in the respective port pin registers.



23.10.1 Port I/O Function Table

Table 23-12 Port I/O Functions

Function	Output					Input								
	ALT1	ALT2	ALT3	ALT4	HW00	HWI0	Input	Input	Input	Input	Input	Input	Input	Input
P0.0	ECAT0. PHY_RST	CAN. NO_TXD	CCU80. OUT21	LEDT50. COL2			U1C1. DX0D	ETH0. CLK_RMIB	ERU0. 0B0					ETH0. CLKRXB
P0.1	USB. DRIVEBUS	U1C1. DOUT0	CCU80. OUT11	LEDT50. COL3				ETH0. CRS_DVB	ERU0. 0A0				ECAT0. P1_RX_CLKA	ETH0. RXDVB
P0.2	ECAT0. P1_TXD2	U1C1. SELO1	CCU80. OUT01		U1C0. DOUT3	U1C0. HWIN3	ETH0. RXDOB		ERU0. 3B3					
P0.3	ECAT0. P1_TXD3		CCU80. OUT20		U1C0. DOUT2	U1C0. HWIN2	ETH0. RXD1B			ERU1. 3B0				
P0.4	ETH0. TX_EN		CCU80. OUT10		U1C0. DOUT1	U1C0. HWIN1		U1C0. DX0A	ERU0. 2B3				ECAT0. P1_RXD3A	
P0.5	ETH0. TXD0	U1C0. DOUT0	CCU80. OUT00		U1C0. DOUT0	U1C0. HWIN0		U1C0. DX0B		ERU1. 3A0			ECAT0. P1_RXD2A	
P0.6	ETH0. TXD1	U1C0. SELO0	CCU80. OUT30					U1C0. DX2A	ERU0. 3B2			CCU80. IN2B	ECAT0. P1_RXD1A	
P0.7	WWDT. SERVICE_OUT	U0C0. SELO0	ECAT0. LED_ERR			DB. TDI	U0C0. DX2B		ERU0. 2B1			CCU80. IN0A	CCU80. IN1A	CCU80. IN2A
P0.8	SCU. EXTCLK	U0C0. SCLKOUT	ECAT0. LED_RUN			DE. TRST	U0C0. DX1B		ERU0. 2A1			CCU80. IN1B		
P0.9		U1C1. SELO0	CCU80. OUT12	LEDT50. COL0	ETH0. MDO	ETH0. MDIA	U1C1. DX2A	USB. ID	ERU0. 1B0				ECAT0. P1_RX_DVA	
P0.10	ETH0. MDC	U1C1. SCLKOUT	CCU80. OUT02	LEDT50. COL1			U1C1. DX1A		ERU0. 1A0				ECAT0. P1_TX_CLKA	
P0.11	ECAT0. P1_LINK_ACT	U1C0. SCLKOUT	CCU80. OUT31		SDMMC. RST		ETH0. RXERB	U1C0. DX1A	ERU0. 3A2				ECAT0. P1_RXD0A	
P0.12		U1C1. SELO0	CCU40. OUT3		ECAT0. MDO	ECAT0. MDIA		U1C1. DX2B	ERU0. 2B2					
P1.0		U0C0. SELO0	CCU40. OUT3	ERU1. PDOUT3			U0C0. DX2A		ERU0. 3B0			CCU40. IN3A		ECAT0. P0_TX_CLKA
P1.1		U0C0. SCLKOUT	CCU40. OUT2	ERU1. PDOUT2		SDMMC. SDWC	U0C0. DX1A		ERU0. 3A0			CCU40. IN2A		ECAT0. P0_RX_CLKA
P1.2	ECAT0. P0_TXD3		CCU40. OUT1	ERU1. PDOUT1	U0C0. DOUT3	U0C0. HWIN3				ERU1. 2B0		CCU40. IN1A		
P1.3	ECAT0. P0_TX_ENA	U0C0. MCLKOUT	CCU40. OUT0	ERU1. PDOUT0	U0C0. DOUT2	U0C0. HWIN2				ERU1. 2A0		CCU40. IN0A		
P1.4	WWDT. SERVICE_OUT	CAN. NO_TXD	CCU80. OUT33		U0C0. DOUT1	U0C0. HWIN1	U0C0. DX0B	CAN. N1_RXDD	ERU0. 2B0			CCU41. IN0C		ECAT0. P0_RXD0A
P1.5	CAN. N1_TXD	U0C0. DOUT0	CCU80. OUT23		U0C0. DOUT0	U0C0. HWIN0	U0C0. DX0A	CAN. N0_RXDA	ERU0. 2A0	ERU1. 0A0		CCU41. IN1C		ECAT0. P0_RXD1A
P1.6	ECAT0. P0_TXD0	U0C0. SCLKOUT			SDMMC. DATA1_OUT	SDMMC. DATA1_IN								

Table 23-12 Port I/O Functions (cont'd)

Function	Output					Input									
	ALT1	ALT2	ALT3	ALT4	HWO0	HWI0	Input	Input	Input	Input	Input	Input	Input	Input	Input
P1.7	ECAT0. P0_TXD1	U0C0. DOUT0		U1C1. SELO2	SDMMC. DATA2_OUT	SDMMC. DATA2_IN									
P1.8	ECAT0. P0_TXD2	U0C0. SELO1		U1C1. SCLKOUT	SDMMC. DATA4_OUT	SDMMC. DATA4_IN									
P1.9	U0C0. SCLKOUT			U1C1. DOUT0	SDMMC. DATA5_OUT	SDMMC. DATA5_IN									ECAT0. P0_RX_DVA
P1.10	ETH0. MDC	U0C0. SCLKOUT		ECAT0. LED_ERR		SDMMC. SDCD						CCU41. IN2C			ECAT0. P0_RXD2A
P1.11	ECAT0. LED_STATE_R UN	U0C0. SELO0		ECAT0. LED_RUN	ETH0. MDO	ETH0. MDIC						CCU41. IN3C			ECAT0. P0_RXD3A
P1.12	ETH0. TX_EN	CAN. N1_TXD		ECAT0. P0_LINK_ACT	SDMMC. DATA6_OUT	SDMMC. DATA6_IN									
P1.13	ETH0. TXD0	U0C1. SELO3		ECAT0. PHY_CLK25	SDMMC. DATA7_OUT	SDMMC. DATA7_IN	CAN. N1_RXDC								
P1.14	ETH0. TXD1	U0C1. SELO2		ECAT0. SYNC0			U1C0. DX0E								
P1.15	SCU. EXTCLK			U1C0. DOUT0								ERU1. 1A0			ECAT0. P0_LINKB
P2.0	CAN. N0_TXD			LEDS0. COL1	ETH0. MDO	ETH0. MDIB				ERU0. 0B3			CCU40. IN1C		
P2.1				LEDS0. COL0	DB.TDO/ TRACESWO		ETH0. CLK_RMIIA					ERU1. 0B0	CCU40. IN0C		ETH0. CLKRXA
P2.2	VADC. EMUX00		CCU41. OUT3	LEDS0. LINE0	LEDS0. EXTENDED0	LEDS0. TSIN0A	ETH0. RXD0A	U0C1. DX0A	ERU0. 1B2				CCU41. IN3A		
P2.3	VADC. EMUX01	U0C1. SELO0	CCU41. OUT2	LEDS0. LINE1	LEDS0. EXTENDED1	LEDS0. TSIN1A	ETH0. RXD1A	U0C1. DX2A	ERU0. 1A2				CCU41. IN2A		
P2.4	VADC. EMUX02	U0C1. SCLKOUT	CCU41. OUT1	LEDS0. LINE2	LEDS0. EXTENDED2	LEDS0. TSIN2A	ETH0. RXERA	U0C1. DX1A	ERU0. 0B2				CCU41. IN1A		
P2.5	ETH0. TX_EN	U0C1. DOUT0	CCU41. OUT0	LEDS0. LINE3	LEDS0. EXTENDED3	LEDS0. TSIN3A	ETH0. RXDVA	U0C1. DX0B	ERU0. 0A2				CCU41. IN0A		ETH0. CRS_DVA
P2.6		ERU1. PDDUT3	CCU80. OUT13	LEDS0. COL3				CAN. N1_RXDA	ERU0. 1B3				CCU40. IN3C	ECAT0. P0_RX_ERRB	
P2.7	ETH0. MDC	CAN. N1_TXD	CCU80. OUT03	LEDS0. COL2								ERU1. 1B0	CCU40. IN2C		
P2.8	ETH0. TXD0	ERU1. PDDUT1	CCU80. OUT32	LEDS0. LINE4	LEDS0. EXTENDED4	LEDS0. TSIN4A	DAC. TRIGGER5						CCU40. IN0B	CCU40. IN2B	CCU40. IN3B
P2.9	ETH0. TXD1	ERU1. PDDUT2	CCU80. OUT22	LEDS0. LINE5	LEDS0. EXTENDED5	LEDS0. TSIN5A	DAC. TRIGGER4						CCU41. IN0B	CCU41. IN1B	CCU41. IN2B
P2.10	VADC. EMUX10	ERU1. PDDUT0	ECAT0. PHY_RST	ECAT0. SYNC1											
P2.14	VADC. EMUX11	U1C0. DOUT0	CCU80. OUT21							U1C0. DX0D					
P2.15	VADC. EMUX12	ECAT0. P1_TXD3	CCU80. OUT11	LEDS0. LINE6	LEDS0. EXTENDED6	LEDS0. TSIN6A	ETH0. COLA	U1C0. DX0C							

Table 23-12 Port I/O Functions (cont'd)

Function	Output					Input								
	ALT1	ALT2	ALT3	ALT4	HWO0	HWI0	Input	Input	Input	Input	Input	Input	Input	Input
P3.0		U0C1. SCLKOUT		ECAT0. P1_TX_ENA			U0C1. DX1B				CCU80. IN2C			
P3.1		U0C1. SELO0	ECAT0. P1_TXD0				U0C1. DX2B		ERU0. 0B1		CCU80. IN1C			
P3.2	USB. DRIVEVBUS	CAN. NO_TXD	ECAT0. P1_TXD1	LEDS0. COLA					ERU0. 0A1		CCU80. IN0C			
P3.3		U1C1. SELO1		ECAT0. MCLK	SDMMC. LED						CCU80. IN3B			
P3.4		U1C1. SELO2			SDMMC. BUS_POWER						CCU80. IN5B	ECAT0. P1_LINKA		
P3.5		U1C1. SELO3		U0C1. DOUT0	SDMMC. CMD_OUT	SDMMC. CMD_IN			ERU0. 3B1			ECAT0. P1_RX_ERRA		
P3.6		U1C1. SELO4		U0C1. SCLKOUT	SDMMC. CLK_OUT	SDMMC. CLK_IN			ERU0. 3A1					
P4.0		ECAT0. PHY_CLK25		U1C0. SCLKOUT	SDMMC. DATA0_OUT	SDMMC. DATA0_IN	U1C1. DX1C		U0C1. DX0E				ECAT0. P0_RX_ERRA	
P4.1		U1C1. MCLKOUT		U0C1. SELO0	SDMMC. DATA3_OUT	SDMMC. DATA3_IN							ECAT0. P0_LINKA	
P5.0				ERU1. P0OUT0				ETH0. RXD0D	U0C0. DX0D	ECAT0. P0_RXD0B				
P5.1	U0C0. DOUT0			ERU1. P0OUT1				ETH0. RXD1D		ECAT0. P0_RXD1B				
P5.2		ECAT0. P0_LINK_ACT		ERU1. P0OUT2				ETH0. GRS_DVD		ECAT0. P0_RXD2B			ETH0. RXDVD	
P5.7	ECAT0. SYNCG			LEDS0. COLA						ECAT0. P0_RXD3B				
P14.0							VADC. G0CH0							
P14.1							VADC. G0CH1							
P14.2							VADC. G0CH2	VADC. G1CH2						
P14.3							VADC. G0CH3	VADC. G1CH3		CAN. NO_RXDB				
P14.4							VADC. G0CH4						ECAT0. LATCH1A	
P14.5							VADC. G0CH5						ECAT0. LATCH0A	
P14.6							VADC. G0CH6					GOORC6	ECAT0. P1_RX_CLKB	
P14.7							VADC. G0CH7					GOORC7	ECAT0. P1_RXD0B	
P14.8					DAC. OUT_0			VADC. G1CH0			ETH0. RXD0C			
P14.9					DAC. OUT_1			VADC. G1CH1			ETH0. RXD1C			

Table 23-12 Port I/O Functions (cont'd)

Function	Output					Input								
	ALT1	ALT2	ALT3	ALT4	HWO0	HWI0	Input	Input	Input	Input	Input	Input	Input	Input
P14.12								VADC. G1CH4						ECAT0. P1_RXD1B
P14.13								VADC. G1CH5						ECAT0. P1_RXD2B
P14.14								VADC. G1CH6				G1ORC6		ECAT0. P1_RXD3B
P14.15								VADC. G1CH7				G1ORC7		ECAT0. P1_RX_DVB
P15.2														ECAT0. P1_RX_ERRB
P15.3														ECAT0. P1_LINKB
P15.8											ETH0. CLK_RMII0			ETH0. CLKRXC
P15.9											ETH0. CRS_DVC			ETH0. RXDVC
HIB_IO_0	HIBOUT	WWDT. SERVICE_OUT						WAKEUPA						
HIB_IO_1	HIBOUT	WWDT. SERVICE_OUT						WAKEUPB						
USB_DP														
USB_DM														
TCK							DB.TCK/ SWCLK							
TMS						DB.TMS/ SWDIO								
PORST														
XTAL1							U0C0. DX0F	U0C1. DX0F	U1C0. DX0F	U1C1. DX0F				
XTAL2														
RTC_XTAL1									ERU0. 1B1					
RTC_XTAL2														

Startup Modes

24 Startup modes

This chapter describes the various startup modes supported by the device along with actions that must be performed by the end user.

24.1 Overview

The on-chip firmware resides in a non volatile memory namely the BootROM and is the first software of any kind to be executed by the CPU right after reset.

The on-chip firmware has two parts:

- Startup Software in BootROM (SSW) which provisions the various boot modes selectable by the user and is the main thread of execution.
- Test Firmware (Testware, not described in this document) which deals with test related routines that can be invoked during chip test in test mode only is not intended to be executed by the user.

The terms startup mode and boot mode mean the same and are used interchangeably throughout this chapter.

24.1.1 Features

Supported boot modes are summarized briefly. Desired boot mode can be enabled by driving the boot mode pins (JTAG TCK and TMS) with appropriate logic levels and issuing a power on reset (PORST). The actual value of these pins is latched into register STCON.HWCON. Some boot modes can only be selected by configuring STCON.SWCON bit field (of SCU module) and applying a system reset (such as a watchdog reset or CPU software reset).

Normal Boot mode

Startup type: *Internal start*

Required Reset type: *PORST or System reset*

An application located at the start of flash is given control after SSW has finished its execution. In case of an erased flash, the SSW will remain in BootROM.

Alternative Boot mode (ABM-0/ABM-1)

Startup type: *Internal start*

Required Reset type: *System reset*

An application located at user defined location on the flash is given control by SSW. The SSW after completing its execution evaluates a header hereafter known as ABM header kept at a well known address on the flash which in turn provides the location of application placed at user defined address. Two such applications can be programmed into the flash and thus two ABMs are supported. An invalid header results in the SSW

Startup modes

aborting further execution and launching the CPU into safe mode which is Diagnostics Monitor Mode (DMM). A PORST is required to exit the safe mode of operation.

Fallback ABM

Startup type: *Internal start*

Required Reset type: *System reset*

SSW evaluates the two ABM headers in succession. The first ABM header found valid by SSW results in application referenced by that header given control to. Should both the headers be found unusable, SSW aborts further execution and places the CPU into a safe mode which is Diagnostics Monitor Mode (DMM). A PORST is required to exit the safe mode of operation.

PSRAM boot

Startup type: *Internal start*

Required Reset type: *System reset*

An application loaded into PSRAM is given control after SSW finishes its execution. The start address of this application is deduced from an ABM like header placed in the last 32 bytes of PSRAM. An invalid header results in the SSW aborting further execution and launching the CPU into safe mode which is Diagnostics Monitor Mode (DMM). A PORST is required to exit the safe mode of operation.

ASC BSL (UART Bootstrap loading)

Startup type: *External start*

Required Reset type: *PORST or System reset*

An application can be downloaded into the start of PSRAM over the USIC ASC interface and executed. The size of the application downloaded is limited to the size of PSRAM on the device.

CAN BSL (CAN Bootstrap loading)

Startup type: *External start*

Required Reset type: *PORST or System reset*

An application can be downloaded into the start of PSRAM over the MultiCAN interface and executed. The size of the application downloaded is limited to the size of PSRAM on the device.

Boot mode Index (BMI)

Startup type: *Internal or External start*

Required Reset type: *PORST or System reset*

Startup modes

A user defined bootmode is offered via following mechanism. With initial factory programming (at customer site), a so called BMI string can be programmed into user configuration block (UCB). The BMI string describes actions that must be performed by SSW before lending into one of the internal or external startup modes.

The firmware can be triggered to initialize the parity of the SRAMs when starting in BMI boot mode, as depicted in [Figure 24-14](#). For other boot modes, the initialization needs to be done in the user start-up code.

24.2 Boot and Startup

This section describes the startup sequence of the XMC4300 as a process taking place before user application software takes control of the system.

24.2.1 Initial boot sequence

There are several tasks which the SSW performs before it gets to the point where the user requested boot mode must be identified and launched. This is to ensure that user applications have a stable execution environment when program control is hand over to them.

The SSW ensures that the flash subsystem has initialized before any user program can be executed out of flash memory.

It identifies the user requested boot mode and launches the same. It is important to state at this point that a few DSRAM1 locations are used by SSW for staging information read out of Flash Configuration Sector. [Figure 24-1](#) depicts usage of DSRAM1 used by SSW. In addition at every reset Device Information Data is reloaded at start of the DSRAM1. Detailed information about the content can be found in [Chapter 24.2.4](#).

Debugger access is prohibited if the global flash read protection is enabled. Before program control is hand over to user application (described next), SSW turns on the Startup protection feature. This restricts access to certain registers (described as startup protected registers) and memories such as the Flash Configuration Sector (FCS).

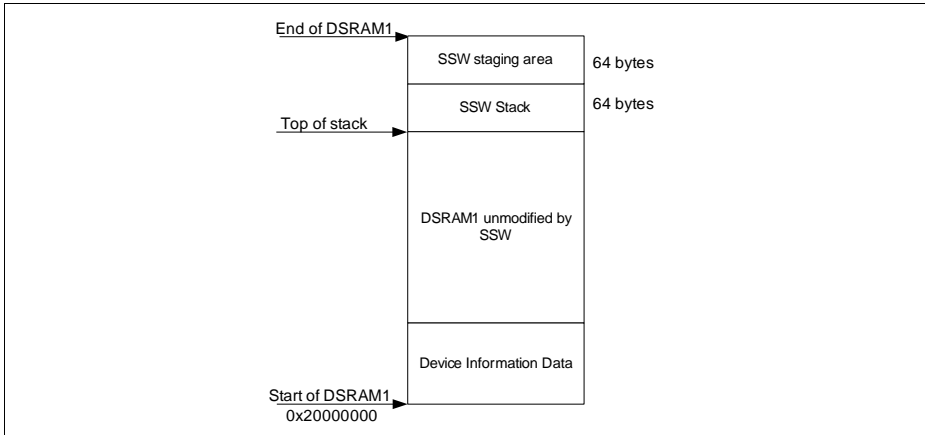


Figure 24-1 DSRAM1 usage by SSW

24.2.2 Reset types and corresponding boot modes

The XMC4000 family supports 2 categories of reset namely Power On Reset (PORST) and System Reset. Every cause of reset which is not a PORST is a System Reset.

24.2.2.1 Power On Reset (PORST)

When the SSW executes after a PORST as the reset type, it gets to choose from one of Normal boot mode, ASC BSL, CAN BSL and BMI based on what is read off the boot pins (JTAG TCK and TMS).

