Flash Test user guide

TRAVEO™ T2G family

About this document

Scope and purpose

This guide describes the architecture, configuration, and use of the Flash Test. This guide also explains the functionality of the driver and provides a reference to the driver's API.

The installation, build process, and general information about the use of the EB tresos Studio are not within the scope of this document. See the *EB tresos Studio for ACG8 user's guide* **[7]** for detailed information on these topics.

Intended audience

This document is intended for anyone who uses the FLSTST software of the TRAVEO™ T2G family.

Document structure

Chapter **1 General overview** gives a brief introduction to the Flash Test, explains the embedding in the AUTOSAR environment, and describes the supported hardware and development environment.

Chapter **2 Using Flash Test** provides detailed steps required to use the Flash Test in the application.

Chapter **3 Structure and dependencies** describes the file structure and the dependencies for the Flash Test.

Chapter **4 EB tresos Studio configuration interface** describes the driver's configuration with the EB tresos Studio software.

Chapter **5 Functional description** gives a functional description of all services offered by the Flash Test.

Chapter **6 Hardware resources** describes all hardware resources used by Flash Test.

The **Appendix** provides a complete API reference and access register table.

Abbreviations and definitions

Table 1 Abbreviations

Abbreviations	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
ASIL	Automotive Safety Integrity Level
Basic Software	Standardized part of software which does not fulfill a vehicle functional job.
DET	Default Error Tracer
DEM	Diagnostic Event Manager
DW	Data Wire, a CPU feature. DW is used for peripheral-to-memory and memory-to-peripheral data transfers. DW is also called Peripheral-DMA (P-DMA) controller. Generically, this feature is called “DMA”.

Abbreviations	Description
OS	Operating System
MCAL	Microcontroller Abstraction Layer
MCU	Microcontroller Unit
EB tresos Studio	Elektrobit Automotive configuration framework
TCM	Tightly Coupled Memory
ITCM	Data Tightly Coupled Memory
DTCM	Instruction Tightly Coupled Memory
ECC	Error Correction Code

Related documents

AUTOSAR requirements and specifications

Bibliography

[1] General specification of basic software modules, AUTOSAR release 4.2.2.

[2] Specification of Flash Test, AUTOSAR release 4.2.2.

[3] Specification of standard types, AUTOSAR release 4.2.2.

[4] Specification of ECU configuration parameters, AUTOSAR release 4.2.2.

[5] Specification of default error tracer, AUTOSAR release 4.2.2.

[6] Specification of diagnostics event manager, AUTOSAR release 4.2.2.

Elektrobit automotive documentation

Bibliography

[7] EB tresos Studio for ACG8 user’s guide.

Hardware documentation

[8] The hardware documents are listed in the delivery notes.

Related standards and norms

Bibliography

[9] Layered software architecture, AUTOSAR release 4.2.2.

Table of contents

1 General overview

1.1 Introduction to Flash Test

Flash Test provides an algorithm to test the constant memory.

1.2 User profile

This guide is intended for users with a basic knowledge of the following:

- Automotive embedded systems
- C programming language
- AUTOSAR standard
- Target hardware architecture