Pins TCK and TMS are stored into the HWCON[1:0] bit field of the SCU.STCON register after PORST and are mirrored into STCON.SWCON[1:0]. STCON.HWCON content can only be modified by PORST as reset source. HWCON bits are read by the SSW upon emergence from PORST.

Following table enlists the boot mode pin encoding and associated boot modes.

Table 24-1 Boot mode pin encoding for PORST

TCK	TMS	HWCON[1:0]	Boot mode
0	1	00 _B	Normal
0	0	01 _B	ASC BSL
1	1	10 _B	BMI
1	0	11 _B	CAN BSL

24.2.2.2 System Reset

When the SSW executes after a System Reset, it chooses one of the following boot modes - Normal, ASC BSL, BMI, CAN BSL, PSRAM boot, ABM-0, ABM-1 and Fallback ABM. Since the Reset Status Information in register SCU.RSTSTAT is the accumulated reset type, it is necessary to clean the bitfield using the SCU register RSTCLR.RSCLR before issuing a System Reset, else SSW will enter the boot mode reflected in STCON.HWCON.

The following table enlists the encoding of the SWCON bitfield and boot modes. In the event when software does not explicitly program SWCON and a system reset is experienced, values on TCK and TMS decide the boot mode.

Table 24-2 System reset boot modes

SWCON[3:0]	Boot mode
0000 _B	Normal
0001 _B	ASC BSL
0010 _B	BMI
0011 _B	CAN BSL
0100 _B	PSRAM boot
1000 _B	ABM-0
1100 _B	ABM-1
1110 _B	Fallback ABM

24.2.3 Boot mode selection

HWCON bit field is read only for PORST (Power ON Reset). For every other reset type (available in SCU_RSTSTAT) register, the SWCON field is assessed.

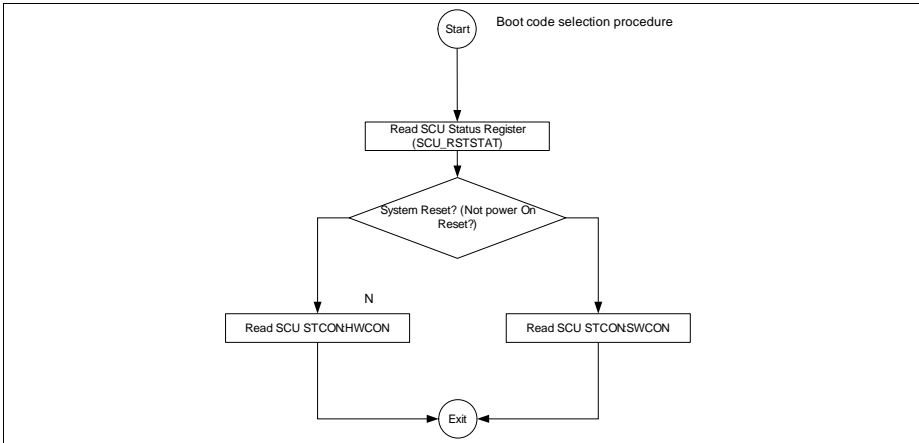


Figure 24-2 Reading Bootcode

Figure 24-3 depicts the decision tree that the SSW has to traverse in order to select the desired boot mode.

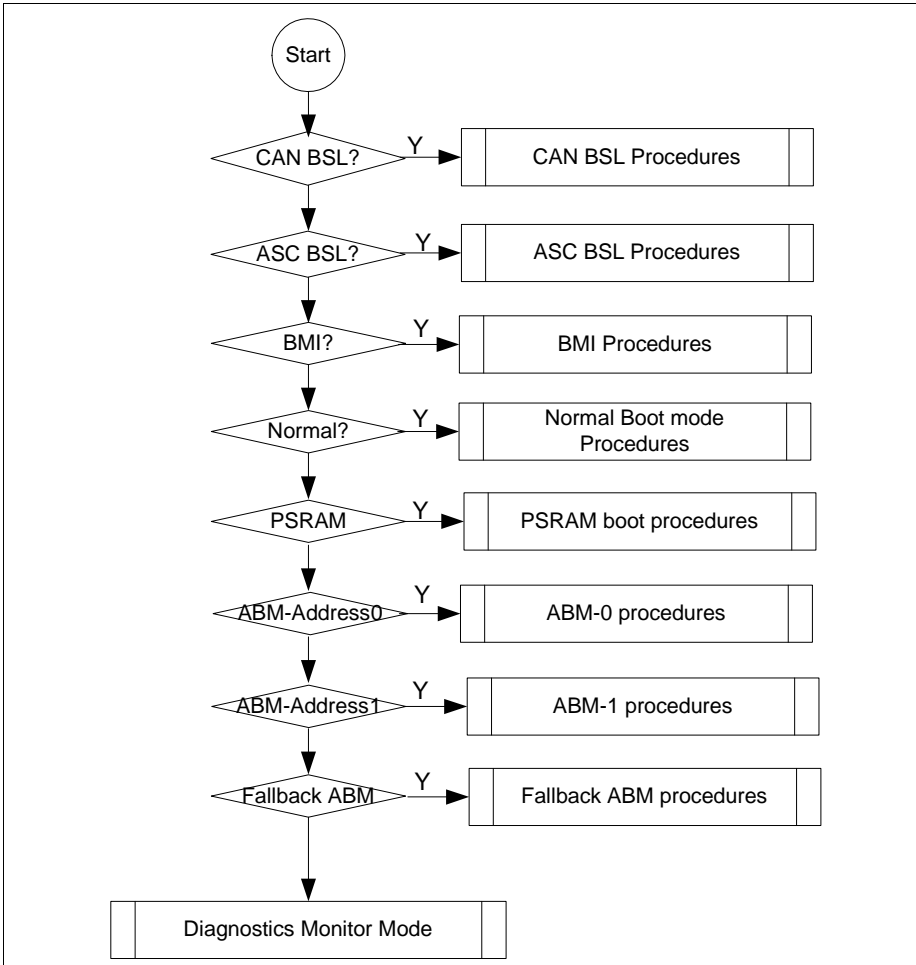


Figure 24-3 Boot mode identification

24.2.4 Unique Chip ID and Device Information Data

The XMC4300 provides device specific information stored at begin of DSRAM1. The first value is a Unique Chip ID. This ID is a 16 Byte value absolute unique for every XMC4000. Following is the DTS calibration data 128 Byte and the BMI string 64 Byte. The last values is the JTAG-ID 4 Byte. More information about the BMI can be found in [Chapter 24.3.8](#).

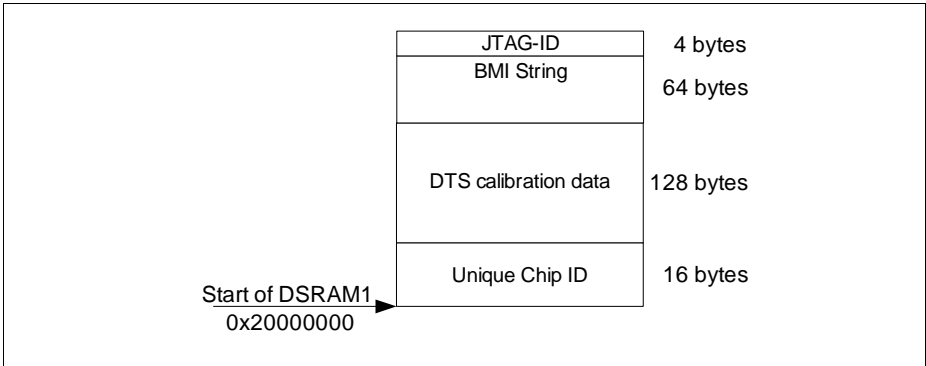


Figure 24-4 Device Information Data

24.3 Boot mode details

This chapter describes the Boot modes more in detail, explaining the Bootstrap Loaders (BSL) for UART and CAN, provide informations about setting up and handling the BMI.

24.3.1 Normal boot mode

This is a boot mode in which user application available at the start of flash (0C000000_H) is given control to after SSW execution.

SSW enables access to coresight system before passing control to the first user instruction. If the HALT after RESET feature were requested for by a connected hardware debugger, SSW configures a breakpoint on the first user instruction.

It is expected that the vector table of user application is available at the start of the flash. Firmware essentially reprograms the Cortex M4's SCB.VTOR register with the start address of flash (0C000000_H) and passes control to user application by programing register R15 (Program Counter) with reset vector contents. The reset vector contents point to a routine that could be in either the cached or the uncached address space of the flash.

User actions for normal boot mode

User must:

- Flash the application at the start of flash
- Drive TCK and TMS as per [Table 24-1](#)
- Issue a PORST

Alternatively

- A currently running program can setup SWCON bit field in the SCU.STCON register
- Clear Reset Source in SCU.RSTSTAT using the register RSTCLR.RSCLR
- Issue a system reset

24.3.2 Boot from PSRAM

This boot mode option requires user code to be downloaded into Program SRAM (PSRAM) first.

The SWCON bit field of the SCU.STCON register is then expected to be programmed with the Boot from PSRAM boot code. Second the reset status must be cleared followed by initiation of any of the system resets.

PORST leaves SRAM contents undefined. With a System Reset previous PSRAM contents retained intact. Application initiated software reset or a watchdog reset are two examples of System Reset.

For SSW to branch to user application in PSRAM, it must first be assured of integrity of user application. This is done by means of a magic key (A5C3E10F_H) and CRC audit. It

Startup modes

is therefore required that the PSRAM boot header be placed at the last 32 bytes of the PSRAM. The layout of the header is depicted in **Figure 24-5**. Polynomial used for checksum calculation is of CRC-32 type (04C11DB7_μ) with a initial value of 0x0H.

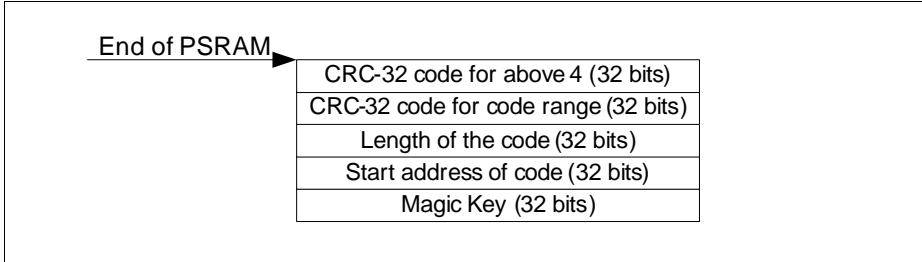


Figure 24-5 PSRAM header layout

This layout is reused in the ABM boot modes. A pictorial representation PSRAM usage for this boot mode is presented in **Figure 24-6**.

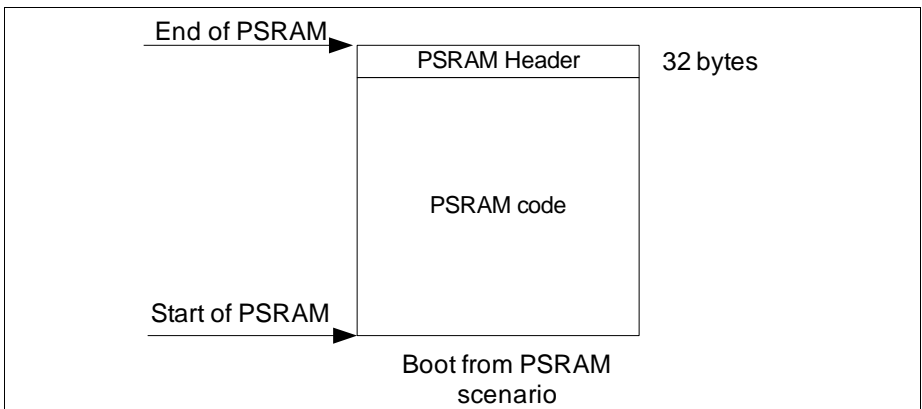


Figure 24-6 PSRAM layout for PSRAM boot

After audits confirm integrity of the code, SSW installs startup protection and cedes control to user application. SCB VTOR is programmed with PSRAM start address and CPU register R15 with application reset vector.

User actions for PSRAM boot mode

User must:

- Download application (Vector table + code) into PSRAM (Currently running program launched by any of the other internal boot modes can do this)

Startup modes

- Create PSRAM header and program the last 32 bytes of PSRAM with this header (Currently running program can either download header from host or create a header after application has been downloaded)
- Program SWCON bit field in the SCU.STCON register
- Clear Reset Source in SCU.RSTSTAT using the register RSTCLR.RSCLR
- Issue a system reset

24.3.3 Alternative boot mode - Address0 (ABM-0)

SSW can cede control to user application residing at an user defined flash address. As described in [Figure 24-5](#), an identical header is expected to be present at a fixed location in the flash. This fixed address for the header is the last 32 bytes (0C00FFE0_H on XMC4300) of the first 64 KB physical sector.

Should the SSW find a corrupted header, execution is aborted and a diagnostics monitor mode (DMM) is entered.

As a norm, the address range of an application is typically linear without any holes. As an exception, an application may be scattered in the flash (with the help of scatter linker scripts) thus leaving holes on the flash. In such as cases, SSW only audits the ABM header and decides if control may be ceded to the reset vector. Distinction between linear and scattered application is made by evaluating application CRC and application length fields of the header. Both of these fields are set to FFFFFFFF_H in the case of scattered application. Polynomial used for CRC calculations is CRC-32 (04C11DB7_H) with a initial value of 0_H. A pictorial representation of this concept is presented in [Figure 24-7](#).

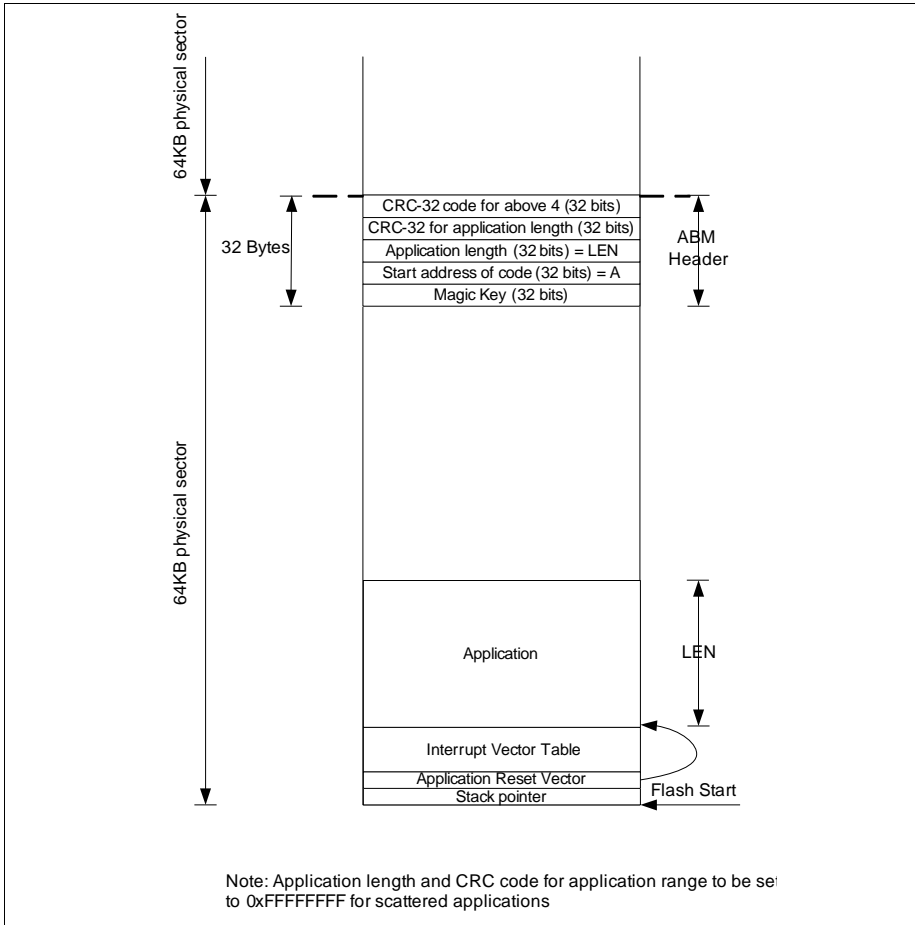


Figure 24-7 ABM concept

User actions

User must:

- Program flash with application and ABM header
- Drive TCK and TMS to launch Normal boot mode
- Configure STCON.SWCON per [Table 24-2](#)
- Clear Reset Source in SCU.RSTSTAT using the register RSTCLR.RSCLR
- Issue a system reset (Example: Watchdog reset, Software reset)

Alternatively

- program the flash with application and ABM header
- Encode the SWCON field in the BMI word with ABM boot code and program into the flash User Configuration Block-2 (UCB2-Page1)
- Drive TCK and TMS for BMI and issue PORST

24.3.4 Alternative boot mode - Address1 (ABM-1)

This is same as ABM-Address0. Address for the header is the last 32 bytes (0C01FFE0_H for XMC4300) of the second 64 KB physical sector.

24.3.5 Fallback ABM

When this boot mode is selected, ABM Address-0 header is audited first. A positive audit results in SSW passing control to user application pointed to by the header. A negative audit results in evaluation of ABM Address-1. Should the audit of ABM-Address1 header fail, SSW launched diagnostics monitor mode (DMM).

24.3.6 ASC BSL mode

SSW supports bootstrap loader modes. When configured, any user application limited to the size of PSRAM on the device can be downloaded into PSRAM over the USIC0 channel 0 (UOC0) and immediately executed.

As an example, this application may be a secondary flash loader that can download a larger application and write the latter into program flash.

Data and code fetches are disabled if the global flash read protection is installed. Initial preparation and generic procedures are described next.

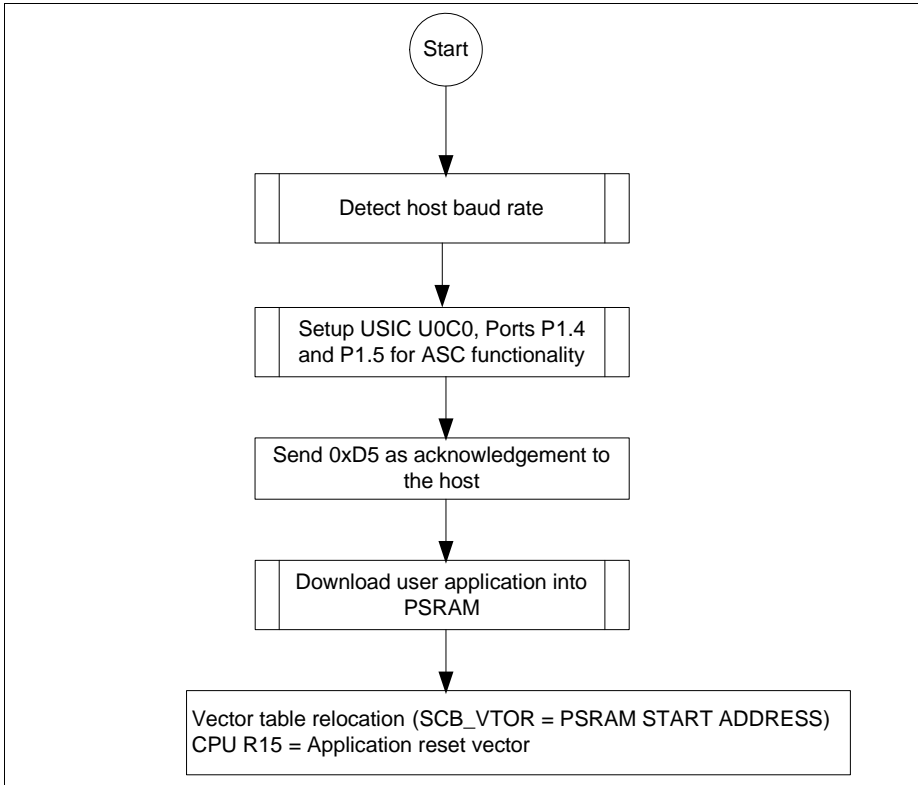


Figure 24-8 ASC BSL mode procedures

Full duplex ASC functionality of U0C0 is used for the BSL mode.

Port pins used are P1.4 (U0C0_DX0B) for USIC RX and P1.5 (U0C0_DOUT0) for USIC TX functionality.

The host starts by transmitting a zero byte to help the device detect the baud rate. After the baud rate has been detected by the device, a download protocol specified next helps download of application of any size only limited by the size of PSRAM on the device.

After the baud rate has been detected, SSW transmits an acknowledgement byte D5_H back to the host. It then awaits 4 bytes describing the length of the application from the host. The least significant byte is received first.

If application length is found acceptable by SSW, an OK (1_H) byte is sent to the host. Then the host will start to send the byte stream of the application. After the byte stream has been received, SSW terminates the protocol by sending a final OK (1_H) byte and then cedes control to the downloaded application.

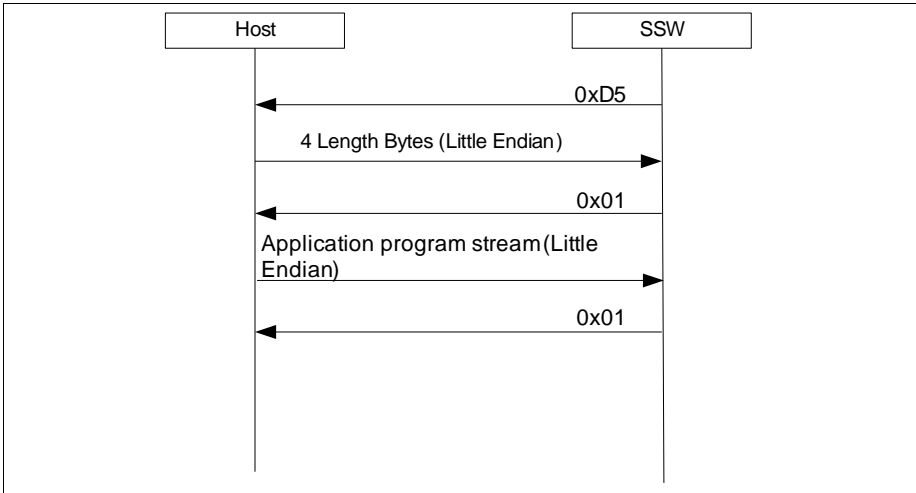


Figure 24-9 Application download protocol

If application length is found to be in error (application length greater than device PSRAM size), a `N_OK (2H)` byte is transmitted back to the host and the SSW resumes awaiting the length bytes.

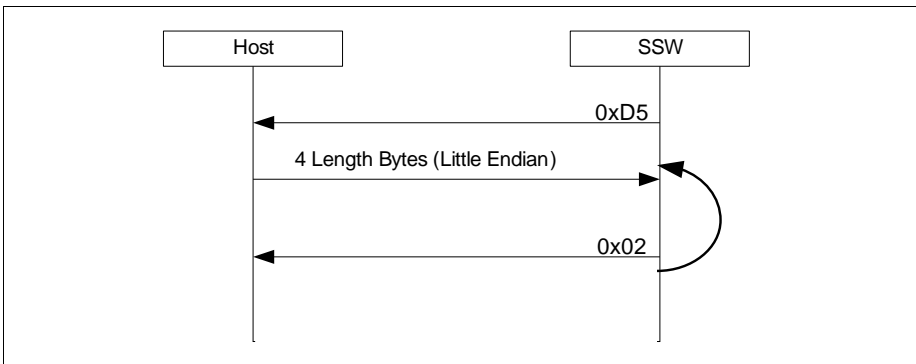


Figure 24-10 Application download protocol FAIL

User actions for ASC BSL mode

User must:

- Configure boot mode pins as described in [Table 24-1](#)
- Reset the device
- Ensure host sends 00_H to the device (used to capture the baud rate)
- Ensure host receives D5_H as acknowledgement from device
- Ensure host sends data length of application and receive a positive acknowledgement
- Ensure host sends the complete application byte stream

Alternatively

- A currently running program can setup SWCON bit field in the SCU.STCON register
- Clear Reset Source in SCU.RSTSTAT using the register RSTCLR.RSCLR
- Issue a system reset
- Ensure host sends 00_H to the device (used to capture the baud rate)
- Ensure host receives D5_H as acknowledgement from device
- Ensure host then sends data length of application and receive a positive acknowledgement
- Ensure host sends the complete application byte stream

24.3.7 CAN BSL mode

The CAN bootstrap loader mode transfers user application via Node-0 or Node-1 of the CAN module into PSRAM.

A stable external clock is mandatory. SSW uses an iterative algorithm to determine the external clock frequency and switches to it. For transfer rates of 1 maps it must be ensured by the end user that the external clock is at least 10 MHz.

A protocol comprising three phases described next leads to user application downloaded into PSRAM. Size of downloaded application is only limited by the size of PSRAM on the device.

[Figure 24-11](#) depicts CAN BSL mode procedures.

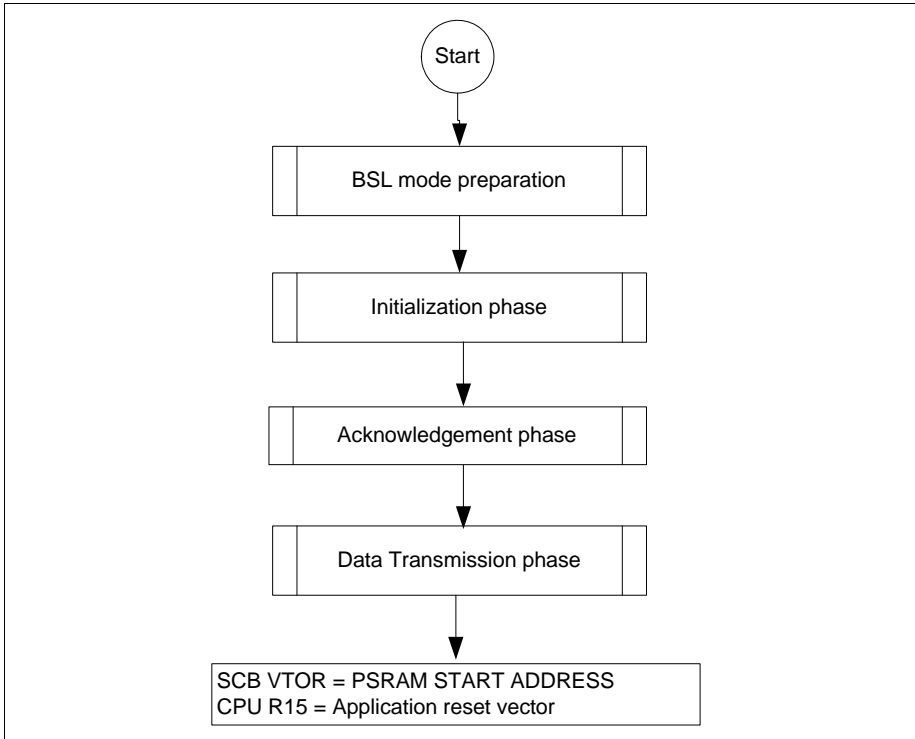


Figure 24-11 CAN BSL procedures

Initialization phase

The baud rate of the host is determined.

SSW switches to external clock source. PLL is brought out of power down mode and configured for pre-scalar mode of operation. The VCO is bypassed and powered down.

A standard CAN base frame comprising eight data bytes is transmitted continuously by the host. The LSB-aligned 11 bit message ID of the frame (555_H) is used by SSW for baud rate detection. Data bytes 0 and 1 contain 'don't care' bytes. Data bytes 2 and 3 contain the acknowledgement identifier which the SSW must use for acknowledging the completion of initialization phase to the host. The next two bytes (4 and 5) indicate the number of eight byte data frames of user application which must be received by SSW and placed into PSRAM, where the total size of the data frames should be selected to be smaller or equal than the available PSRAM size. The last two bytes (6 and 7) indicate the message identifier that would be used by the host while transmitting the user application.

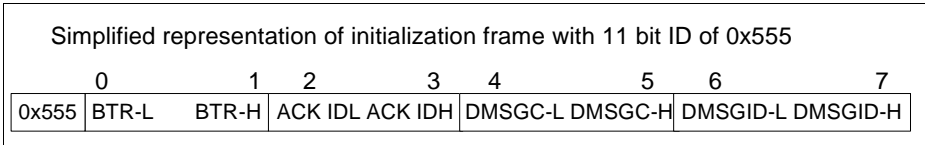


Figure 24-12 Data field of CAN BSL Initialization frame

At start of the BSL port pins P1.4 and P1.5 are used as receive inputs. If the SSW detects initialization frame on P1.4, it would configure P1.5 for TX functionality. If SSW detects initialization frame on P1.5, it configures P1.4 for TX functionality.

Acknowledgement phase

An acknowledgement frame is sent to the host indicating completion of initialization phase.

After SSW computes the baud rate of the host and reconfigures the NBTR register of node-0, it waits until the initialization frame is correctly and fully received. SSW signals its intent to use the bus by transmitting a dominant (0) bit in its ACK slot.

After the dominant bit has been transmitted, an acknowledgement frame using the ACK-ID extracted from the initialization frame is sent to the host. This ACK-ID is used as the LSB-aligned 11-bit message ID and for data bytes 2 and 3. The configured NBTR register value is captured through data bytes 0 and 1. If the size of application intended to be downloaded is greater than the size of PSRAM on the device, a negative acknowledgement as depicted below is sent, where data bytes 4-7 contains the error code of 0xDEADBEEF. SSW enters acknowledgement phase again. If the size of the application is smaller or equal to the PSRAM size, then data byte 4 to 7 of the frame is a direct copy of the equivalent data bytes of the initialization frame. **Figure 24-13** depicts data part of acknowledgement frame.

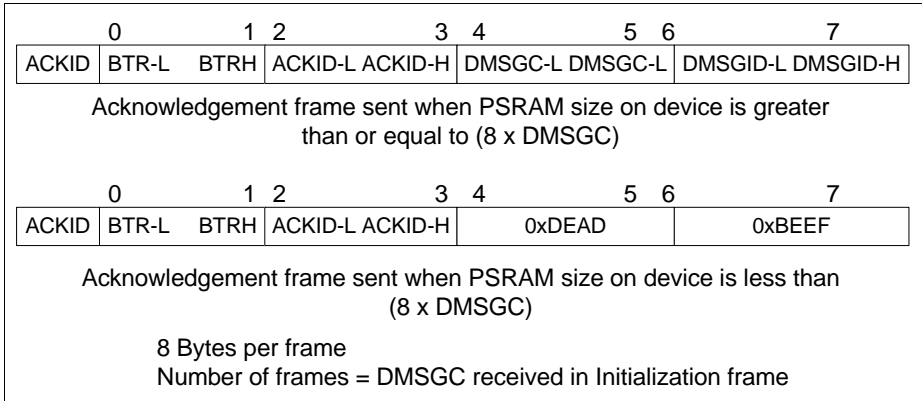


Figure 24-13 CAN Acknowledgement frame

Data transmission phase

Host transmits user application in several CAN frames to the device.

After the SSW transmits the acknowledgement frame, it prepares to receive user application over several CAN frames. Each CAN frame carries eight data bytes and the number of CAN frames is limited to the value retrieved from the DMSGC (Data Message Count) field of the initialization frame. Message identifier is essentially the DMSGID extracted from the initialization frame.

Data received in a frame is placed into PSRAM sequentially. After all frames have been received, SSW cedes control to the first user instruction at the start of PSRAM.

User actions for CAN BSL mode

User must:

- Configure the boot mode pins as described in [Table 24-1](#)
- Reset the device
- Ensure CAN host continuously transmits initial frame described in [Figure 24-12](#)
- Ensure host receives acknowledgement frame and transmits the application stream

24.3.8 Boot Mode Index (BMI)

BMI provides a provision for end user to customize boot sequence. A 32 bit BMI word describes a set of activities that must be performed by SSW. The BMI word must be programmed into page-1 of User Configuration Block-2 (UCB2-Page1).

BMI word along with associated parameters is known as the BMI string. [Figure 24-14](#) depicts this pictorially.

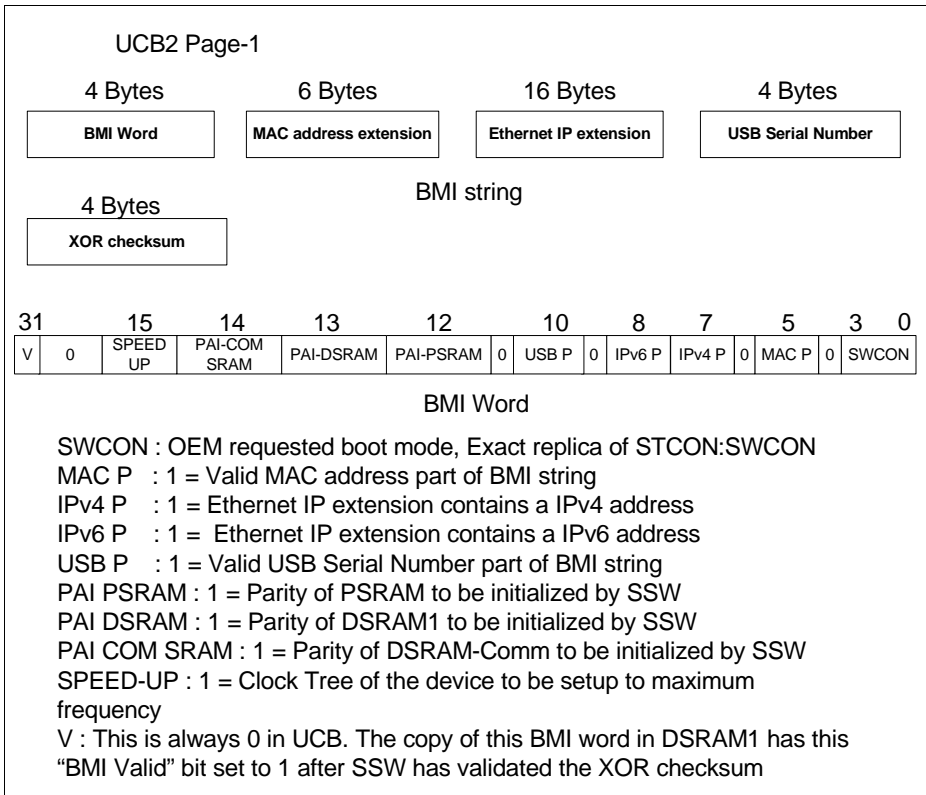


Figure 24-14 BMI String layout

The BMI string is written into UCB2-Page1 by the customer. SSW upon a any reset copies the BMI string into a 64 byte location on DSRAM-1([Figure 24-4](#)). A 4 byte XOR checksum is appended to the string.

Of the 64 bytes, the BMI string is stored starting from the lowest address with the BMI word stored first followed by the MAC address, IP address and USB serial number.

All elements of the BMI string and the XOR checksum are stored in little-endian format. SSW performs byte wise XOR checksum over the 30 bytes BMI string. The checksum variable is initialized with zero and then XOR-ed with the first byte of the BMI string. The resulting checksum is then XOR-ed again with the second byte, and the operation continues until the last byte of the BMI string.

All elements of BMI string are stored linearly without any holes. Following table provides details of the layout.

Table 24-3 BMI string layout offset table

Element	Offset
BMI Word (LSB)	0
MAC (LSB)	4
IP address (LSB)	10
IP address (MSB) for IPv4	13
IP address (MSB) for IPv6	25
USB serial number (LSB)	26
XOR checksum	33

Details of actions taken by SSW are listed in the [Figure 24-15](#).

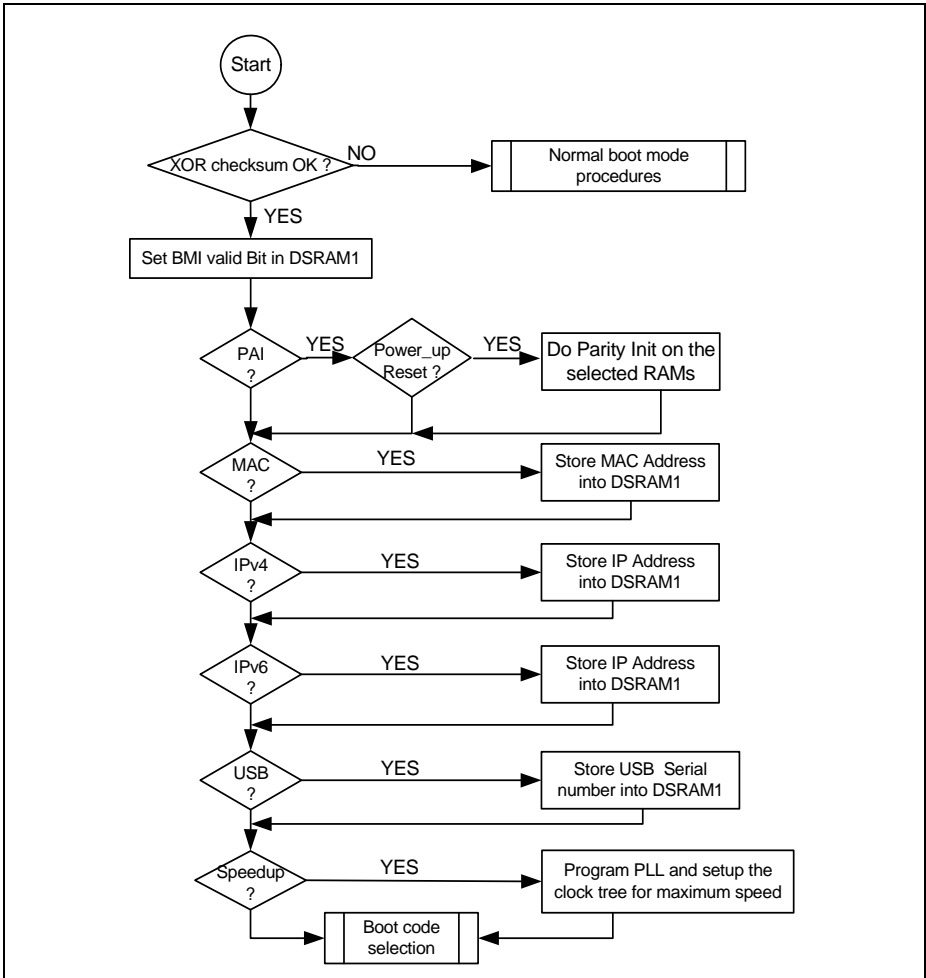


Figure 24-15 BMI Actions

The BMI Speedup feature yields a frequency which is never guaranteed to be maximum system frequency. This is due to the fact that the source clock provided as input to PLL is the fast internal clock f_{OFI} which is never an accurate 24 MHz. SSW programs PLL to ensure that f_{sys} is lower than maximum system frequency.

User actions for BMI

User must:

- Flash BMI string into the User Configuration Block (UCB2-Page1)
- Configure boot mode pins as described in [Table 24-1](#)
- Reset the device

Alternatively

- A currently running program can setup SWCON bit field in the SCU.STCON register
- Clear Reset Source in SCU.RSTSTAT using the register RSTCLR.RSCLR
- Issue a system reset

24.4 Debug behavior

This section describes the Halt after reset (HAR), flash protection, debugger access and diagnostics monitor mode features.

24.4.1 Boot modes and hardware debugger support

The SSW cannot be debugged. It is not possible to halt the CPU after a reset until the SSW has finished its execution. The SSW disables the coresight module thus inhibiting installation of breakpoints and watchpoints. All memory accesses that can otherwise be requested over the debug bus are also inhibited. The coresight module is enabled subject to certain conditions at the time when SSW has finished its activities and is about to cede control to user applications.

Halt after reset feature

Halt after reset feature results in halting of the CPU when the first instruction from the user program enters the CPU pipeline. This is implemented only for Normal boot mode, ABM-0 and ABM-1. It is further applicable only for a PORST case. A breakpoint is installed at the address retrieved from reset vector location of the application exception vector table. However for this to happen, the hardware debugger is expected to have registered its request with coresight for halting the CPU.