1.3 Embedding in the AUTOSAR environment

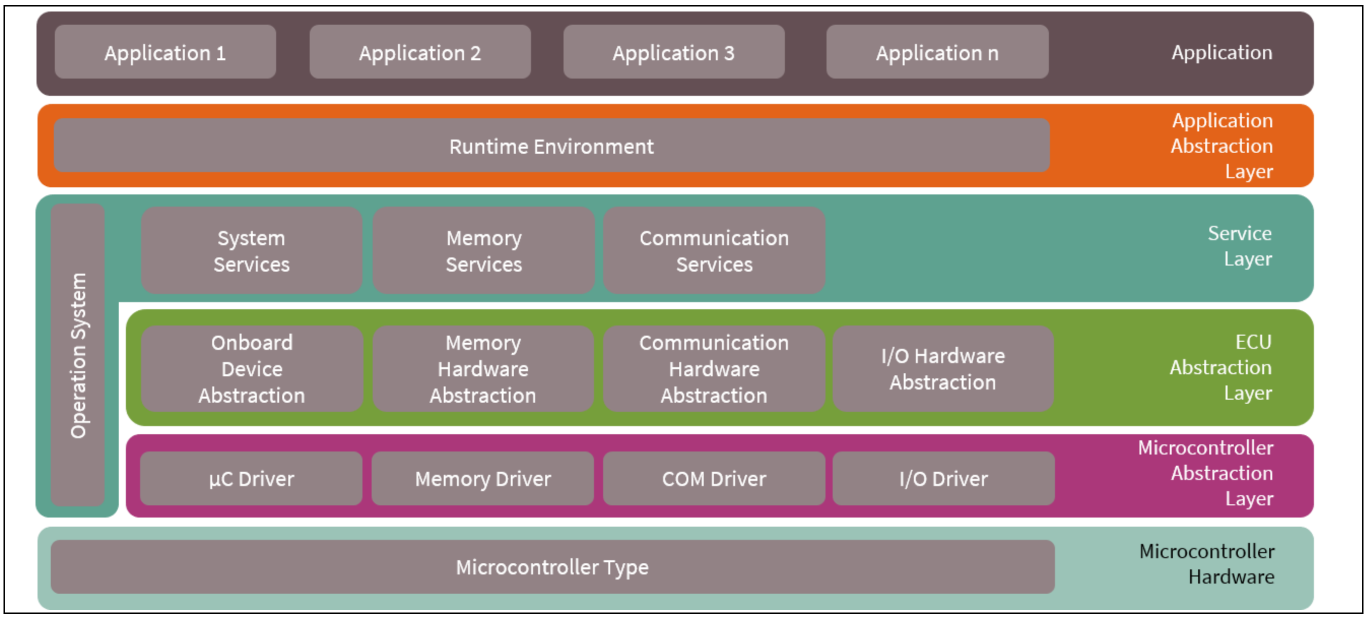

Figure 1 Overview of AUTOSAR software layers

Figure 1 shows the layered AUTOSAR software architecture. The Flash Driver (**Figure 2**) is part of the microcontroller abstraction layer (MCAL), the lowest layer of basic software in the AUTOSAR environment.

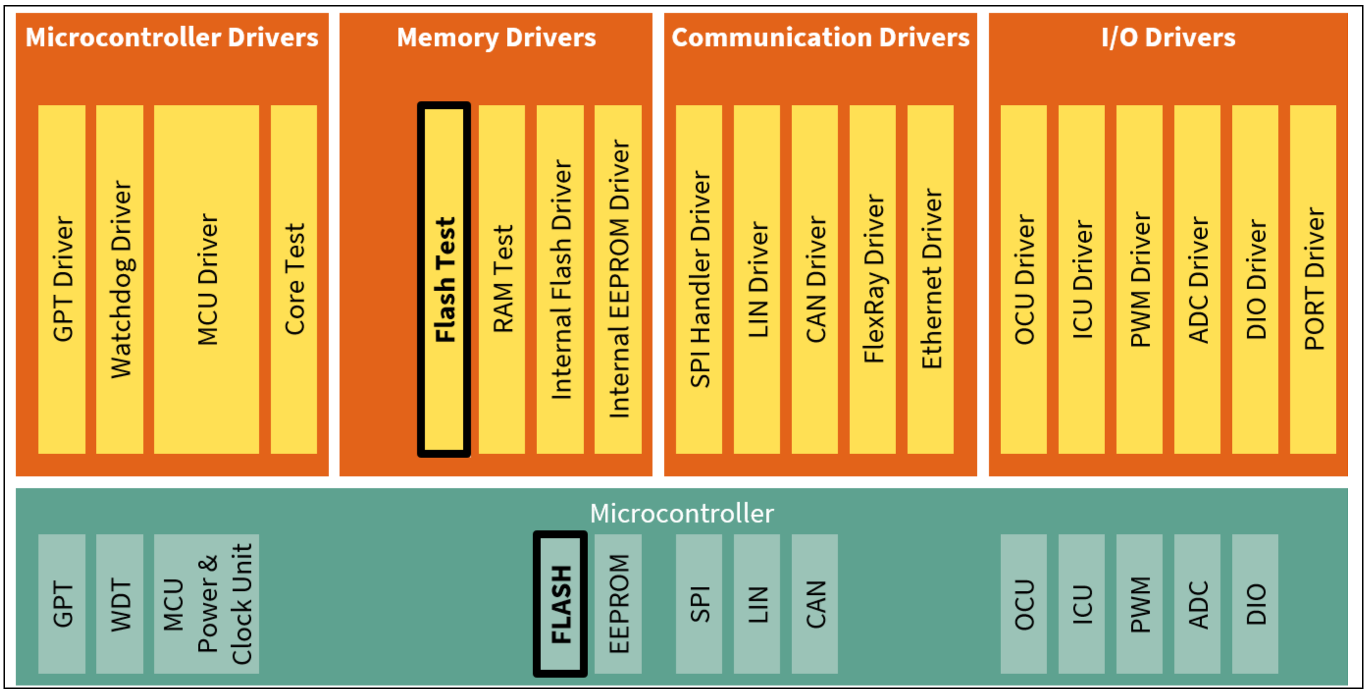

Figure 2 **Flash Test in MCAL layer**

For an overview of the AUTOSAR-layered software architecture, see *layered software architecture* [9].