The CPU simply cannot be halted while the SSW executes. The hardware debugger can only register a halt request with the coresight while the SSW is executing. This request is picked up by SSW towards the end of its execution resulting in installation of breakpoint on the first user application instruction.

Debugger support after SSW execution

Debugger support is enabled after SSW execution. But this is subject to the state of flash protection. As long as the user flash is not protected, user programs can be debugged. Should the SSW detect that flash has been password protected, it disables the coresight (ARM debug) interface. Hence debugging is not possible on flash protected devices.

Flash access after SSW execution

Special attention must be paid to the table below. Flash access is unconditionally provided for Normal and ABM boot modes regardless of flash protection. CPU can fetch CODE and RO-DATA from the program flash for the two boot modes. Flash access is however only conditionally permitted for BSL and PSRAM boot modes. [Table 24-4](#) lists the criteria for flash and debugger accesses.

Table 24-4 Flash and Debug access policy

HWCON/SWCON	Flash protected	Flash access	Debugger access
UART/CAN BSL	N	Y	Y
UART/CAN BSL	Y	N	N
Normal/ABM	N	Y	Y
Normal/ABM	Y	Y	N
Boot from PSRAM	N	Y	Y
Boot from PSRAM	Y	N	N

Empty flash and debugger behavior

A value of 00000000_H at 0C000004_H (Reset vector) indicates empty flash. In case the SSW detects this values a safe loop in SSW is entered. A hardware debugger can be attached in that situation to program the flash with a user application.

24.4.2 Failures and handling

It is possible to find out the progress made by SSW during its execution and also the reason for boot failures.

Diagnostics Monitor Mode (DMM)

During its course of execution, SSW has either something normal to trace or an error to report. Access to the Coresight system is disabled during SSW execution.

The SCU provides a dedicated register SCU_TSW0 which the SSW can write to. SCU_TSW0 is mirrored by the TCU (Test Control Unit) and the latter's contents can be conveniently scanned out.

This register is known as the TRACE_ERROR_REG and the JTAG instruction for accessing it is the FW_TRACE_ERR (63_H).

Errors classified as recoverable result in a watchdog reset. Fatal errors require a PORST. The power consumption is reduced to a minimum level.

[Figure 24-16](#) represents this concept.

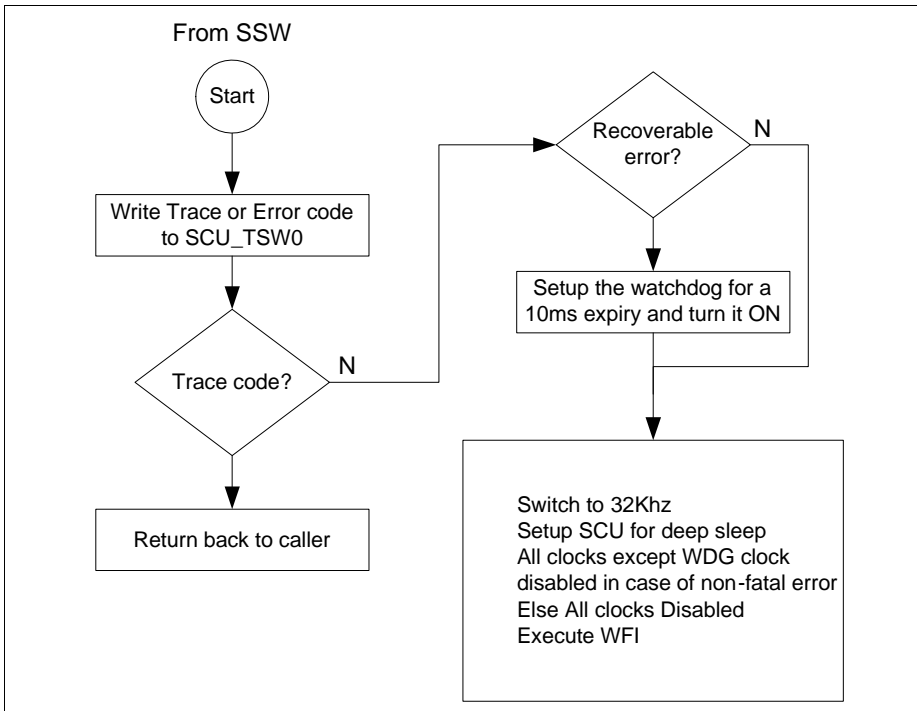


Figure 24-16 Diagnostics monitor mode

Encoding of the trace word

SSW can write a 32 bit word representing either a trace code or an error code. TSW0[15:0] represents the code. TSW0[17:16] represent the code type.

A code type of 0b00 represents invalid code contents, 0b01 represents a trace code, 0b10 represents a fatal error and 0b11 represents a non fatal error.

TSW0[31:18] bits are currently reserved.

List of errors and their classification

Table 24-5 lists errors that lead to SSW launching DMM.

Table 24-5 Error events and codes

Event	Severity	16 bit Code
Flash rampup error	Fatal	1 _H
FCS CRC mismatch	Fatal	2 _H
Invalid SWCON	Fatal	3 _H
MBIST error	Fatal	4 _H
PSRAM boot header CRC mismatch	Fatal	5 _H
ABM header CRC mismatch	Fatal	6 _H
Invalid bootcode	Fatal	7 _H
NMI exception	Fatal	8 _H
Hardfault exception	Fatal	9 _H
Memory management exception	Fatal	A _H
Busfault exception	Fatal	B _H
Usage fault exception	Fatal	C _H
ASC baudrate calculation error	Fatal	D _H
Invalid BMI password error	Fatal	E _H
EVR rampup error	Fatal	F _H
ASC BSL receive error	Fatal	10 _H
CAN BSL errors	Fatal	11 _H

24.5 Power, Reset and Clock

SSW operates at 24 MHz. The fast internal oscillator provides the system clock frequency (f_{OFI}). Frequency scaling is conditionally performed in BMI boot mode of operation based on the BMI word.

Registers modified by SSW

SSW during its course of execution modifies default register settings of a few registers. They are listed in [Table 24-6](#).

Table 24-6 Registers modified by SSW

Startup mode	Register	Bitfield	Value
All	CPU_CPACR	23:20	1111 _b
All	CPU_SHCR	18:16	111 _b
All	CPU_CCR	4:3	11 _b
All	SCU_PRSTAT2	6	0 _b
All	FCE_CLC	1:0	00 _b
All	FCE_CFG0	10:8	000 _b
All	FCE_IR0	31:0	Dependent on calculations
All	FCE_CRC0	31:0	Dependent on calculations
All	FCE_RES0	31:0	Dependent on calculations
All	P1_IOCR4	15:11	10010 _b
ASC BSL	SCU_PRSTAT0	11	0 _b
ASC BSL	U0C0_KSCFG	0	1 _b
ASC BSL	U0C0_BRG	14:10	11111 _b
ASC BSL	U0C0_BRG	25:16	Dependent on detected baud rate
ASC BSL	U0C0_FDR	15:14	01 _b
ASC BSL	U0C0_FDR	9:0	Dependent on detected baud rate
ASC BSL	U0C0_FDR	25:16	Dependent on detected baud rate
ASC BSL	U0C0_DX0CR	2:0 9	001 _b 1 _b
ASC BSL	U0C0_CCR	0 12:8	1 _b 00111 _b
ASC BSL	U0C0_SCTR	1 9:8 21:16 27:24	1 _b 01b 000111b 0111b
ASC BSL	U0C0_TCSR	11:10 8	01 _b 1 _b

Table 24-6 Registers modified by SSW

Startup mode	Register	Bitfield	Value
ASC BSL	U0C0_CCR	9:8 3:0	01 _b 0010 _b
ASC BSL with Frequency scaling (FS)	SCU_OSCHPCTRL	5:4	00 _b
ASC BSL with FS	SCU_PLLCON0	1:0 16	00 _b 0 _b
ASC BSL with FS	SCU_PLLCON1	22:16 14:8	1 _b 1001 _b
ASC BSL with FS	SCU_PLLCON2	0	0 _b
ASC BSL with FS	SCU_SYSCLKCR	17:16	01 _b
ASC BSL with FS	SCU_PLLSTAT	9:0	1110100100 _b
CAN BSL	SCU_OSCHPCTRL	5:4 20:16	00 _b Dependent on external clock frequency
CAN BSL	SCU_PLLCON0	17:16	00 _b
CAN BSL	SCU_SYSCLKCR	17:16	01 _b
CAN BSL	SCU_PLLSTAT	9:0	1110010101 _b
CAN BSL	SCU_PRSTAT1	4	0 _b
CAN BSL	CAN_CLC	1:0	00 _b
CAN BSL	CAN_FDR	9:0 15:14 25:16	1111111111 _b 01 _b Dependent on baud rate
CAN BSL	CAN_NPCR1	2:0	011 _b
CAN BSL	CAN_MCR.CLKSEL	3:0	0001 _b
CAN BSL	CAN_MOAR0/1	28:0	Dependent on data
CAN BSL	CAN_MOAMR0/1	27:0	1FFFFFF _H
CAN BSL	CAN_MOFMR0/1	27:24	1000 _b
CAN BSL	CAN_MODATAL0/1, CAN_MODATAH0/1	31:0	Dependent on data
CAN BSL	P1_IOCRR4	7:3	10010 _b

Table 24-6 Registers modified by SSW

Startup mode	Register	Bitfield	Value
CAN BSL	CAN_NCR0/1	0	0 _b
CAN BSL	CAN_NBTR0/1	5:0,14:8 7:6	Frequency/Baud rate dependent 11b
CAN BSL	CAN_NFCR0/1	20:19	10 _b
CAN BSL	CAN_LIST1/2	24 23:16 15:8	0 _b 1H 1H
CAN BSL	P1_IOCRR4	15:11	10001 _b

Debug and Trace System

25 Debug and Trace System (DBG)

The XMC4300 Series Microcontrollers provide a large variety of debug, trace and test features. They are implemented with a standard configuration of the ARM CoreSight™ module together with a daisy chained standard TAP controller. Debug and trace functions are integrated into the ARM Cortex-M4. The debug system supports serial wire debug (SWD) and trace functions in addition to standard JTAG debug and parallel trace.

References

- [22] Cortex-M4 Technical Reference Manual
- [23] CoreSight™ ETM-M4 Technical Reference Manual
- [24] CoreSight™ Technology System Design Guide
- [25] CoreSight™ Components Technical Reference Manual
- [26] ARM Debug Interface v5 Architecture Specification
- [27] Embedded Trace Macrocell Architecture Specification
- [28] ARMv7-M Architecture Reference Manual

25.1 Overview

The Debug and Trace System implements ARM CoreSight™ debug and trace features with the objective of debugging the entire SoC. The CoreSight™ infrastructure includes a debug subsystem and a trace subsystem. The debug functionality includes processor halt, single-step, processor core register access, Vector Catch, unlimited software break points and full system memory access. The debug function includes a breakpoint unit supporting 2 literal comparators and 6 instruction comparators and a watchpoint unit supporting 4 watchpoints. The processing element (CPU) is paired with an instruction/data ETM (ETM-M4). CoreSight™ enables different trace sources to be enabled into one stream. The unique trace stream, marked with suitable identifiers and timestamps. Trace can be done using either a 4-bit parallel or a Serial Wire interface. Less data can be traced with Serial Wire interface, but only one output pin is required for application. Parallel trace has a greater bandwidth, but uses 5 more pins.

Features

The accurate Debug and Trace System provides the following functionality:

- Serial Wire Debug Port (SW-DP)
- JTAG Debug Port (SWJ-DP)
- Flash Patch Breakpoint (FPB)
- Data Watchpoint and Trace (DWT)
- Embedded Trace Module (ETM)

Debug and Trace System (DBG)

- Instrumentation Trace Macrocell (ITM)
- Trace Port Interface Unit (TPIU)
- Halt after reset (HAR)

Note: Please refer to ARM Reference Documentation Cortex-M4-r0p0 for more detailed information on the debug and trace functionality.

Application Mapping

Table 25-1 Debug System available features mapped to functions

SW-DP	Provides Serial Wire Debug, which allows to debug via 2 pins. Instrumentation Trace is provided via a third pin.
SWJ-DP	This debug port provides native JTAG debug capabilities.
FPB	The FPB implements hardware breakpoints and patches code and data from code space to system space.
DWT	Implemented watch points, trigger resources, and system profiling. The DWT contains four comparators that can be configured as a hardware watchpoint, an ETM trigger, a PC sampler event trigger or a data address sampler event trigger, data sampler, interrupt trace and CPU statistics
ETM	The ETM provides Instruction Trace capabilities
ITM	Application driven trace source, supports printf style debugging. The ITM generates trace information as packets out of four sources (Software Trace, Hardware Trace, Time Stamping and Global System Time Stamping).
TPIU	The TPIU encodes and provides trace information to the debugger. As ports the single wire viewer (TRACESWO) or 4-bit Trace Port (TRACEDATA[3:0], TRACECLK) can be used.
HAR	Allows to halt the CPU before application code is entered.

Block Diagram

The Debug and Trace system block diagram is shown in **Figure 25-1**.

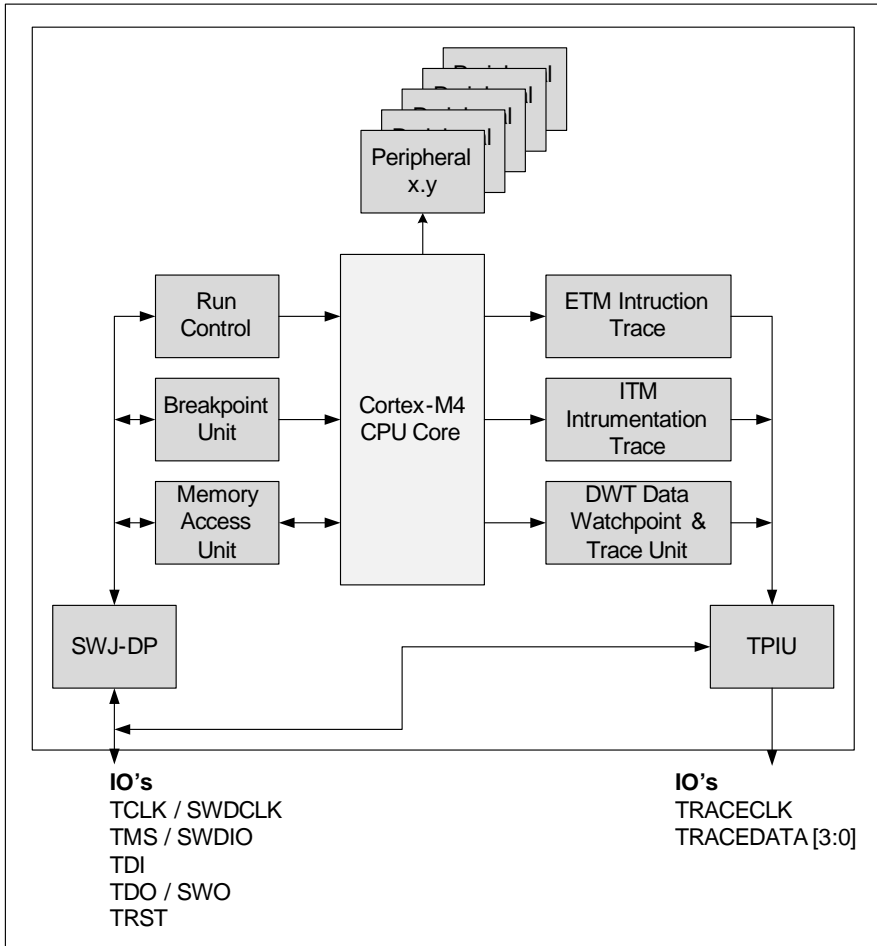


Figure 25-1 Debug and Trace System block diagram

25.2 Debug System Operation

The Debug System provides general debug options and additional trace functions. Debug options are based on break points and CPU halt. The trace capability supports data access trace and instruction execution trace.

25.2.1 Flash Patch Breakpoint (FPB)

The FPB implements hardware breakpoints. Six instruction comparators can be configured to generate a breakpoint instruction to the CPU on a match. The original M4 code patch function is not available.

25.2.2 Data Watchpoint and Trace (DWT)

The four DWT comparators can be configured to generate PC sampling packets at defined intervals, PC or Data watch point packets and a Watch point event to halt the CPU. To enable the features, the DWT provides counters with clock cycle, folded instructions, load store unit operation, sleep cycles, clock per instruction and interrupt overhead count.

25.2.3 Instrumentation Trace Macrocell (ITM)

The ITM supports printf style debugging and is an application trace source. The ITM is available to trace application software execution, and allows to emit diagnostic system information. Three different sources are supported to emit trace information as packet, which are software trace, hardware trace and time stamping. The software trace allows software to write directly to ITM stimulus register using printf function. For hardware trace the ITM emits packets generated by the DWT. Relative to packets the ITM emits timestamps, which are generated by a 48-bit counter. The packets emitted by the ITM are output to the trace port interface (TPIU). The TPIU formatter adds some extra packets and then outputs the complete packet sequence to the debugger. The ITM function can be activated by the TRCEN register bit, which is located in the Debug Exception and Monitor control register. ITM data can also be transferred using the Serial Wire interface. The ITM data are the only Trace source data which can be transferred via the Serial Wire interface.

25.2.4 Embedded Trace Macrocell (ETM)

The ETM enables program execution reconstruction. As a short description, data are traced using the DWT component or the ITM, whereas instructions are traced using ETM. The ETM transmits the information as packets and is triggered by internal resources. These internal trigger resources must be programmed independently and the trigger source is selected using the available Trigger Event Register. Available events are address match, provided by an address comparator or a logic equation between two events. The trigger source are one of the DWT module provided comparators (four are

Debug and Trace System (DBG)

available). This allows to monitor clock cycle matching events and data address matching events. The packets which are generated by the ETM are transmitted on the Trace Port output Unit (TPIU). The TPIU adds some extra packets and transmits the complete packet sequence to the debug tool.

25.2.5 Trace Port Interface Unit (TPIU)

The TPIU collects on-chip trace data from ITM and ETM and sends this debug data to the external trace capture hardware. The TPIU uses dedicated trace ports. Maximum available trace ports are four.

25.3 Power, Reset and Clock

For requirements based on power, reset and clock signaling consult CoreSight™ Technology System Design Guide [24] for detailed information. Note, there is no power management implemented for the debug system.

25.3.1 Reset

Reset and implementation is provided according to the ARM reference documentation [22].

25.3.1.1 CoreSight™ resets

The SWJ-DP and SW-DP register are in the power on reset domain. Besides this reset a tool controlled Debug reset can be generated based on a debug register configuration. Activating the reset request in the debug control and status register results in an activation of the CTRL/STAT.CDBGRESTREQ. The reset allows a debugger to reset the debug logic in a CoreSight™ system without affecting the functional behavior of the target system. The Debug logic is reset by system reset, if no tool is registered at the debug system.

System Reset (Warm Reset)

A System or warm reset initializes the majority of the processor, excluding NVIC and debug logic, (FPB, DWT, and ITM). The System reset affects the Debug system only, if the tool is not registered (CTRL/STAT.CDBGPWRUPREQ not set).

SWJ-DP reset

nTRST reset initializes the state of the JTAG SWJ-DP TAP controller. Normal processor operation is not affected.

SW-DP reset

Only the PORESETn reset initializes the SW-DP and the other debug logic.

Normal operation

During normal operation the resets PORESETn SYSRESETn and DAPRESETn are deasserted. If the SWJ-DP or SWDP ports are not in use, nTRST must be tied to 0 or 1.

25.3.1.2 Serial Wire interface driven system reset

The CTRL/STAT.SYSRESETREQ allows to reset the CPU core in Serial Wire interface mode. The CTRL/STAT.SYSRESETREQ is asserted when the CTRL/STAT.SYSRESETREQ bit of the Application Interrupt and Reset Control register is set. This causes a reset, intended to force a large system reset of all major components, except for debug logic. [25].

25.4 Initialization and System Dependencies

This chapter provides information about Debug access and Flash protection, Halt After Reset sequences, Halting Debug and Peripheral suspend, Timestamping for Trace and the available tool interfaces and how to enable the tool interface. Additionally the ID Codes and the ROM Table is presented, including the values a debug tool uses to identify the device and available debug functionality. The last section shows the JTAG Debug instruction code definition.

25.4.1 Debug accesses and Flash protection

The Flash has integral measures to protect its content from unauthorized access and modification, see the section Read and Write Protection in the PMU chapter and startup chapter. Special care is taken, that the debugger can't bypass this protection. Because of this, per default and after a system reset the debug interface is disabled. Depending on the boot scenario and the Flash protection setup, the Startup Software enables the debug interface. If it is left disabled, the user can define a protocol, e.g. a password-protected unlock sequence via an SPI port, to enable the debug interface.

25.4.2 Halt after reset

The XMC4300 product supports two different possibilities of halt after reset. One after a power on reset ("Power-on Reset"), which is called Cold Reset Halt situation or Halt after reset (HAR) and the second one after a system reset, which is called Warm Reset Halt situation. For a HAR the debug tool has to register during the start up software (SSW) execution time. The SSW halts at the end of SSW on a successful tool registration, before application code is entered. The Warm Reset Halt is based on break point setting on the very first application code line and is entered by a system reset.

Debug and Trace System (DBG)

For security reasons it is required to prevent a debug access to the processor before and while the boot firmware code from ROM (SSW) is being executed. A bit DAPSA, (DAP has system access) in the SCU is implemented, allowing the access from CoreSight™ debug system to the processor core. The default value of this bit is disabled debug access. The register is reset by System Reset. The System Reset disables the debug access each time SSW is being executed. At the end of the SSW the DAPSA is enabled always (independent of any other register setting or signaling value), to allow debug access to the CPU. A tool accessing the SoC during the SSW execution time reads back a zero and a write is going to a virtual, none existing address.

In a HAR situation the system always comes from a "Power-on Reset". A tool can register for a HAR by sending a pattern enabling the CTRL/STAT.CDBGPWRUPREQ and the DHCSR.C_DEBUGEN [28] register inside the CoreSight™ module. The registration has to be done after the "Power-on Reset" in a time interval smaller then the time the SSW is executed. This registration time is called tssw and the bits must be set before tssw is expired. The timing value for tssw needs to be handled by the debug tool software (no hardware timer available) and informs the tool software during a Cold Reset about the time frame available to set the HAR conditions.

The following figure (**Figure** HAR - Halt After Reset) shows the software flow based on the modules participating to the HAR.

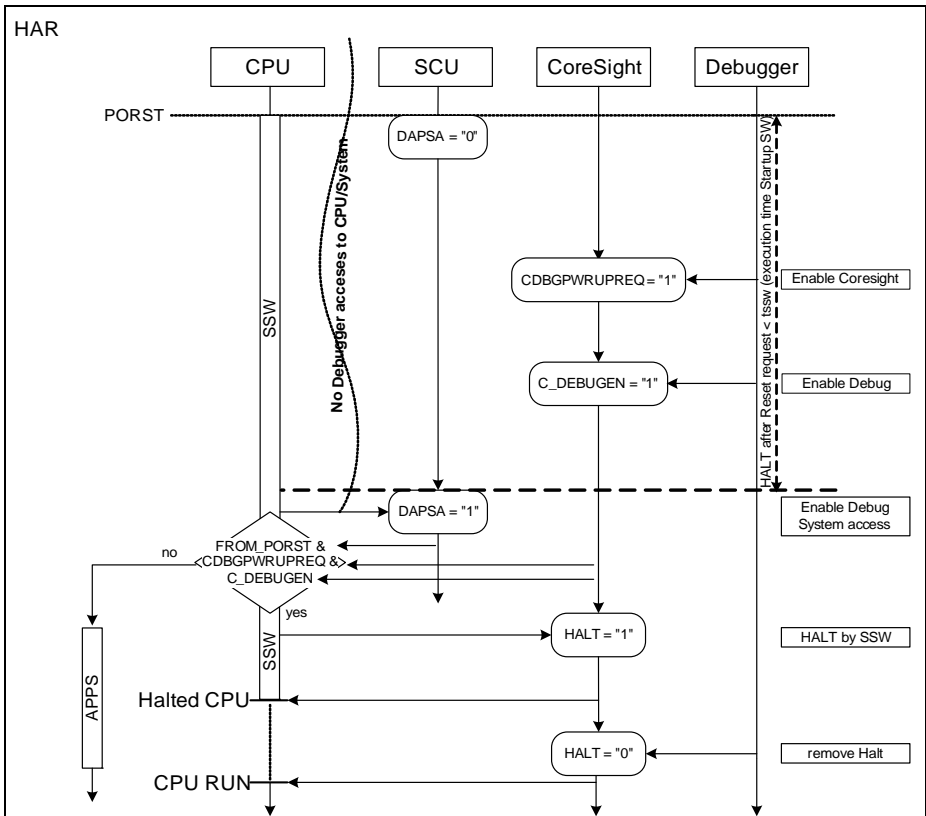


Figure 25-2 HAR - Halt After Reset

A Halt after system reset (Warm Reset) can be achieved by programming a break point at the first instruction of the application code. Before a Warm Reset the CTRL/STAT.CDBGPWRUPREQ and the DHCSR.C_DEBUGEN setting has to be ensured. After a system reset, the HAR situation is not considered, as the reset is not coming from "Power-on Reset".

Note: The CTRL/STAT.CDBGPWRUPREQ and DHCSR.C_DEBUGEN does not have to be set after a system reset, if they have already been set before.

A tool hot plug condition allows to debug the system starting with a tool registration (setting CTRL/STAT.CDBGPWRUPREQ register) and a debug system enable (setting the DHCSR.C_DEBUGEN register). Afterwards break points can be set or the CPU can directly by HALTED by an enable of C_HALT.

Debug and Trace System (DBG)

The following Figure (Figure Hot Plug and Warm Reset) illustrates the debug tool HOT PLUG situation or the Halt after Warm Reset (system reset) and how to proceed to come to a Halt situation.

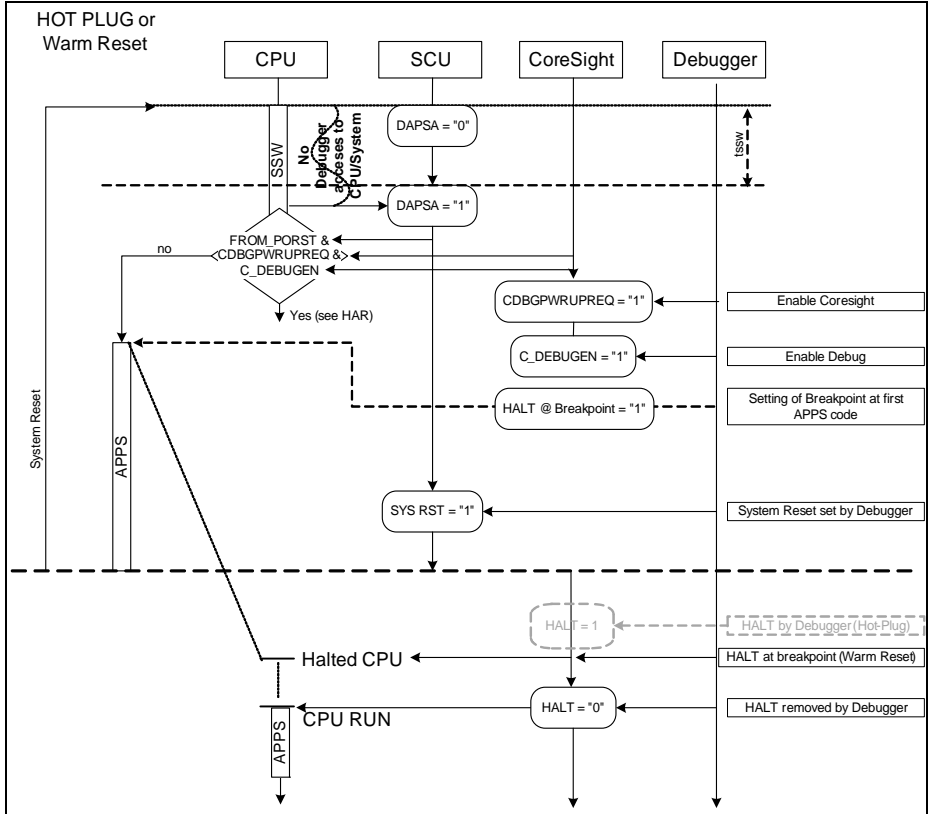


Figure 25-3 HOT PLUG or Warm Reset

25.4.3 Halting Debug and Peripheral Suspend

If the program execution of the CPU is stopped by the debugger, e.g. with a breakpoint, it is possible to suspend the peripherals as well. This allows to debug critical states of the whole microcontroller. It is particularly useful, e.g. to suspend the Watchdog Timer as it can't be serviced by a halted CPU.

In other cases it is important to keep some peripherals running, e.g. a PWM or a CAN node, to avoid system errors or even critical damage to the application. Because of this,

Debug and Trace System (DBG)

the peripherals allow to configure how they behave when the CPU enters the halting debug mode.

It can be decided at the peripheral to support a Hard Suspend or a Soft Suspend. At a Hard Suspend situation the clock at the peripheral is switched off immediately, without waiting on acknowledge from the module. At a soft suspend the peripheral can decide when to suspend, usually at the end of the actual active transfer.

A Watchdog timer is only running when the suspend bus is not active. This is particularly useful as it can't be serviced by a halted CPU. A configuration option is available, which allows to enable the Watchdog timer also during suspend. This allows to debug Watchdog behavior, if a debugger is connected.

The user has to ensure, that always only those peripherals are sensitive to suspend, which are intended to be. To address this, each peripheral supporting suspend does have an enable register which allows to enable the suspend feature. The following table ([Table 25-2](#)) shows the peripherals, supporting or not supporting peripheral suspend or detailed information on the peripheral suspend behavior during soft suspend can be found at the respective peripheral chapter.

Table 25-2 Peripheral Suspend Support

Peripheral	Supported	Default mode	Hard Suspend	Soft Suspend	Suspend Reset
RTC	no	---	---	---	---
WDT ¹⁾	yes	active	yes	no	system reset
LEDTS	yes	not active	yes	no	debug reset
SDMMC	no	---	---	---	---
EBU	no	---	---	---	---
ETH	no	---	---	---	---
USB	no	---	---	---	---
ECAT	no	---	---	---	---
USIC	yes	not active	no	yes	debug reset
MultiCAN	yes	not active	yes	yes	debug reset
VADC	yes	not active	yes	yes	debug reset
DSD	yes	not active	yes	yes	debug reset
DAC	no	---	---	---	---
CCU4 ¹⁾	yes	not active	yes	yes	system reset
CCU8 ¹⁾	yes	not active	yes	yes	system reset
POSIF ¹⁾	yes	not active	yes	yes	system reset

Debug and Trace System (DBG)

- 1) A system reset results in a suspend configuration loss. If it is required to have suspend configuration available after system reset, a HW breakpoint has to be set at the first instruction of use code and reconfiguration of suspend behavior at the peripheral has to be performed again.

25.4.4 Timestamping

A 48-bit timestamping capability is required by the debug system in order to get accurate correlation between the ETM trace and trace data from ITM and DWT. This is also named global timestamping. Timestamp packets encode timestamp information, generic control and synchronization information. The Timestamp counter is a free running global counter.

A synchronization packet is a timestamp packet control. It is emitted at each DWT trigger (DWT must be configured to trigger the ITM).

25.4.5 Debug tool interface access (SWJ-DP)

Debug capabilities can be accessed by a debug tool via Serial Wire (SW) or JTAG interface (JTAG - Debug Port). By default, the JTAG interface is active. The User might switch to SW interface as the full JTAG pins are not available to the user. To enable SW interface a dedicated JTAG sequence on TMS/SWDIO and TCK/SWCLK is required to switch to the Serial Wire Debug interface. A successful sequence disables JTAG interface and enables SW interface.

The sequences to do this are described in [Section 25.4.5.1](#) and [Section 25.4.5.2](#)

25.4.5.1 Switch from JTAG to SWD

The sequence for switching from JTAG to SWD is:

- Send 50 or more TCK cycles with TMS = 1
- Send the 16-bit sequence on TMS = 1110011110011110 (0xE79E LSB first)
- Send 50 or more TCK cycles with TMS = 1

25.4.5.2 Switch from SWD to JTAG

The sequence for switching from SWD to JTAG is:

- Send 50 or more TCK cycles with TMS = 1
- Send the 16-bit sequence on TMS = 1110011100111100 (0xE73C LSB first)
- Send 50 or more TCK cycles with TMS = 1

25.4.6 ID Codes

Available ID Codes are used by a debug tool to identify the available debug components during tool setup.

Table 25-3 ARM CoreSight™ Component ID codes

ID	Value ¹⁾	Description
JTAG IDCODE	XXXX X083 _H	The TAP JTAG IDCODE
SWJ-DP	4BA0 0477 _H	The ARM JTAG ID
SW_DP	2BA0 1477 _H	The ARM SW-DP ID

1) For "X" values please refer to Data Sheet

25.4.7 ROM Table

To identify Infineon as manufacturer and XMC4300 as device, the ROM table has to be read out.

Table 25-4 Perpheral ID Values of XMC4300 ROM Table

Name	Offset	Value ¹⁾	Bits	Reference	Infineon JTAG ID Code
PID0	FE0 _H	XXXXXXXX _B	[7:0]	Part Number [7:0]	JTAG IDCODE [19:12]
PID1	FE4 _H	0001XXXX _B	[7:4] [3:0]	JEP106 ID code [3:0] Part Number [11:8]	JTAG IDCODE [4:1] JTAG IDCODE [23:20]
PID2	FE8 _H	XXXX1100 _B	[7:4] [3] [2:0]	Revision JEDEC assigned ID fields JEP106 ID code [6:4]	JTAG IDCODE [31:28] '1' JTAG IDCODE [7:5]
PID3	FEC _H	00000000 _B	[7:4] [3:0]	RevAnd, minor revision field Customer-modified block (if non-zero)	
PID4	FD0 _H	00000000 _B	[7:4] [3:0]	4KB count JEP106 continuation code	4KB count JTAG IDCODE [11:8]

1) For "X" values please refer to Data Sheet

25.4.8 JTAG debug port

A standard JTAG IEEE1149 Boundary-Scan statemachine is implemented. It includes all mandatory instructions (SAMPLE/PRELOAD, EXTEST) and also some optional instructions (IDCODE, CLAMP, HIGHZ), and some custom/optional instructions are implemented. The optional RUNBIST instruction is not used and will be treated as BYPASS. No USERCODE-instruction is implemented, it will show only 0 values

Table 25-5 JTAG INSTRUCTIONS

Opcode	Range	Type	Instruction
0000 0000	00H	Reserved	
0000 0001	01H - 08H (8 instr.)	IEEE1149 Boundary-Scan	INTEST
0000 0010			SAMPLE/PRELOAD
0000 0011			RUNBIST
0000 0100			IDCODE
0000 0101			USERCODE
0000 0110			CLAMP
0000 0111			EXTEST
0000 1001- 0000 1111			Reserved
0001 0001 - 0100 1111	11H - 4FH 63instr.		
1111 1111	FFH	IEEE 1149.1	BYPASS

JTAG Instruction Definition

BYPASS

The BYPASS instruction bypasses all serial register path, which are accessed over the JTAG TAP port. In the Capture-DR state the rising edge of the TCK clock sets the bypass register to zero. The bypass instruction is primarily used to allow a shorter access path to cascaded devices when the JTAG interface connects a number of devices in series. All unused instructions must connect the TAP controller bypass register output to the Test Access Port output TDO. The BYPASS instruction has no effect on the operation of the device.

25.5 Debug System Registers

For CoreSight™ register overview and detailed register definitions, please refer to ARM documentation CoreSight™ Components Technical Reference Manual [25] and ARMv7-M Architecture Reference Manual [28].

25.6 Debug and Trace Signals

XMC4300 MC Product family provides debug capability using ARM CoreSight™ Debug port SWJ-DP. SWJ-DP includes two debug interfaces namely JTAG Debug Port (JTAG-DP) and Serial Wire Debug Port (SW-DP).

The JTAG-DP interface has 4 (without Reset pin TRST for low pin package) or 5 Pins, see [Table 25-6](#).

The serial wire Debug Port has 2 (Clock + Bidirectional data) or 3 pins (Clock + Bidirectional data + Asynchronous Trace output). They are overlaid on the JTAG-DP pins (TCK, TMS and TDO) for efficient use of package pins, see [Table 25-7](#).

Additionally 5 ETM trace port output signals (TRACECLK, TRACEDATA[3:0]) are available, see [Table 25-8](#).

Sub chapters below additionally describe pull resistors to the IO and the suggested debug connector pin assignment.

Table 25-6 JTAG Debug signal description

Signal	Direction	Function
TCK	I	JTAG Test Clock. This pin is the clock for debug module when running in JTAG debug mode.
TMS	I	JTAG Test Mode Select. The TMS pin selects the next state in the TAP state machine.
TDI	I	JTAG Test Data In. This is the serial data input for the shift register
TDO	O	JTAG Test Data Output. This is the serial data output from the shift register. Data is shifted out of the device on the negative edge of the TCK signal
TRST	I	JTAG Test reset.

Table 25-7 Serial Wire Debug signal description

Signal	Direction	Function
SWDCLK	I	Serial Wire Clock. This pin is the clock for debug module when running in Serial Wire debug mode.
SWDIO	I / O	Serial Wire debug data IO. Used by an external debug tool to communicate with and control the Cortex-M4 CPU.
SWO	O	Serial Wire Output. The SWO pin provides data from the ITM and/or ETM for an external debug tool to evaluate the instrumentation trace.

Table 25-8 ETM Trace Port signal description

Signal	Direction	Function
TRACECLK	O	Trace Clock. Provides the sample clock for trace data on the TRACEDATA pins when tracing is enabled by an external tracing tool.
TRACEDATA[3:0]	O	Trace Data bits 3 to 0. Provide ETM trace data when tracing is enabled by an external debug tool. The debug tool can interpret the compressed information and make it available to the user.

ETM Trace port output enable

The ETM module allows to control the trace port signal on a shared GPIO at the IO port level. The enabling is done by the tool software, by configuring in the ETM main Control register (ETMCR) [23].

25.6.1 Internal pull-up and pull-down on JTAG pins

It is a requirement to ensure none floating JTAG input pins, as they are directly connected to flip-flops controlling the debug function. To avoid any uncontrolled I/O voltage levels internal pull-up and pull-downs on JTAG input pins are provided.

- TRST: Internal pull-down
- TMS/SWDIO: Internal pull-up
- TCK/SWCLK: Internal pull-down

25.6.2 Debug Connector

The suggested connector is the Cortex Debug and ETM connector, which is a 20-pin connector. The connector supports JTAG debug, Serial-Wire debug, Serial Wire viewer (via SWO connection when Serial Wire debug mode is used) and instruction trace operations. The following **Figure 25-4** shows the 20-pin connector for debug and trace.

There may be systems, which required HW to detect debugger presence. This can be enabled by using the GNDDetect pin of the Debug and Trace connector. The pin is driven low by the tool, when the tool connector is plugged into the connector at the PCB. GNDDetect for tool detection requires to have a weak pull-up on the PCB, connected to this pin. At the connector it is suggested not to have the KEY pin available as the KEY is used to identify the correct connector plug-in.