1.4 Supported hardware

This version of Flash Test supports the TRAVEO™ T2G microcontroller family. No further special external hardware devices are required. See the Resource module users guide for the supported subderivative.

1.5 Development environment

The development environment corresponds to AUTOSAR release 4.2.2. The modules Base, Make, and Resource are needed for proper functionality of Flash Test.

1.6 Character set and encoding

All source code files of Flash Test are restricted to the ASCII character set. The files are encoded in UTF-8 format, with only the 7-bit subset (values 0x00 # 0x7F) being used.

2 Using Flash Test

2.1 Installation and prerequisites

Note: *Before continuing with this chapter, refer to the EB tresos Studio for ACG8 user's guide* **[7]**. *This provides required basic information about the installation procedure of EB tresos ECU AUTOSAR components and the use of the EB tresos Studio and the EB tresos ECU AUTOSAR build environment. It also provides an explanation of how to set up and integrate your own application within the EB tresos ECU AUTOSAR build environment.*

The installation of Flash Test corresponds with the general installation procedure of EB tresos AUTOSAR components given in the documents mentioned above. If the driver has been installed successfully, the driver will appear in the module list of the EB tresos Studio (see *EB tresos Studio for ACG8 user's guide* **[7]**).

This document assumes that the project is properly set up and is using the application template as described in the *EB tresos Studio for ACG8 user's guide* **[7]**. This template provides the necessary folder structure, project, and makefiles needed to configure and compile an application within the build environment. You must be familiar with the use of the command line shell.

2.2 Configuring Flash Test

This section provides a short overview about the configuration structure defined by AUTOSAR to use Flash Test. The configuration of test algorithms, test address, and test size is possible.

For a detailed description of the EB tresos configuration, see chapter **4 EB tresos Studio configuration interface**.

2.2.1 Architecture specifics

This section describes architecture-specific parameters and extensions.

- `FlsTstIncludeFile`: Specifies the file name which is used to include some definitions (for example, declaration for error callout handler).
- `FlsTstErrorCalloutFunction`: Specifies an error callout handler, which is called when any errors are detected during runtime.
- `FlsTstDuplicateAddress`: Specifies the start address of the duplicate memory block in duplicate memory algorithm.
- `FlsTstUseFaultStructForECC`: Specifies the fault structure to be used for ECC algorithm and ECC circuit test.
- `FlsTstUseDWUnitForCRC`: Specifies the DW unit that uses CRC.
- `FlsTstUseDWChSelectForCRC`: Specifies the DW channel that uses CRC.
- `FlsTstCodeFlashAddressToInsertEccError`: Code flash address where ECC error is inserted with `FlsTst_TestEcc()`.
- `FlsTstWorkFlashAddressToInsertEccError`: Work flash address where ECC error is inserted with `FlsTst_TestEcc()`.
- `FlsTstSram0AddressToInsertEccError`: Sram0 address where ECC error is inserted with `FlsTst_TestEcc()`.
- `FlsTstSram1AddressToInsertEccError`: Sram1 address where ECC error is inserted with `FlsTst_TestEcc()`.

- `FlsTstSram2AddressToInsertEccError`: Sram2 address where ECC error is inserted with `FlsTst_TestEcc()`.
- `FlsTstITCMFAddressToInsertEccError`: ITCM address where ECC error is inserted with `FlsTst_TestEcc()`.
- `FlsTstDTCMFAddressToInsertEccError`: DTCM address where ECC error is inserted with `FlsTst_TestEcc()`.
- `FlsTstCodeFlash1AddressToInsertEccError`: Code flash1 address where ECC error is inserted with `FlsTst_TestEcc()`.
- `FlsTstWorkFlash1AddressToInsertEccError`: Work flash1 address where ECC error is inserted with `FlsTst_TestEcc()`.
- `FlsTstRegisterSettingCalloutFunction`: Notify register addresses and values that need to be set by the user.

2.3 Adapting your application

To use Flash Test in your application, first include Flash Test header files by adding the following code lines to your source:

```
#include "FlsTst.h" /* AUTOSAR Flash Test */
```

This publishes all needed function and data prototypes and symbolic names of the configuration into the application.

Flash Test initialization can be done with the following function call and parameter. The parameter is a symbolic name (for example, `FlsTstConf_FlsTstConfigSet_FlsTstConfigSet_0`).

```
FlsTst_Init(&FlsTstConf_FlsTstConfigSet_FlsTstConfigSet_0);
```

Initialization can be done by calling `FlsTst_Init()`.

After initialization, it is possible to start the diagnostics.

Example

```
  void Test_Main(void)
  {
     Std_ReturnType                        Std_ReturnVal;
     FlsTst_BlockIdFgndType                FlsTst_FgndBlkId = 1;
     FlsTst_TestResultFgndType             FlsTst_FgndResult;
     FlsTst_TestSignatureFgndType          FlsTst_FgndSignature;
     FlsTst_ErrorDetailsType               FlsTst_ErrorDitail;

     FlsTst_Init((&FlsTstConf_FlsTstConfigSet_FlsTstConfigSet_0));

     Std_ReturnVal = FlsTst_StartFgnd(FlsTst_FgndBlkId);

     if( Std_ReturnVal == E_OK)
     {
        FlsTst_FgndResult = FlsTst_GetTestResultFgnd();
        if( FlsTst_FgndResult == FLSTST_OK )
        {
            FlsTst_FgndSignature = FlsTst_GetTestSignatureFgnd();
        }
        else
        {
            FlsTst_ErrorDitail = FlsTst_GetErrorDetails();
        }
    }
}
```

2.4 Required items for using Flash Test module

The method for using Flash Test is described in each API.

2.4.1 FlsTst_StartFgnd/FlsTst_MainFunction

- Using CRC algorithm

  When using the CRC algorithm, do the following:

  – Use DW controller to enable.
  – When testing TCM area, allow access from salve port.

  For the setting method, see hardware documentation [8].

The CRC resource to be used can be set in the configuration. For more details, see **FlsTstUseDWUnitForCRC**, and **FlsTstUseDWChSelectForCRC** in Section **4.1.1**.

The calculation method of the signature using the CRC algorithm is as follows (note that 8-bit CRC is not supported):

Table 2 The calculation method of 16-bit CRC

CRC result width:	**16 bits**
Polynomial:	1021h
Initial value:	FFFFh
Input data reflected:	No
Result data reflected:	No
XOR value:	0000h

Example

	(Address)									**Signature**
	+0	+1	+2	+3	+4	+5	+6	+7	+8	
Data bytes (hexadecimal)	00	00	00	00						84C0
	F2	01	83							D374
	33	22	55	AA	BB	CC	DD	EE	FF	F53F
	FF	FF	FF	FF						1D0F

Table 3 The calculation method of 32-bit CRC

CRC result width:	**32 bits**
Polynomial:	04C11DB7h
Initial value:	FFFFFFFFh
Input data reflected:	Yes
Result data reflected:	Yes
XOR value:	FFFFFFFFh

Example

	(Address)									**Signature**
	+0	+1	+2	+3	+4	+5	+6	+7	+8	
Data bytes (hexadecimal)	00	00	00	00						2144DF1C
	F2	01	83							24AB9D77
	33	22	55	AA	BB	CC	DD	EE	FF	B0AE863D
	FF	FF	FF	FF						FFFFFFFF

- Using checksum algorithm

  The calculation method of the signature using checksum algorithm is as follows:

Example 1

- Data1 : 0x01020304
- Data2 : 0x05060708
- Data3 : 0x09

	01	**02**	**03**	**04**
+	05	06	07	08
+	00	00	00	09
Signature	06	08	0A	15

Example 2

- Data1 : 0xFFEEDDCC
- Data2 : 0xBBAA9988
- Data3 : 0x77

	FF	**EE**	**DD**	**CC**
+	BB	AA	99	88
+	00	00	00	77
Signature	BB	99	77	CB

- Using duplicated memory algorithm

  This algorithm is applicable when memory is duplicated. The address of the duplicate data is set by the "`FlsTstDuplicateAddress`" parameter.

- Using ECC algorithm

  This algorithm is applicable when ECC setting of the test area is enabled.

  Even if the ECC setting is enabled, this algorithm cannot be used for uninitialized test areas.

  If ECC error occurs in the cache, FlashTest is excluded from detection.

  When your application calls the `FlsTst_EccFaultJudgment()` API with the fault handler, the detected ECC error is reflected in the test result.