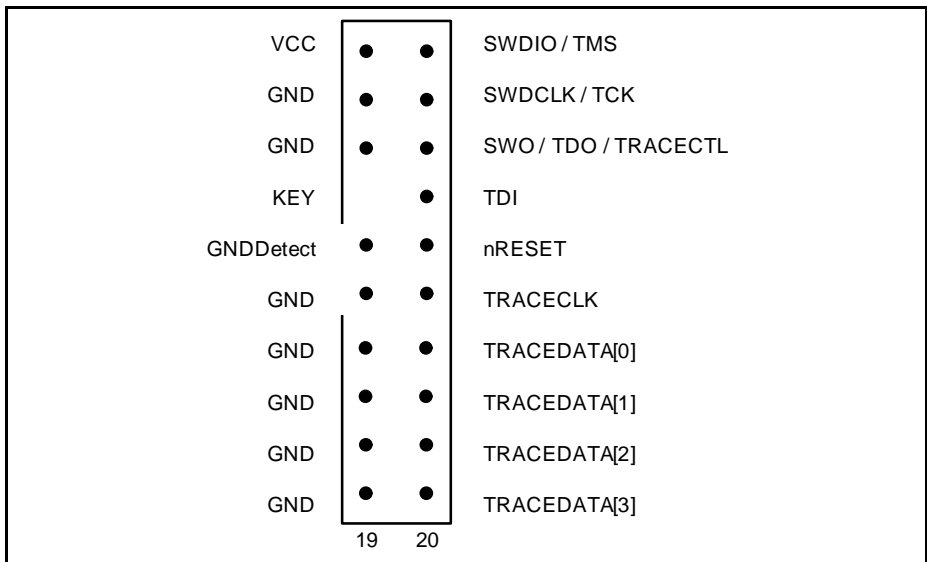


Figure 25-4 Debug and Trace connector

Lists of Figures and Tables

List of Figures

Figure 1-1	System Block Diagram	1-3
Figure 2-1	Cortex-M4 Block Diagram	2-3
Figure 2-2	Core registers	2-6
Figure 2-3	Memory map	2-20
Figure 2-4	Ordering of Memory Accesses	2-22
Figure 2-5	Little-endian format	2-24
Figure 2-6	Vector table	2-30
Figure 2-7	Exception stack frame	2-35
Figure 2-8	Example of SRD use	2-53
Figure 2-9	CFSR	2-75
Figure 2-10	Interrupt Priority Register	2-91
Figure 3-1	Multilayer Bus Matrix	3-2
Figure 4-1	Block Diagram on Service Request Processing	4-2
Figure 4-2	Example for Service Request Distribution	4-3
Figure 4-3	DMA Line Handler	4-8
Figure 4-4	Event Request Unit Overview	4-14
Figure 4-5	Event Request Select Unit Overview	4-15
Figure 4-6	Event Trigger Logic Overview	4-16
Figure 4-7	ERU Cross Connect Matrix	4-18
Figure 4-8	Output Gating Unit for Output Channel y	4-19
Figure 4-9	ERU Interconnects Overview	4-33
Figure 5-1	GPDMA Block Diagram	5-3
Figure 5-2	Transfer Hierarchy for Peripherals	5-5
Figure 5-3	Transfer Hierarchy for Memory	5-6
Figure 5-4	Hardware Handshaking Interface	5-10
Figure 5-5	Software Handshaking Interface	5-11
Figure 5-6	Example of Destination Scatter Transfer	5-18
Figure 5-7	Source Gather when SGR.SGI = 0x1	5-19
Figure 5-8	Breakdown of Block Transfer	5-23
Figure 5-9	Channel FIFO Contents	5-23
Figure 5-10	Breakdown of Block Transfer where max_abrst = 2, Case 1	5-24
Figure 5-11	Channel FIFO Contents	5-24
Figure 5-12	Breakdown of Block Transfer where max_abrst = 2, Case 2	5-25
Figure 5-13	Channel FIFO Contents	5-25
Figure 5-14	Multi-Block Transfer Using Linked Lists When <code>CFG.SS_UPD_EN</code> is set to '1'	5-28
Figure 5-15	Multi-Block Transfer Using Linked Lists When <code>CFG.SS_UPD_EN</code> is set to '0'	5-28
Figure 5-16	Mapping of Block Descriptor (LLI) in Memory to Channel Registers When <code>CFG.SS_UPD_EN = 1</code>	5-31
Figure 5-17	Mapping of Block Descriptor (LLI) in Memory to Channel Registers When	

List of Figures

	CFG.SS_UPD_EN = 0 5-31	
Figure 5-18	Multi-Block DMA Transfer with Source Address Auto-Reloaded and Contiguous Destination Address	5-37
Figure 5-19	DMA Transfer Flow for Source Address Auto-Reloaded and Linked List Destination Address	5-38
Figure 5-20	Multi-Block DMA Transfer with Source and Destination Address Auto-Reloaded	5-41
Figure 5-21	DMA Transfer Flow for Source and Destination Address Auto-Reloaded .	5-42
Figure 5-22	Multi-Block DMA Transfer with Source Address Auto-Reloaded and Linked List Destination Address	5-46
Figure 5-23	DMA Transfer Flow for Source Address Auto-Reloaded and Linked List Destination Address	5-47
Figure 5-24	Multi-Block DMA Transfer with Linked List Source Address and Contiguous Destination Address	5-50
Figure 5-25	DMA Transfer Flow for Source Address Auto-Reloaded and Linked List Destination Address	5-51
Figure 5-26	Multi-Block with Linked Address for Source and Destination.	5-54
Figure 5-27	Multi-Block with Linked Address for Source and Destination Where SAR and DAR Between Successive Blocks are Contiguous	5-55
Figure 5-28	DMA Transfer Flow for Source and Destination Linked List Address	5-56
Figure 5-29	DMA Event to Service Request Flow	5-57
Figure 6-1	FCE Block Diagram	6-3
Figure 6-2	CRC kernel configuration register	6-4
Figure 6-3	CRC kernel status register	6-5
Figure 6-4	Register monitoring scheme	6-7
Figure 7-1	Cortex-M4 processor address space	7-2
Figure 8-1	PMU Block Diagram	8-1
Figure 8-2	Prefetch Unit	8-3
Figure 8-3	Basic Flash Program Sequence	8-16
Figure 9-1	Watchdog Timer Block Diagram	9-3
Figure 9-2	Reset without pre-warning	9-4
Figure 9-3	Reset after pre-warning	9-5
Figure 9-4	Reset upon servicing in a wrong window	9-6
Figure 9-5	Reset upon servicing with a wrong magic word	9-6
Figure 10-1	Real-Time Clock Block Diagram Structure	10-2
Figure 10-2	Block Diagram of RTC Time Counter	10-3
Figure 11-1	SCU Block Diagram	11-3
Figure 11-2	Service Request Subsystem	11-6
Figure 11-3	Parity Error Control Logic	11-8
Figure 11-4	Trap Subsystem	11-9
Figure 11-5	Offset adjustment of the RESULT curve	11-12
Figure 11-6	Gain adjustment of the RESULT curve	11-13

List of Figures

Figure 11-7	Out of Range Comparator Control	11-14
Figure 11-8	System States Diagram	11-15
Figure 11-9	Hibernate state in time keeping mode	11-17
Figure 11-10	Hibernate controlled with external voltage regulator	11-18
Figure 11-11	Initial power-up sequence	11-19
Figure 11-12	Supply Voltage Monitoring	11-20
Figure 11-13	Alternate function selection of HIB_IO_0 and HIB_IO_1 pins of Hibernate Domain 11-23	
Figure 11-14	System Level Power Control example - externally controlled with two pins 11-24	
Figure 11-15	System Level Power Control example - externally controlled with single pin 11-25	
Figure 11-16	System Level Power Control example - internally controlled	11-26
Figure 11-17	EtherCAT Reset Control	11-28
Figure 11-18	Clock Control Unit	11-30
Figure 11-19	Clock Generation Block Diagram	11-31
Figure 11-20	Clock Selection Unit	11-34
Figure 11-21	External Clock Selection	11-37
Figure 11-22	External Clock Input Mode for the High-Precision Oscillator	11-38
Figure 11-23	External Crystal Mode Circuitry for the High-Precision Oscillator	11-38
Figure 11-24	PLL Normal Mode	11-41
Figure 11-25	PLL Prescaler Mode Diagram	11-44
Figure 11-26	PLLUSB Block Diagram	11-49
Figure 11-27	Initialization sequence	11-53
Figure 11-28	System Level Power On Reset Control	11-55
Figure 11-29	Clock initialization sequence	11-58
Figure 12-1	LEDTS Block Diagram	12-3
Figure 12-2	Time-Multiplexed LEDTS Functions on Pin (Example)	12-6
Figure 12-3	Activate Internal Compare/Line Register for New Time Slice	12-7
Figure 12-4	LED Function Control Circuit (also provides pad oscillator enable).	12-9
Figure 12-5	Touch-Sense Oscillator Control Circuit	12-10
Figure 12-6	Hardware-Controlled Pad Turns for Autoscan of Four TSIN[x] with Extended Frames 12-11	
Figure 12-7	Pin-Low-Level Extension Function	12-12
Figure 12-8	Over-rule Control on Pin for Touch-Sense Function	12-20
Figure 13-1	SDMMC Block Diagram	13-3
Figure 13-2	Data Transfer sequence	13-12
Figure 13-3	Synchronous Abort sequence	13-15
Figure 13-4	External Pin Connections of SDMMC	13-92
Figure 14-1	ETH Block Diagram	14-5
Figure 14-2	Descriptor Ring and Chain Structure	14-27
Figure 14-3	TxDMA Operation in Default Mode	14-32
Figure 14-4	TxDMA Operation in OSF Mode	14-34

List of Figures

Figure 14-5	Receive DMA Operation	14-37
Figure 14-6	Rx/Tx Descriptors	14-41
Figure 14-7	Receive Descriptor Format	14-42
Figure 14-8	Transmit Descriptor Format	14-48
Figure 14-9	Receive Descriptor Fields When DMA Clears the Own Bit	14-53
Figure 14-10	Transmit Descriptor Fields	14-56
Figure 14-11	Transmitter Descriptor Fields - Alternate (Enhanced) Format	14-59
Figure 14-12	Receive Descriptor Fields - Alternate (Enhanced) Format	14-66
Figure 14-13	Wake-Up Frame Filter Register	14-76
Figure 14-14	SMA Interface Block	14-80
Figure 14-15	Management Write Operation	14-81
Figure 14-16	Management Read Operation	14-82
Figure 14-17	Media Independent Interface	14-83
Figure 14-18	Reduced Media Independent Interface	14-84
Figure 14-19	RMII Block Diagram	14-84
Figure 14-20	Transmission Bit Ordering	14-85
Figure 14-21	Start of MII and RMII Transmission in 100 Mbit/s Mode	14-86
Figure 14-22	End of MII and RMII Transmission in 100 Mbit/s Mode	14-86
Figure 14-23	Start of MII and RMII Transmission in 10 Mbit/s Mode	14-87
Figure 14-24	End of MII and RMII Transmission in 10 Mbit/s Mode	14-87
Figure 14-25	Receive Bit Ordering	14-88
Figure 14-26	Networked Time Synchronization	14-89
Figure 14-27	Propagation Delay Calculation in Clocks Supporting Peer-to-Peer Path Correction 14-93	
Figure 14-28	System Time Update Using Fine Method	14-102
Figure 14-29	ETH Core Service Request Structure	14-105
Figure 14-30	ETH Register memory Map	14-108
Figure 14-31	Ethernet MII - PHY Interconnect	14-332
Figure 14-32	Ethernet RMII - PHY Interconnect	14-334
Figure 15-1	EtherCAT Slave Controller Block Diagram	15-2
Figure 15-2	II Interface Signals	15-4
Figure 15-3	PHY management Interface Signals	15-5
Figure 15-4	EtherCAT State Machine	15-8
Figure 15-5	EtherCAT Slave Controller Address Space Overview	15-9
Figure 15-6	ESI EEPROM Layout	15-10
Figure 15-7	EtherCAT Slave Controller Sync/Latch Signals	15-12
Figure 15-8	EtherCAT Interrupt Masking and Interrupt Signals	15-15
Figure 15-9	EtherCAT Module Initialization	15-17
Figure 15-10	EtherCAT Slave Controller PHY Interconnects	15-136
Figure 16-1	USB Module Block Diagram	16-2
Figure 16-2	OTG DRD Connections	16-3
Figure 16-3	USB Host Connections	16-4
Figure 16-4	USB Device Connections	16-5

List of Figures

Figure 16-5	Transmit Transaction-Level Operation in Slave Mode	16-9
Figure 16-6	Receive Transaction-Level Operation in Slave Mode	16-10
Figure 16-7	Functionality when HFIRRIIdCtrl = 0 _B	16-17
Figure 16-8	Functionality when HFIRRIIdCtrl = 1 _B	16-18
Figure 16-9	Transmit FIFO Write Task in Slave Mode	16-21
Figure 16-10	Receive FIFO Read Task in Slave Mode	16-22
Figure 16-11	Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in Slave Mode 16-26	
Figure 16-12	Normal Interrupt OUT/IN Transactions in Slave Mode	16-32
Figure 16-13	Normal Isochronous OUT/IN Transactions in Slave Mode	16-37
Figure 16-14	Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in DMA Mode 16-42	
Figure 16-15	Interrupt Service Routine for Bulk/Control OUT Transaction in DMA Mode 16-43	
Figure 16-16	Normal Interrupt OUT/IN Transactions in DMA Mode	16-48
Figure 16-17	Normal Isochronous OUT/IN Transactions in DMA Mode	16-53
Figure 16-18	Descriptor Memory Structures	16-56
Figure 16-19	Memory Structure	16-57
Figure 16-20	Frame List for Periodic Channels	16-58
Figure 16-21	Full Speed Isochronous Transfer Scheduling	16-69
Figure 16-22	Two-Stage Control Transfer	16-91
Figure 16-23	Receive FIFO Packet Read in Slave Mode	16-93
Figure 16-24	Processing a SETUP Packet	16-97
Figure 16-25	Slave Mode Bulk IN Transaction	16-100
Figure 16-26	Slave Mode Bulk IN Transfer (Pipelined Transaction	16-102
Figure 16-27	Slave Mode Bulk IN Two-Endpoint Transfer	16-104
Figure 16-28	Slave Mode Bulk OUT Transaction	16-107
Figure 16-29	ISOC OUT Application Flow for Periodic Transfer Interrupt Feature	16-112
Figure 16-30	Isochronous OUT Core Internal Flow for Periodic Transfer Interrupt Feature 16-114	
Figure 16-31	Periodic IN Application Flow for Periodic Transfer Interrupt Feature	16-122
Figure 16-32	Periodic IN Core Internal Flow for Periodic Transfer Interrupt Feature	16-125
Figure 16-33	Processing a SETUP Packet	16-132
Figure 16-34	Periodic IN Application Flow for Periodic Transfer Interrupt Feature	16-143
Figure 16-35	Periodic IN Core Internal Flow for Periodic Transfer Interrupt Feature	16-146
Figure 16-36	Descriptor Memory Structures	16-149
Figure 16-37	Out Data Memory Structure	16-150
Figure 16-38	IN Data Memory Structure	16-157

List of Figures

Figure 16-39	Descriptor Lists for Handling Control Transfers	16-166
Figure 16-40	Three-Stage Control Write	16-173
Figure 16-41	Three-Stage Control Read	16-174
Figure 16-42	Two-Stage Control Transfer.	16-176
Figure 16-43	Back-to-Back SETUP Packet Handling During Control Write . . .	16-179
Figure 16-44	Back-to-Back SETUP During Control Read.	16-181
Figure 16-45	Extra Tokens During Control Write Data Phase	16-183
Figure 16-46	Extra IN Tokens During Control Read Data Phase	16-185
Figure 16-47	Premature SETUP During Control Write Data Phase	16-188
Figure 16-48	Premature SETUP During Control Read Data Phase	16-190
Figure 16-49	Premature Status Phase During Control Write	16-192
Figure 16-50	Premature Status Phase During Control Read	16-194
Figure 16-51	Lost ACK During Last Packet of Control Read	16-195
Figure 16-52	IN Descriptor List	16-197
Figure 16-53	Non ISO IN Descriptor/Data Processing	16-199
Figure 16-54	Bulk IN Transfers	16-200
Figure 16-55	OUT Descriptor List	16-202
Figure 16-56	Non ISO OUT Descriptor/Data Buffer Processing	16-205
Figure 16-57	Bulk OUT Transfers	16-206
Figure 16-58	ISO IN Data Flow	16-208
Figure 16-59	ISO IN Descriptor/Data Processing	16-210
Figure 16-60	Isochronous IN Transfers.	16-211
Figure 16-61	Isochronous OUT Descriptor/Data Buffer Processing	16-213
Figure 16-62	ISO Out Data Flow	16-214
Figure 16-63	A-Device SRP	16-215
Figure 16-64	B-Device SRP	16-216
Figure 16-65	A-Device HNP	16-217
Figure 16-66	B-Device HNP	16-218
Figure 16-67	Host Mode Suspend and Resume With Clock Gating	16-220
Figure 16-68	Host Mode Suspend and Remote Wakeup With Clock Gating . .	16-221
Figure 16-69	Device Mode FIFO Address Mapping	16-227
Figure 16-70	Host Mode FIFO Address Mapping	16-230
Figure 16-71	Interrupt Hierarchy	16-232
Figure 16-72	Core Interrupt Handler	16-233
Figure 16-73	CSR Memory Map	16-236
Figure 17-1	USIC Module/Channel Structure	17-5
Figure 17-2	Baud Rate Generator	17-10
Figure 17-3	Principle of Data Buffering	17-11
Figure 17-4	Data Access Structure without additional Data Buffer	17-12
Figure 17-5	Data Access Structure with FIFO.	17-14
Figure 17-6	General Event and Interrupt Structure	17-17
Figure 17-7	Transmit Events and Interrupts	17-19
Figure 17-8	Receive Events and Interrupts	17-20

List of Figures

Figure 17-9	Baud Rate Generator Event and Interrupt	17-21
Figure 17-10	Input Conditioning for DX0 and DX[5:3]	17-23
Figure 17-11	Input Conditioning for DX[2:1]	17-24
Figure 17-12	Delay Compensation Enable in DX1	17-25
Figure 17-13	Divider Mode Counter	17-28
Figure 17-14	Protocol-Related Counter (Capture Mode)	17-29
Figure 17-15	Time Quanta Counter	17-29
Figure 17-16	Master Clock Output Configuration	17-30
Figure 17-17	Transmit Data Path	17-32
Figure 17-18	Transmit Data Validation	17-35
Figure 17-19	Receive Data Path	17-37
Figure 17-20	FIFO Buffer Overview	17-39
Figure 17-21	FIFO Buffer Partitioning	17-41
Figure 17-22	Standard Transmit Buffer Event Examples	17-43
Figure 17-23	Transmit Buffer Events	17-44
Figure 17-24	Standard Receive Buffer Event Examples	17-47
Figure 17-25	Receiver Buffer Events in Filling Level Mode	17-48
Figure 17-26	Receiver Buffer Events in RCI Mode	17-49
Figure 17-27	Bypass Data Validation	17-51
Figure 17-28	TCI Handling with FIFO / Bypass	17-53
Figure 17-29	ASC Signal Connections for Full-Duplex Communication	17-54
Figure 17-30	ASC Signal Connections for Half-Duplex Communication	17-55
Figure 17-31	Standard ASC Frame Format	17-56
Figure 17-32	ASC Bit Timing	17-59
Figure 17-33	Transmitter Pulse Length Control	17-61
Figure 17-34	Pulse Output Example	17-62
Figure 17-35	SSC Signals for Standard Full-Duplex Communication	17-74
Figure 17-36	4-Wire SSC Standard Communication Signals	17-76
Figure 17-37	SSC Data Signals	17-76
Figure 17-38	SSC Shift Clock Signals	17-77
Figure 17-39	SCLKOUT Configuration in SSC Master Mode	17-79
Figure 17-40	SLPHSEL Configuration in SSC Slave Mode	17-80
Figure 17-41	SSC Slave Select Signals	17-81
Figure 17-42	Data Frames without/with Parity	17-84
Figure 17-43	Quad-SSC Example	17-87
Figure 17-44	Connections for Quad-SSC Example	17-88
Figure 17-45	MSLS Generation in SSC Master Mode	17-90
Figure 17-46	SSC Closed-loop Delay	17-104
Figure 17-47	SSC Closed-loop Delay Timing Waveform	17-106
Figure 17-48	SSC Master Mode with Delay Compensation	17-107
Figure 17-49	SSC Complete Closed-loop Delay Compensation	17-108
Figure 17-50	IIC Signal Connections	17-110
Figure 17-51	IIC Frame Example (simplified)	17-111