  To detect ECC error with the ECC algorithm, the following settings are required:

  - It is necessary to enable fault capture of ECC error of the area under test.
  - It is necessary to enable fault interrupt. (The user must clear the factor of interrupt.)

  For the setting method, see Hardware Documentation **[8]**.

*Note: You can specify the fault structure to use with configuration setting. For more details, see **FlsTstUseFaultStructForECC** in Section **4.1.5**.*

2.4.2 FlsTst_TestEcc

An ECC error is inserted at the address set in the configuration. For configuration related to **FlsTstCodeFlashAddressToInsertEccError**, **FlsTstWorkFlashAddressToInsertEccError**, **FlsTstSram0AddressToInsertEccError**, **FlsTstSram1AddressToInsertEccError**, **FlsTstSram2AddressToInsertEccError**, **FlsTstITCMAddressToInsertEccError**, **FlsTstDTCMAddressToInsertEccError**, **FlsTstCodeFlash1AddressToInsertEccError** and **FlsTstWorkFlash1AddressToInsertEccError**. See Section **4.1.5**.

The address to insert an ECC error, the target resource (section **5.8 Supported memory**) of this API must be initialized.

Code flash and ITCM require initialization of 8 bytes from the start address. Other resources require initialization of 4 bytes from the start address.

This API requires privileged execution because it accesses the registers which require privileged access.

To detect ECC error with the `FlsTst_TestEcc()` API, the following settings are required.

- Enable ECC function of resource to be tested.
- Enable fault capture of resources to be tested.
- Enable the ECC "auto correct" function of the resource to be tested (When checking ECC 1 bit error of each resource).
- Enable access to the address to be tested.
- Disable instruction cache and data cache of Arm® Cortex®-M7 processor.

For setting method, see hardware documentation **[8]**.

2.4.2.1 Adapting custom configuration

This section describes when the **FlsTstRegisterSettingCalloutFunction** configuration is enabled. When this configuration is enabled, the callout function will be called during execution of the *FlsTst_TestEcc()* API to allow the user to set values in the FLASHC_FLASH_CTL register. If disabled, the callout function is not called, and the *FlsTst_TestEcc()* API sets the values in the FLASHC_FLASH_CTL register. Enable the configuration only if the FLASHC_FLASH_CTL register needs to be set by the user.

Note: *This configuration is disabled by default. If the default configuration is used, there is no need to read this section.*

- About callout function

  If the configuration is enabled, the settings for the FLASHC_FLASH_CTL register must be set in the callout function *FlsTst_UserCalloutRegisterWrite()*. This callout function is called within the *FlsTst_TestEcc()* API.

  The user must implement the callout function. The syntax for the callout function is as follows:

```
void FlsTst_UserCalloutRegisterWrite
(
volatile unsigned long * TargetRegisterAddress,
unsigned long RegisterValue
)
```

Note: *The callout function name is fixed. The argument TargetRegisterAddress is the address of the target register (FLASHC_FLASH_CTL) to be set. The argument RegisterValue is the set value to be set in the register. This value switches FLASHC_FLASH_CTL (MAIN_ECC_INJ_EN[bit:17] or WORK_ECC_INJ_EN[bit:21]). This value does not affect other bits. The user sets registers in this callout according to the information in these arguments. Note that a header file defining this callout function should be included in the configuration file* ***FlsTstIncludeFile****.The number of calls for the callout function depends on the resource under test. The fault capture settings of the user determine the resource under test. For example, if the target is ECC 1 bit and ECC 2 bit errors in CodeFlash, the callout function will be called four times. When checking for ECC 1 bit errors, the callout function is called because FLASHC_FLASH_CTL (MAIN_ECC_INJ_EN[bit:17]) must be enabled. After the check is complete, a callout function is called to restore FLASHC_FLASH_CTL (MAIN_ECC_INJ_EN[bit:17]) to its original value. The same applies when checking for ECC 2 bit errors, so the callout function is called four times.*

- About bus error

  When the API checks for ECC errors, a bus error occurs if the setting of FLASHC_FLASH_CTL (MAIN_ERR_SILENT[bit:18] or WORK_ERR_SILENT[bit:22]) is set to disable (0). Below is a description of how to handle a bus error.

  - Check that the values in the Arm® registers match the following values

    The BFARVALID bit and the PRECISERR bit of the BFSR(BusFault Status Register) are 1.
    The value of BFAR (BusFault Address Register) is the same as the error insertion address by *FlsTst_TestECC()*.