List of Figures

Figure 17-52	Start Symbol Timing	17-119
Figure 17-53	Repeated Start Symbol Timing	17-119
Figure 17-54	Stop Symbol Timing	17-120
Figure 17-55	Data Bit Symbol	17-120
Figure 17-56	IIC Master Transmission	17-125
Figure 17-57	Interrupt Events on Data Transfers	17-128
Figure 17-58	IIS Signals	17-136
Figure 17-59	Protocol Overview	17-137
Figure 17-60	Transfer Delay for IIS	17-137
Figure 17-61	Connection of External Audio Devices	17-138
Figure 17-62	Transfer Delay with Delay 1	17-140
Figure 17-63	No Transfer Delay	17-141
Figure 17-64	MCLK and SCLK for IIS	17-145
Figure 17-65	USIC Module and Channel Registers	17-154
Figure 17-66	USIC Module Structure in XMC4300	17-226
Figure 17-67	USIC Channel I/O Lines	17-227
Figure 18-1	Classical 11bit ID CAN Data Frame	18-4
Figure 18-2	29 bit ID CAN Data Frame	18-5
Figure 18-3	CAN Error Frames	18-6
Figure 18-4	Partition of Nominal Bit Time	18-7
Figure 18-5	Overview of the MultiCAN+ Module. The module has <u>2</u> nodes and <u>64</u> objects. XMC4300 18-10	
Figure 18-6	MultiCAN+ Block Diagram	18-13
Figure 18-7	General Interrupt Structure	18-15
Figure 18-8	MultiCAN+ Clock Generation	18-16
Figure 18-9	CAN Bus Bit Timing Standard	18-20
Figure 18-10	CAN Node Interrupts	18-24
Figure 18-11	Example Allocation of Message Objects to a List	18-25
Figure 18-12	Message Objects Linked to CAN Nodes	18-27
Figure 18-13	Loop-Back Mode	18-31
Figure 18-14	Received Message Identifier Acceptance Check	18-33
Figure 18-15	Effective Transmit Request of Message Object	18-35
Figure 18-16	Message Interrupt Request Routing	18-36
Figure 18-17	Message Pending Bit Allocation	18-37
Figure 18-18	Reception of a Message Object	18-41
Figure 18-19	Transmission of a Message Object	18-44
Figure 18-20	FIFO Structure with FIFO Base Object and n FIFO Slave Objects	18-47
Figure 18-21	Gateway Transfer from Source to Destination	18-51
Figure 18-22	MultiCAN+ Kernel Registers	18-57
Figure 18-23	MultiCAN+ Register Address Map XMC	18-61
Figure 18-24	MultiCAN+ Module Implementation and Interconnections with n := <u>1</u> and m := <u>64</u> for XMC4300 18-113	
Figure 18-25	CAN Implementation-specific Special Function Registers	18-114

List of Figures

Figure 18-26	MultiCAN+ Module Clock Generation	18-115
Figure 18-27	CAN Module Receive Input Selection	18-120
Figure 18-28	Interrupt Compressor $n = 2$, $m = 64$ and $k = 8$	18-123
Figure 18-29	MultiCAN+ Register Address Map XMC	18-124
Figure 19-1	ADC Structure Overview	19-3
Figure 19-2	ADC Kernel Block Diagram	19-4
Figure 19-3	Conversion Request Unit	19-6
Figure 19-4	Clock Signal Summary	19-9
Figure 19-5	Queued Request Source	19-13
Figure 19-6	Interrupt Generation of a Queued Request Source	19-16
Figure 19-7	Scan Request Source	19-17
Figure 19-8	Arbitration Round with 4 Arbitration Slots	19-20
Figure 19-9	Conversion Start Modes	19-23
Figure 19-10	Alias Feature	19-27
Figure 19-11	Result Monitoring through Limit Checking	19-29
Figure 19-12	Result Monitoring through Compare with Hysteresis.	19-31
Figure 19-13	Boundary Flag Switching	19-32
Figure 19-14	Boundary Flag Control	19-33
Figure 19-15	Conversion Result Storage	19-35
Figure 19-16	Result Storage Options	19-36
Figure 19-17	Result FIFO Buffers	19-38
Figure 19-18	Standard Data Reduction Filter	19-42
Figure 19-19	Standard Data Reduction Filter with FIFO Enabled.	19-43
Figure 19-20	FIR Filter Structure.	19-44
Figure 19-21	IIR Filter Structure	19-45
Figure 19-22	Result Difference	19-46
Figure 19-23	Parallel Conversions	19-47
Figure 19-24	Synchronization via ANON and Ready Signals	19-49
Figure 19-25	Timer Mode for Equidistant Sampling	19-50
Figure 19-26	Broken Wire Detection	19-51
Figure 19-27	Signal Path Test.	19-53
Figure 19-28	External Analog Multiplexer Example	19-54
Figure 20-1	Block Diagram of DAC Module including Digidac Submodule	20-2
Figure 20-2	Trigger Generator Block Diagram	20-4
Figure 20-3	Data FIFO Block Diagram	20-5
Figure 20-4	Data Output Stage Block Diagram	20-5
Figure 20-5	Pattern Generator Block Diagram	20-6
Figure 20-6	Noise Generator Block Diagram	20-7
Figure 20-7	Ramp Generator Block Diagram	20-8
Figure 20-8	Data Handling for the FIFO Buffer	20-10
Figure 20-9	Example 5-bit Patterns and their corresponding Waveform Output	20-11
Figure 20-10	Signed 12-bit pseudo random Noise Example Output.	20-12
Figure 20-11	Unsigned 12-bit Ramp Generator Example Output	20-12

List of Figures

Figure 21-1	CCU4 block diagram	21-5
Figure 21-2	Register description in figures (example).	21-6
Figure 21-3	CCU4 slice block diagram	21-7
Figure 21-4	Slice input selector diagram.	21-9
Figure 21-5	Slice connection matrix diagram	21-11
Figure 21-6	Timer start/stop control diagram	21-12
Figure 21-7	Starting multiple timers synchronously	21-13
Figure 21-8	CC4y Status Bit	21-14
Figure 21-9	Shadow registers overview	21-16
Figure 21-10	Shadow transfer enable logic.	21-17
Figure 21-11	Shadow transfer timing example - center aligned mode	21-18
Figure 21-12	Usage of the CCU4x.MCSS input	21-19
Figure 21-13	Edge aligned mode, CC4yTC.TCM = 0 _B	21-20
Figure 21-14	Center aligned mode, CC4yTC.TCM = 1 _B	21-21
Figure 21-15	Single shot edge aligned - CC4yTC.TSSM = 1 _B , CC4yTC.TCM = 0 _B	21-21
Figure 21-16	Single shot center aligned - CC4yTC.TSSM = 1 _B , CC4yTC.TCM = 1 _B	21-22
Figure 21-17	Start (as start)/ stop (as stop) - CC4yTC.STRM = 0 _B , CC4yTC.ENDM = 00 _B	21-24
Figure 21-18	Start (as start)/ stop (as flush) - CC4yTC.STRM = 0 _B , CC4yTC.ENDM = 01 _B	21-24
Figure 21-19	Start (as flush and start)/ stop (as stop) - CC4yTC.STRM = 1 _B , CC4yTC.ENDM = 00 _B	21-25
Figure 21-20	Start (as start)/ stop (as flush and stop) - CC4yTC.STRM = 0 _B , CC4yTC.ENDM = 10 _B	21-25
Figure 21-21	External counting direction.	21-26
Figure 21-22	External gating	21-27
Figure 21-23	External count	21-28
Figure 21-24	External load	21-29
Figure 21-25	External capture - CC4yCMC.CAP0S != 00 _B , CC4yCMC.CAP1S = 00 _B	21-31
Figure 21-26	External capture - CC4yCMC.CAP0S != 00 _B , CC4yCMC.CAP1S != 00 _B	21-32
Figure 21-27	Slice capture logic	21-33
Figure 21-28	External Capture - CC4yTC.SCE = 1 _B	21-34
Figure 21-29	Slice Capture Logic - CC4yTC.SCE = 1 _B	21-34
Figure 21-30	Capture Extended Read Back - Depth 4	21-36
Figure 21-31	Depth 4 software access example	21-37
Figure 21-32	Capture Extended Read Back - Depth 2	21-38
Figure 21-33	Depth 2 software access example	21-38
Figure 21-34	External modulation clearing the ST bit - CC4yTC.EMT = 0 _B	21-39
Figure 21-35	External modulation clearing the ST bit - CC4yTC.EMT = 0 _B	

List of Figures

CC4yTC.EMS = 1_B 21-40

Figure 21-36 External modulation gating the output - **CC4yTC.EMT = 1_B** 21-40

Figure 21-37 Trap control diagram 21-41

Figure 21-38 Trap timing diagram, **CC4yPSL.PSL = 0_B** (output passive level is 0_B) 21-42

Figure 21-39 Trap synchronization with the PWM signal, **CC4yTC.TRPSE = 1_B** 21-43

Figure 21-40 Status bit override 21-44

Figure 21-41 Multi channel pattern synchronization 21-45

Figure 21-42 Multi Channel mode for multiple Timer Slices 21-45

Figure 21-43 CC4y Status bit and Output Path 21-46

Figure 21-44 Multi Channel Mode Control Logic 21-47

Figure 21-45 Timer Concatenation Example 21-48

Figure 21-46 Timer Concatenation Link 21-49

Figure 21-47 Capture/Load Timer Concatenation 21-50

Figure 21-48 32 bit concatenation timing diagram 21-51

Figure 21-49 Timer concatenation control logic 21-52

Figure 21-50 Dither structure overview 21-53

Figure 21-51 Dither control logic 21-55

Figure 21-52 Dither timing diagram in edge aligned - **CC4yTC.DITHE = 01_B** 21-55

Figure 21-53 Dither timing diagram in edge aligned - **CC4yTC.DITHE = 10_B** 21-56

Figure 21-54 Dither timing diagram in edge aligned - **CC4yTC.DITHE = 11_B** 21-56

Figure 21-55 Dither timing diagram in center aligned - **CC4yTC.DITHE = 01_B** 21-56

Figure 21-56 Dither timing diagram in center aligned - **CC4yTC.DITHE = 10_B** 21-57

Figure 21-57 Dither timing diagram in center aligned - **CC4yTC.DITHE = 11_B** 21-57

Figure 21-58 Floating prescaler in compare mode overview 21-59

Figure 21-59 Floating Prescaler in capture mode overview 21-60

Figure 21-60 PWM with 100% duty cycle - Edge Aligned Mode 21-61

Figure 21-61 PWM with 100% duty cycle - Center Aligned Mode 21-61

Figure 21-62 PWM with 0% duty cycle - Edge Aligned Mode 21-62

Figure 21-63 PWM with 0% duty cycle - Center Aligned Mode 21-62

Figure 21-64 Floating Prescaler capture mode usage 21-63

Figure 21-65 Floating Prescaler compare mode usage - Edge Aligned 21-64

Figure 21-66 Floating Prescaler compare mode usage - Center Aligned 21-64

Figure 21-67 Capture mode usage - single channel 21-68

Figure 21-68 Three Capture profiles - **CC4yTC.SCE = 1_B** 21-69

Figure 21-69 High dynamics capturing with software controlled timestamp 21-70

Figure 21-70 Extended read back during high load 21-71

Figure 21-71 Capture grouping with extended read back 21-71

Figure 21-72 Memory structure for extended read back 21-73

Figure 21-73 Slice interrupt structure overview 21-75

Figure 21-74 Slice Interrupt Node Pointer overview 21-76

Figure 21-75 CCU4 service request overview 21-77

Figure 21-76 CCU4 registers overview 21-81

List of Figures

Figure 22-1	CCU8 block diagram	22-6
Figure 22-2	Register description in figures (example).	22-7
Figure 22-3	CCU8 slice block diagram	22-8
Figure 22-4	Slice input selector diagram.	22-11
Figure 22-5	Slice connection matrix diagram	22-13
Figure 22-6	Timer start/stop control diagram	22-14
Figure 22-7	Start multiple timers synchronously.	22-15
Figure 22-8	CC8y Status Bits	22-16
Figure 22-9	Shadow registers overview	22-18
Figure 22-10	Shadow transfer enable logic.	22-19
Figure 22-11	Shadow transfer timing example - center aligned mode	22-20
Figure 22-12	Usage of the CCU8x.MCSS input	22-21
Figure 22-13	Cascaded shadow transfer linking.	22-22
Figure 22-14	Cascade shadow transfer timing	22-23
Figure 22-15	Edge aligned mode, CC8yTC.TCM = 0_B	22-24
Figure 22-16	Center aligned mode, CC8yTC.TCM = 1_B	22-25
Figure 22-17	Single shot edge aligned - CC8yTC.TSSM = 1_B, CC8yTC.TCM = 0_B	22-26
Figure 22-18	Single shot center aligned - CC8yTC.TSSM = 1_B, CC8yTC.TCM = 1_B	22-26
Figure 22-19	Compare channels diagram.	22-27
Figure 22-20	Dead Time scheme	22-28
Figure 22-21	Dead Time generator scheme	22-29
Figure 22-22	Dead Time control cell	22-30
Figure 22-23	Dead Time trigger with the Multi Channel pattern	22-31
Figure 22-24	Edge Aligned with two independent channels scheme	22-31
Figure 22-25	Edge Aligned - four outputs with dead time	22-32
Figure 22-26	Edge Aligned with combined channels scheme.	22-33
Figure 22-27	Edge Aligned - Asymmetric PWM timing, CC8yCR1.CR1 < CC8yCR2.CR2 22-34	
Figure 22-28	Edge Aligned - Asymmetric PWM timing, CC8yCR1.CR1 > CC8yCR2.CR2 22-35	
Figure 22-29	Center Aligned with two independent channels scheme	22-36
Figure 22-30	Center aligned - Independent channel with dead time.	22-36
Figure 22-31	Center Aligned Asymmetric mode scheme	22-37
Figure 22-32	Asymmetric Center aligned mode with dead time	22-38
Figure 22-33	Start (as start)/ stop (as stop) - CC8yTC.STRM = 0_B, CC8yTC.ENDM = 00_B 22-39	
Figure 22-34	Start (as start)/ stop (as flush) - CC8yTC.STRM = 0_B, CC8yTC.ENDM = 01_B 22-40	
Figure 22-35	Start (as flush and start)/ stop (as stop) - CC8yTC.STRM = 1_B, CC8yTC.ENDM = 00_B 22-40	
Figure 22-36	Start (as start)/ stop (as flush and stop) - CC8yTC.STRM = 0_B,	

List of Figures

	CC8yTC.ENDM = 10_B 22-41	
Figure 22-37	External counting direction.	22-42
Figure 22-38	External gating.	22-43
Figure 22-39	External count.	22-44
Figure 22-40	Timer load selection.	22-44
Figure 22-41	External load.	22-45
Figure 22-42	External capture - CC8yCMC.CAPOS != 00_B , CC8yCMC.CAP1S = 00_B	22-47
Figure 22-43	External capture - CC8yCMC.CAPOS != 00_B , CC8yCMC.CAP1S != 00_B	22-48
Figure 22-44	Slice capture logic.	22-49
Figure 22-45	External Capture - CC8yTC.SCE = 1_B	22-50
Figure 22-46	Slice Capture Logic - CC8yTC.SCE = 1_B	22-50
Figure 22-47	Capture Extended Read Back - Depth 4.	22-52
Figure 22-48	Depth 4 software access example.	22-53
Figure 22-49	Capture Extended Read Back - Depth 2.	22-54
Figure 22-50	Depth 2 software access example.	22-54
Figure 22-51	External modulation resets the ST bit - CC8yTC.EMS = 0_B	22-55
Figure 22-52	External modulation clearing the ST bit - CC8yTC.EMS = 1_B	22-56
Figure 22-53	External modulation gating the output - CC8yTC.EMT = 1_B	22-56
Figure 22-54	Trap control diagram.	22-57
Figure 22-55	Trap timing diagram, CC8yTCST.CDIR = 0 CC8yPSL.PSL = 0	22-58
Figure 22-56	Trap synchronization with the PWM signal.	22-59
Figure 22-57	Status bit override.	22-60
Figure 22-58	Multi channel pattern synchronization.	22-61
Figure 22-59	CCU8 Multi Channel overview.	22-62
Figure 22-60	Multi Channel mode for multiple Timer Slices.	22-63
Figure 22-61	Output Control Diagram.	22-64
Figure 22-62	Multi Channel Pattern Synchronization Control.	22-65
Figure 22-63	Timer concatenation example.	22-66
Figure 22-64	Timer concatenation link.	22-67
Figure 22-65	Capture/Load Timer Concatenation.	22-68
Figure 22-66	32 bit concatenation timing diagram.	22-69
Figure 22-67	Timer concatenation control logic.	22-70
Figure 22-68	Parity checker structure.	22-71
Figure 22-69	Parity checker logic.	22-73
Figure 22-70	Dither structure overview.	22-74
Figure 22-71	Dither control logic.	22-76
Figure 22-72	Dither timing diagram in edge aligned - CC8yTC.DITHE = 01_B	22-76
Figure 22-73	Dither timing diagram in edge aligned - CC8yTC.DITHE = 10_B	22-77
Figure 22-74	Dither timing diagram in edge aligned - CC8yTC.DITHE = 11_B	22-77
Figure 22-75	Dither timing diagram in center aligned - CC8yTC.DITHE = 01_B	22-77
Figure 22-76	Dither timing diagram in center aligned - CC8yTC.DITHE = 10_B	22-78

List of Figures

Figure 22-77	Dither timing diagram in edge aligned - CC8yTC .DITHE = 11 _B . . .	22-78
Figure 22-78	Floating prescaler in compare mode overview	22-80
Figure 22-79	Floating Prescaler in capture mode overview	22-81
Figure 22-80	PWM with 100% duty cycle - Edge Aligned Mode	22-82
Figure 22-81	PWM with 100% duty cycle - Center Aligned Mode.	22-82
Figure 22-82	PWM with 0% duty cycle - Edge Aligned Mode	22-83
Figure 22-83	PWM with 0% duty cycle - Center Aligned Mode.	22-83
Figure 22-84	Floating Prescaler capture mode usage	22-84
Figure 22-85	Floating Prescaler compare mode usage - Edge Aligned	22-85
Figure 22-86	Floating Prescaler compare mode usage - Center Aligned	22-85
Figure 22-87	Capture mode usage - single channel	22-89
Figure 22-88	Three Capture profiles - CC8yTC .SCE = 1 _B	22-90
Figure 22-89	High dynamics capturing with software controlled timestamp	22-91
Figure 22-90	Extended read back during high load	22-92
Figure 22-91	Capture grouping with extended read back	22-92
Figure 22-92	Memory structure for extended read back	22-94
Figure 22-93	Parity Checker connections	22-96
Figure 22-94	Parity Checker timing example	22-97
Figure 22-95	Slice interrupt node pointer overview	22-99
Figure 22-96	Slice interrupt selector overview	22-99
Figure 22-97	CCU8 service request overview.	22-100
Figure 22-98	CCU8 registers overview	22-105
Figure 23-1	General Structure of a digital Port Pin	23-3
Figure 23-2	Port Pin in Power Save State.	23-8
Figure 23-3	Analog Port Structure.	23-9
Figure 23-4	Simplified Port Structure	23-35
Figure 24-1	DSRAM1 usage by SSW	24-4
Figure 24-2	Reading Bootcode	24-6
Figure 24-3	Boot mode identification.	24-7
Figure 24-4	Device Information Data	24-8
Figure 24-5	PSRAM header layout	24-10
Figure 24-6	PSRAM layout for PSRAM boot.	24-10
Figure 24-7	ABM concept	24-12
Figure 24-8	ASC BSL mode procedures.	24-14
Figure 24-9	Application download protocol	24-15
Figure 24-10	Application download protocol FAIL.	24-15
Figure 24-11	CAN BSL procedures.	24-17
Figure 24-12	Data field of CAN BSL Initialization frame	24-18
Figure 24-13	CAN Acknowledgement frame	24-19
Figure 24-14	BMI String layout	24-20
Figure 24-15	BMI Actions	24-22
Figure 24-16	Diagnostics monitor mode	24-25
Figure 25-1	Debug and Trace System block diagram.	25-3

List of Figures

Figure 25-2	HAR - Halt After Reset	25-8
Figure 25-3	HOT PLUG or Warm Reset	25-9
Figure 25-4	Debug and Trace connector	25-16

List of Tables

Table 1	Bit Function Terminology	P-3
Table 2	Register Access Modes	P-3
Table 2-1	Summary of processor mode, execution privilege level, and stack use options 2-5	
Table 2-2	Core register set summary	2-6
Table 2-3	PSR register combinations	2-9
Table 2-4	CMSIS functions to generate some Cortex-M4 instructions	2-18
Table 2-5	CMSIS functions to access the special registers	2-19
Table 2-6	Memory access behavior	2-22
Table 2-7	CMSIS functions for exclusive access instructions	2-26
Table 2-8	Properties of the different exception types	2-28
Table 2-9	Exception return behavior	2-36
Table 2-10	Faults	2-37
Table 2-11	Fault status and fault address registers	2-39
Table 2-12	Core peripheral register regions	2-42
Table 2-13	CMSIS functions for NVIC control	2-45
Table 2-14	Memory attributes summary	2-47
Table 2-15	TEX, C, B, and S encoding	2-48
Table 2-16	Cache policy for memory attribute encoding	2-49
Table 2-17	Memory region attributes for a microcontroller	2-49
Table 2-18	AP encoding	2-49
Table 2-19	Registers Overview	2-54
Table 2-20	Priority grouping	2-66
Table 2-21	System fault handler priority fields	2-70
Table 2-22	Example SIZE field values	2-100
Table 3-1	Access Priorities per Slave	3-3
Table 4-1	Abbreviations	4-1
Table 4-2	Interrupt and DMA services per Module	4-4
Table 4-3	Interrupt Node assignment	4-5
Table 4-4	DMA Handler Service Request inputs	4-9
Table 4-5	DMA Request Source Selection	4-10
Table 4-6	Registers Address Space	4-23
Table 4-7	4-23
Table 4-8	ERU0 Pin Connections	4-33
Table 4-9	ERU1 Pin Connections	4-36
Table 5-1	Abbreviations table	5-1
Table 5-2	Transfer Type, Flow Control and Handshake Combinations	5-9
Table 5-3	Parameters Used in Transfer Examples	5-21
Table 5-4	Parameters in Transfer Operation	5-22
Table 5-5	Programming of Transfer Types and Channel Register Update Method .	5-29

List of Tables

Table 5-6	Registers Address Space	5-59
Table 5-7	Register Overview	5-60
Table 5-8	CTLL.SRC_MSIZ and CTLL.DST_MSIZ Field Decoding	5-76
Table 5-9	CTLL.SRC_TR_WIDTH and CTLL.DST_TR_WIDTH Field Decoding ...	5-76
Table 5-10	CTLL.TT_FC Field Decoding	5-76
Table 5-11	PROTCTL field to HPROT Mapping	5-93
Table 6-1	FCE Abbreviations	6-1
Table 6-2	Registers Address Space - FCE Module	6-11
Table 6-3	Registers Overview - CRC Kernel Registers	6-11
Table 6-4	FCE Service Requests	6-24
Table 6-5	Hamming Distance as a function of message length (bits)	6-25
Table 7-1	Memory Regions	7-3
Table 7-2	Memory Map	7-4
Table 7-3	Parity Test Enabled Memories and Supported Parity Error Indication ...	7-8
Table 7-4	Registers Address Space	7-10
Table 7-5	Registers Overview	7-10
Table 8-1	Flash Memory Map	8-7
Table 8-2	Sector Structure of PFLASH	8-7
Table 8-3	Structure of UCB Area	8-8
Table 8-4	Command Sequences for Flash Control	8-11
Table 8-5	UCB Content	8-17
Table 8-6	Registers Address Space	8-33
Table 8-7	Registers Overview	8-33
Table 8-8	Registers Address Space	8-35
Table 8-9	Registers Overview	8-35
Table 8-10	Registers Address Space	8-37
Table 8-11	Addresses of Flash0 Registers	8-37
Table 9-1	Application Features	9-2
Table 9-2	Registers Address Space	9-11
Table 9-3	Register Overview	9-11
Table 9-4	Pin Table	9-16
Table 10-1	Application Features	10-1
Table 10-2	Registers Address Space	10-8
Table 10-3	Register Overview	10-8
Table 10-4	Pin Connections	10-20
Table 11-1	Service Requests	11-7
Table 11-2	SCU Trap Request Overview	11-10
Table 11-3	Reset Overview	11-29
Table 11-4	Clock Signals	11-32
Table 11-5	Valid values of clock divide registers for f_{CCU} , f_{CPU} and f_{PERIPH} clocks ...	11-35

List of Tables

Table 11-6	PLL example configuration values	11-41
Table 11-7	PLL example configuration values	11-48
Table 11-8	PLLUSB example configuration values	11-49
Table 11-9	Base Addresses of sub-sections of SCU registers	11-62
Table 11-10	Registers Address Space	11-62
Table 11-11	Registers Overview	11-62
Table 11-12	Memory Parity Bus Widths	11-103
Table 12-1	Abbreviations in chapter	12-1
Table 12-2	LEDTS Applications	12-2
Table 12-3	LEDTS Interrupt Events	12-14
Table 12-4	LEDTS Events' Interrupt Node Control	12-15
Table 12-5	Interpretation of FNCOL Bit Field.	12-17
Table 12-6	LEDTS Pin Control Signals	12-19
Table 12-7	Registers Address Space	12-22
Table 12-8	Register Overview of LEDTS.	12-22
Table 12-9	Pin Connection	12-36
Table 13-1	SDMMC Applications	13-2
Table 13-2	Registers Address Space	13-17
Table 13-3	Register Overview	13-17
Table 13-4	Determination of transfer type	13-26
Table 13-5	Relation between parameters and the name of response type ..	13-30
Table 13-6	Response bit definition for each response type	13-30
Table 13-7	Relation between transfer complete and data timeout error	13-62
Table 13-8	Relation between command complete and command timeout error	13-62
Table 13-9	Relation between command CRC error and command time-out error ...	13-67
Table 13-10	Relation between Auto CMD12 CRC error and Auto CMD12 timeout error	13-78
Table 13-11	Maximum current value definition	13-83
Table 13-12	SDMMC Pin Connections	13-90
Table 14-1	Abbreviations	14-1
Table 14-2	Destination Address Filtering Table	14-17
Table 14-3	Source Address Filtering Table	14-18
Table 14-4	Receive Descriptor 0	14-42
Table 14-5	Receive Descriptor 0 When COE (Type 2) Is Enabled	14-45
Table 14-6	Receive Descriptor 1	14-46
Table 14-7	Receive Descriptor 2 (Default Operation)	14-47
Table 14-8	Receive Descriptor 3	14-47
Table 14-9	Transmit Descriptor 0	14-48
Table 14-10	Transmit Descriptor 1	14-51
Table 14-11	Transmit Descriptor 2	14-52
Table 14-12	Transmit Descriptor 3	14-53

List of Tables

Table 14-13	Receive Descriptor Fields (RDES2)	14-55
Table 14-14	Receive Descriptor Fields (RDES3)	14-55
Table 14-15	Transmit Time Stamp Status – Normal Descriptor Format Case (TDES0RAM) 14-56	
Table 14-16	Transmit Time Stamp Control – Normal Descriptor Format Case (TDES1RAM) 14-57	
Table 14-17	Transmit Descriptor Fields (TDES2RAM)	14-57
Table 14-18	Transmit Descriptor Fields (TDES3)	14-57
Table 14-19	Transmit Descriptor Word 0 (TDES0)	14-60
Table 14-20	Transmit Descriptor Word 1 (TDES1)	14-63
Table 14-21	Transmit Descriptor 2 (TDES2)	14-64
Table 14-22	Transmit Descriptor 3 (TDES3)	14-64
Table 14-23	Transmit Descriptor 6 (TDES6)	14-64
Table 14-24	Transmit Descriptor 7 (TDES7)	14-65
Table 14-25	Receive Descriptor Fields (RDES0)	14-67
Table 14-26	Receive Descriptor Fields 1 (RDES1)	14-69
Table 14-27	Receive Descriptor Fields 2 (RDES2)	14-71
Table 14-28	Receive Descriptor Fields 3 (RDES3)	14-71
Table 14-29	Receive Descriptor Fields 4 (RDES4)	14-72
Table 14-30	Receive Descriptor Fields 6 (RDES6)	14-73
Table 14-31	Receive Descriptor Fields 7 (RDES7)	14-74
Table 14-32	PTP Messages for which Snapshot is Taken on Receive Side for Ordinary Clock 14-95	
Table 14-33	PTP Messages for which Snapshot is Taken for Transparent Clock Implementation 14-95	
Table 14-34	PTP Messages for which Snapshot is Taken for Peer-to-Peer Transparent Clock Implementation 14-95	
Table 14-35	Message Format Defined in IEEE 1588-2008	14-96
Table 14-36	IPv4-UDP PTP Frame Fields Required for Control and Status ...	14-96
Table 14-37	IPv6-UDP PTP Frame Fields Required for Control and Status ...	14-98
Table 14-38	Ethernet PTP Frame Fields Required for Control And Status ...	14-99
Table 14-39	Registers Address Space - ETH Module	14-108
Table 14-40	ETH Registers Overview	14-109
Table 14-41	ETH Pin Connections for MIII	14-332
Table 14-42	ETH Pin Connections for RMIII	14-335
Table 14-43	ETH Pin Connections for MDIO	14-336
Table 14-44	ETH System Related connections	14-336
Table 15-1	Abbreviations	15-1
Table 15-2	EtherCAT EEPROM Configuration Area	15-11
Table 15-3	RUN LED States	15-13
Table 15-4	ERR LED States	15-14
Table 15-5	LINK_ACT LED States	15-14
Table 15-6	Registers Address Space	15-19

List of Tables

Table 15-7	Register Overview	15-19
Table 15-8	Decoding port state in ESC DL Status register high byte (typical modes only) 15-45	
Table 15-9	Table 63: SII EEPROM Interface Register overview	15-75
Table 15-10	MII Management Interface Register Overview	15-81
Table 15-11	Table 78: FMMU Register overview	15-87
Table 15-12	SyncManager Register overview	15-94
Table 15-13	ECAT Interconnects.	15-137
Table 16-1	Abbreviations	16-1
Table 16-2	Host Programming Operations	16-19
Table 16-3	IN Data Memory Structure Values	16-59
Table 16-4	IN Buffer Pointer	16-61
Table 16-5	OUT Data Memory Structure Values	16-63
Table 16-6	IN Buffer Pointer	16-64
Table 16-7	Asynchronous Transfer Descriptor	16-66
Table 16-8	Device Programming Operations	16-85
Table 16-9	OUT Data Memory Structure Values	16-151
Table 16-10	OUT - L Bit and MTRF Bit	16-155
Table 16-11	OUT Buffer Pointer	16-155
Table 16-12	IN Data Memory Structure Values	16-158
Table 16-13	IN - L Bit, SP Bit and Tx bytes	16-160
Table 16-14	IN - Buffer Pointer	16-161
Table 16-15	IN Buffer Pointer	16-162
Table 16-16	Combinations of OUT Endpoint Interrupts for Control Transfer	16-164
Table 16-17	RAM Space Allocation	16-224
Table 16-18	FIFO Name - Data RAM Size	16-226
Table 16-19	FIFO Name - Data RAM Size	16-229
Table 16-20	Registers Address Space	16-235
Table 16-21	Register Overview	16-237
Table 16-22	Data FIFO (DFIFO) Access Register Map	16-241
Table 16-23	Minimum Duration for Soft Disconnect	16-307
Table 16-24	Pin Connections	16-345
Table 17-1	Abbreviations	17-1
Table 17-2	Input Signals for Different Protocols	17-7
Table 17-3	Output Signals for Different Protocols	17-8
Table 17-4	USIC Communication Channel Behavior	17-16
Table 17-5	Data Transfer Events and Interrupt Handling	17-19
Table 17-6	Baud Rate Generator Event and Interrupt Handling	17-21
Table 17-7	Protocol-specific Events and Interrupt Handling	17-22
Table 17-8	Transmit Shift Register Composition	17-33
Table 17-9	Receive Shift Register Composition	17-38
Table 17-10	Transmit Buffer Events and Interrupt Handling	17-44
Table 17-11	Receive Buffer Events and Interrupt Handling	17-49

List of Tables

Table 17-12	SSC Communication Signals	17-75
Table 17-13	Master Transmit Data Formats	17-121
Table 17-14	Slave Transmit Data Format	17-122
Table 17-15	Valid TDF Codes Overview	17-123
Table 17-16	TDF Code Sequence for Master Transmit	17-126
Table 17-17	TDF Code Sequence for Master Receive (7-bit Addressing Mode)	17-126
Table 17-18	TDF Code Sequence for Master Receive (10-bit Addressing Mode)	17-127
Table 17-19	IIS IO Signals	17-135
Table 17-20	USIC Kernel-Related and Kernel Registers	17-154
Table 17-21	Registers Address Space	17-157
Table 17-22	FIFO and Reserved Address Space	17-158
Table 17-23	USIC Module 0 Channel 0 Interconnects	17-228
Table 17-24	USIC Module 0 Channel 1 Interconnects	17-231
Table 17-25	USIC Module 0 Module Interconnects	17-235
Table 17-26	USIC Module 1 Channel 0 Interconnects	17-235
Table 17-27	USIC Module 1 Channel 1 Interconnects	17-239
Table 17-28	USIC Module 1 Module Interconnects	17-242
Table 18-1	Fixed Module Constants	18-1
Table 18-2	Minimum Operating Frequencies [MHz]	18-17
Table 18-3	Minimum Operating Frequencies [MHz] required for 500kBaud	18-17
Table 18-4	Panel Commands Overview	18-28
Table 18-5	Message Transmission Bit Definitions	18-42
Table 18-6	Registers Address Space - MultiCAN+ Kernel Registers	18-57
Table 18-7	Registers Overview - MultiCAN+ Kernel Registers	18-58
Table 18-8	Panel Commands	18-64
Table 18-9	Encoding of the LEC Bit field	18-80
Table 18-10	Bit Timing Analysis Modes (CFMOD = 10)	18-90
Table 18-11	CAN Bus State Information	18-91
Table 18-12	Reset/Set Conditions for Bits in Register MOCTRn	18-93
Table 18-13	MOSTATn Reset Values	18-99
Table 18-14	Transmit Priority of Msg. Objects Based on CAN Arbitration Rules	18-110
Table 18-15	MultiCAN+ Module External Registers	18-114
Table 18-16	MultiCAN+ I/O Control Selection and Setup for XMC4300	18-119
Table 18-17	Interrupt Router Inputs	18-120
Table 18-18	General Timer Module Inputs	18-121
Table 18-19	CAN-to-USIC Connections	18-121
Table 19-1	Abbreviations used in ADC chapter	19-1
Table 19-2	VADC Applications	19-3
Table 19-3	Properties of Result FIFO Registers	19-39
Table 19-4	Function of Bitfield DRCTR	19-40

List of Tables

Table 19-5	EMUX Control Signal Coding	19-55
Table 19-6	Registers Address Space	19-58
Table 19-7	Registers Overview	19-58
Table 19-8	TS16_SSIG Trigger Set VADC	19-65
Table 19-9	Sample Time Coding	19-99
Table 19-10	General Converter Configuration in the XMC4300	19-132
Table 19-11	Synchronization Groups in the XMC4300	19-133
Table 19-12	Analog Connections in the XMC4300	19-134
Table 19-13	Digital Connections in the XMC4300	19-135
Table 20-1	Registers Address Space	20-15
Table 20-2	Register Overview of DAC	20-15
Table 20-3	Analog Connections	20-29
Table 20-4	Service Request Connections	20-29
Table 20-5	Trigger Connections	20-29
Table 20-6	Pattern Generator Synchronization Connections	20-30
Table 21-1	Abbreviations table	21-1
Table 21-2	Applications summary	21-3
Table 21-3	CCU4 slice pin description	21-7
Table 21-4	Connection matrix available functions	21-10
Table 21-5	Dither bit reverse counter	21-54
Table 21-6	Dither modes	21-54
Table 21-7	Timer clock division options	21-58
Table 21-8	Bit reverse distribution	21-65
Table 21-9	Interrupt sources	21-74
Table 21-10	External clock operating conditions	21-78
Table 21-11	Registers Address Space	21-81
Table 21-12	Register Overview of CCU4	21-82
Table 21-13	CCU40 Pin Connections	21-138
Table 21-14	CCU40 - CC40 Pin Connections	21-139
Table 21-15	CCU40 - CC41 Pin Connections	21-140
Table 21-16	CCU40 - CC42 Pin Connections	21-141
Table 21-17	CCU40 - CC43 Pin Connections	21-142
Table 21-18	CCU41 Pin Connections	21-143
Table 21-19	CCU41 - CC40 Pin Connections	21-144
Table 21-20	CCU41 - CC41 Pin Connections	21-145
Table 21-21	CCU41 - CC42 Pin Connections	21-146
Table 21-22	CCU41 - CC43 Pin Connections	21-147
Table 22-1	Abbreviations table	22-1
Table 22-2	Applications summary	22-3
Table 22-3	CCU8 slice pin description	22-9
Table 22-4	Connection matrix available functions	22-12
Table 22-5	Dead time prescaler values	22-28
Table 22-6	Dither bit reverse counter	22-75

List of Tables

Table 22-7	Dither modes	22-75
Table 22-8	Timer clock division options	22-79
Table 22-9	Bit reverse distribution	22-86
Table 22-10	Interrupt sources	22-98
Table 22-11	External clock operating conditions	22-101
Table 22-12	Registers Address Space	22-104
Table 22-13	Register Overview of CCU8	22-106
Table 22-14	CCU80 Pin Connections	22-179
Table 22-15	CCU80 - CC80 Pin Connections	22-180
Table 22-16	CCU80 - CC81 Pin Connections	22-182
Table 22-17	CCU80 - CC82 Pin Connections	22-183
Table 22-18	CCU80 - CC83 Pin Connections	22-185
Table 23-1	Port/Pin Overview	23-1
Table 23-2	Registers Address Space	23-11
Table 23-3	Register Overview	23-12
Table 23-4	Registers Access Rights and Reset Classes	23-13
Table 23-5	Standard PCx Coding	23-17
Table 23-6	Pad Driver Mode Selection	23-19
Table 23-7	Function of the Bits PRx and PSx	23-26
Table 23-8	PCx Coding in Deep Sleep mode	23-28
Table 23-9	Package Pin Mapping Description	23-30
Table 23-10	Package Pin Mapping	23-30
Table 23-11	Port I/O Function Description	23-35
Table 23-12	Port I/O Functions	23-37
Table 24-1	Boot mode pin encoding for PORST	24-4
Table 24-2	System reset boot modes	24-5
Table 24-3	BMI string layout offset table	24-21
Table 24-4	Flash and Debug access policy	24-24
Table 24-5	Error events and codes	24-26
Table 24-6	Registers modified by SSW	24-27
Table 25-1	Debug System available features mapped to functions	25-2
Table 25-2	Peripheral Suspend Support	25-10
Table 25-3	ARM CoreSight™ Component ID codes	25-12
Table 25-4	Peripheral ID Values of XMC4300 ROM Table	25-12
Table 25-5	JTAG INSTRUCTIONS	25-13
Table 25-6	JTAG Debug signal description	25-14
Table 25-7	Serial Wire Debug signal description	25-15
Table 25-8	ETM Trace Port signal description	25-15

www.infineon.com

Published by Infineon Technologies AG