  - If both Arm® register values match, return from bus error.

Set FLASHC_FLASH_CTL (MAIN_ECC_INJ_EN[bit:17] or WORK_ECC_INJ_EN[bit:21]) to disabled, then return from the bus error.

- If Arm® register values do not match, take normal handling.

  The bus error that occurred is not considered to be an error caused by the FlsTst_TestECC() function. Therefore, take the usual handling for bus errors.

Note: *ECC error insertion addresses and how to determine which flash controller to access are described. ECC error insertion addresses can be found in **FlsTstCodeFlashAddressToInsertEccError**, **FlsTstWorkFlashAddressToInsertEccError**, **FlsTstCodeFlash1AddressToInsertEccError**, and **FlsTstWorkFlash1AddressToInsertEccError**. It is possible to determine whether CodeFlash or WorkFlash is the target by using FLASHC_FLASH_CTL (MAIN_ECC_INJ_EN[bit:17] or WORK_ECC_INJ_EN[bit:21]). For example, if FLASHC_FLASH_CTL (MAIN_ECC_INJ_EN[bit:17]) is set to enable(1), CodeFlash is targeted. Also, the value of the argument TargetRegisterAddress of the callout function can determine which flash macro should be accessed. However, if the ECC error insertion address and the Flash controller to be accessed that are unique in the user environment, these determinations are not necessary. For example, only one flash controller, FLASHC_FLASH_CTL(MAIN_ERR_SILENT[bit:18]) is set to disable(0) and FLASHC_FLASH_CTL(WORK_ERR_SILENT[bit:22]) is set to enable(1). In this case, the ECC error insertion address is **FlsTstCodeFlashAddressToInsertEccError**. Since there is only one flash controller to be accessed, no determination is required. In any case, this assumption is based on the fact that FlsTst_TestECC() is started with FLASHC_FLASH_CTL (MAIN_ECC_INJ_EN[bit:17] or WORK_ECC_INJ_EN[bit:21]) set to disable(0).*

2.5 Starting the build process

Do the following to build your application:

Note: *For a clean build, use the build command with target `clean_all`. before (`make clean_all`).*

1. On the command shell, type the following command to generate the necessary configuration-dependent files. See **3.3 Generated files**.

```
> make generate
```

2. Type the following command to resolve required file dependencies:

```
> make depend
```

3. Type the following command to compile and link the application:

```
> make (optional target: all)
```

The application is now built. All files are compiled and linked to a binary file, which can be downloaded to the target hardware.

2.6 Memory mapping

The *FlsTst_MemMap.h* file in the directory *$(TRESOS_BASE)/plugins/MemMap_TS_T40D13M0I0R0/include* is a sample. It is replaced by the file generated by the MEMMAP module. Input to the MEMMAP module is generated as *FlsTst_Bswmd.arxml* in the directory *$(PROJECT_ROOT)/output/generated/swcd* of your project folder.

2.6.1 Memory allocation keyword

- `FLSTST_START_SEC_CODE_ASIL_B` / `FLSTST_STOP_SEC_CODE_ASIL_B`

  The memory section type is CODE. All executable code is allocated in this section.

- `FLSTST_START_SEC_CONST_ASIL_B_UNSPECIFIED` / `FLSTST_STOP_SEC_CONST_ASIL_B_UNSPECIFIED`

  The memory section type is CONST. All constants are allocated in this section.

- `FLSTST_START_SEC_VAR_INIT_ASIL_B_UNSPECIFIED` / `FLSTST_STOP_SEC_VAR_INIT_ASIL_B_UNSPECIFIED`

  The memory section type is VAR. All initialized variables are allocated in this section.

- `FLSTST_START_SEC_VAR_NO_INIT_ASIL_B_UNSPECIFIED` / `FLSTST_STOP_SEC_VAR_NO_INIT_ASIL_B_UNSPECIFIED`

  The memory section type is VAR. All uninitialized variables are allocated in this section.

2.6.2 Restriction on memory allocation

Some sections need to be allocated to specific memory region.

- The section surrounded by `FLSTST_START_SEC_VAR_NO_INIT_ASIL_B_UNSPECIFIED` `/FLSTST_STOP_SEC_VAR_NO_INIT_ASIL_B_UNSPECIFIED`
  - When using DMA:
    This driver does not support the use of DMA descriptor placed in CPU's tightly coupled memories (TCMs).
    So, DMA descriptor needs to be allocated to write-through-area or non-cache-area.
    Therefore, the section will not be allocated to TCMs, instead will be allocated to a user-specific memory region configured by the CPU's Memory Protection Unit (MPU) as write-through or non-cache-able.
  - When not using DMA:
    There is no restriction on memory allocation.

Note: *This restriction is only applied to Arm® Cortex®-M7 devices because they include TCMs and inner cache. There is no restriction for using Arm® Cortex®-M4 devices.*

3 Structure and dependencies

Flash Test consists of static, configuration, and generated files.

3.1 Static files

$(PLUGIN_PATH)=$(TRESOS_BASE)/plugins/FlsTst_TS_* is the path to Flash Test module plugin.

$(PLUGIN_PATH)/lib_src contains all static source files of Flash Test. These files represent the functionality of the driver. Therefore, they are independent of any configuration sets. The files are packed together into a static library.

$(PLUGIN_PATH)/src contains configuration-dependent source files or special derivative files. Each file will be rebuilt when the configuration set is changed.

All necessary source files will be automatically compiled and linked during the build process and all include paths will be set if Flash Test is enabled.

$(PLUGIN_PATH)/include is the basic public include directory needed by the user to include *FlsTst.h*.

$(PLUGIN_PATH)/autosar directory contains the AUTOSAR ECU parameter definition with vendor, architecture, and derivative specific adaptations to create a correct matching parameter configuration for the Flash Test module.

3.2 Configuration files

The configuration of Flash Test is done with the EB tresos Studio. When saving a project, the configuration description is written to the file *FlsTst.xdm*. It is located under *$(PROJECT_ROOT)/config* in your project folder. This file serves as input for the generation of the configuration dependent source and header files during the build process.

3.3 Generated files

During the build process, the following files are generated on the basis of the current configuration description. These files are located in the subfolder *output/generated* of the project folder.

- *include/FlsTst_Cfg.h* provides all symbolic names of the configuration and is included by *FlsTst.h*.
- *include/FlsTst_ExternalInclude.h* contains include files specified by the user.
- *src/FlsTst_PBcfg.c* contains the constant structure for Flash Test configuration.

Note: *Generated source files need not be added to your application make file. They will be compiled and linked automatically during the build process.*

- *swcd/FlsTst_Bswmd.arxml* contains BswModuleDescription.

Note: *Additional steps are required for the generation of BSW module description.*
Select "Build Project" and click "generate_swcd" from the "Project" menu of EB tresos Studio.

3.4 Dependencies

3.4.1 DET

If the default error detection is enabled in the Flash Test module configuration, the DET needs to be installed, configured, and integrated into the application.

3.4.2 DEM

If the failure notification is enabled in the Flash Test module configuration the DEM needs to be installed, configured, and integrated into the application.

3.4.3 BSW scheduler (BSWM)

Flash Test uses the following services of the BSW scheduler (originally named SchM, which is now BswM) to enter and leave critical sections:

- SchM_Enter_FlsTst_FLSTST_EXCLUSIVE_AREA_0(void)
- SchM_Exit_FlsTst_FLSTST_EXCLUSIVE_AREA_0(void)

The user must ensure that the BSW scheduler is properly configured and initialized before using the Flash Test services.

3.4.4 Error callout handler

The error callout handler is called on every error that is detected, independent of whether the default error detection is enabled or disabled. The error callout handler is an ASIL safety extension that is not specified by AUTOSAR. It is configured via the configuration parameter `FlsTstErrorCalloutFunction`.

4 EB tresos Studio configuration interface

The GUI is not part of this delivery. For further information, see the *EB tresos Studio for ACG8 user's guide* [7].

4.1 Containers and configuration parameters

4.1.1 FlsTstConfigSet

4.1.1.1 FlsTstBlockNumberBgnd

Description

This parameter shall represent the number of test blocks available for the background test.

Number of configured FlsTstBlocks in `FlsTstBlockBgndConfigSet` (or 0 if no FlsTstBlocks are configured).

Remarks

None

4.1.1.2 FlsTstBlockNumberFgnd

Description

This parameter shall represent the number of test blocks available for the foreground test.

Number of configured FlsTstBlocks in `FlsTstBlockFgndConfigSet` (or 0 if no FlsTstBlocks are configured).

Remarks

None

4.1.1.3 FlsTstTestCompletedNotification

Description

Pointer to function, which shall be called after finishing the background Flash Test interval.

Remarks

The header file containing the declarations of `FlsTstTestCompletedNotification` must be included using the parameter `FlsTstIncludeFile`.

4.1.1.4 FlsTstUseDWUnitForCRC

Description

Specify the DW unit that uses CRC algorithm.

Remarks

None

4.1.1.5 FlsTstUseDWChSelectForCRC

Description

Specify the DW channel that uses CRC algorithm.

Remarks

None

4.1.1.6 FlsTstBlockBgndConfigSet

Description

This container defines the blocks in background mode.

Remarks

None

4.1.1.7 FlsTstBlockFgndConfigSet

Description

This container defines the blocks in foreground mode.

Remarks

None

4.1.2 FlsTstBlock

4.1.2.1 FlsTstBlockBaseAddress

Description

Start address of the Flash block.

Remarks

This value must be set to 8-byte alignment.

4.1.2.2 FlsTstBlockIndex

Description

Foreground Test: Index identifies the block to be tested by `FlsTst_StartFgnd();`

Background Test: The scheduling for background test shall follow an order defined by this index. '0' means highest priority.

Remarks

These values are unique and have sequential numbering starting from 0.

4.1.2.3 FlsTstBlockSize

Description

This parameter shall represent the Flash Test block size.

Remarks

None

4.1.2.4 FlsTstSignatureAddress

Description

Address of the signature reference value of the Flash Test block.

Remarks

None

4.1.2.5 FlsTstDuplicateAddress

Description

Start address of the duplicate memory block.

Remarks

None

4.1.2.6 FlsTstTestAlgorithm

Description

This is the configuration of the test algorithm for foreground mode and background mode. The algorithm availability is implementation-specific.

Remarks

None

4.1.3 FlsTstConfigurationOfOptApiServices

4.1.3.1 FlsTstGetCurrentStateApi

Description

Adds or removes the service `FlsTst_GetCurrentState()` from the code.

Remarks

None

4.1.3.2 FlsTstGetErrorDetailsApi

Description

Adds or removes the service `FlsTst_GetErrorDetails()` from the code.

Remarks

None

4.1.3.3 FlsTstGetTestResultBgndApi

Description

Adds or removes the service `FlsTst_GetTestResultBgnd()` from the code.

Remarks

None

4.1.3.4 FlsTstGetTestResultFgndApi

Description

Adds or removes the service `FlsTst_GetTestResultFgnd()` from the code.

Remarks

None

4.1.3.5 FlsTstGetTestSignatureBgndApi

Description

Adds or removes the service `FlsTst_GetTestSignatureBgnd()` from the code.

Remarks

None

4.1.3.6 FlsTstGetTestSignatureFgndApi

Description

Adds or removes the service `FlsTst_GetTestSignatureFgnd()` from the code.

Remarks

None

4.1.3.7 FlsTstStartFgndApi

Description

Adds or removes the service `FlsTst_StartFgnd()` from the code.

Remarks

None

4.1.3.8 FlsTstSuspendResumeApi

Description

Adds or removes the services `FlsTst_Suspend()` and `FlsTst_Resume()` from the code.

Remarks

None

4.1.3.9 FlsTstTestEccApi

Description

Adds or removes the service `FlsTst_TestEcc()` from the code.

Remarks

None

4.1.3.10 FlsTstVersionInfoApi

Description

Adds or removes the service `FlsTst_GetVersionInfo()` from the code.

Remarks

None

4.1.4 FlsTstDemEventParameterRefs

4.1.4.1 FLSTST_E_FLSTST_FAILURE

Description

Reference to the DemEventParameter, which shall be issued when the error "Flash Failure" occurs.

Remarks

The header file containing the declarations of `FLSTST_E_FLSTST_FAILURE` must be included using the parameter `FlsTstIncludeFile`.

4.1.5 FlsTstGeneral

4.1.5.1 FlsTstDevErrorDetect

Description

Switch for enabling the default error detection.

Remarks

None

4.1.5.2 FlsTstNumberOfTestedCells

Description

Configures the number of cells to be tested in background mode during one scheduled task (`FlsTst_MainFunction()` call).

Remarks

The unit of "cell" is byte.

For example, if the value of this configuration is 8, 8 bytes will be tested in background mode during one scheduled task.

4.1.5.3 FlsTstNumberOfTestedCellsAtomic

Description

Configures the number of cells to be tested in background mode without checking user requests (Abort, Suspend).

Remarks

This value must be a multiple of 4.

The unit of "cell" is byte.

For example, if the value of this configuration is 4, 4 bytes will be tested in background mode without checking user requests.

4.1.5.4 FlsTstTestCompletedNotificationSupported

Description

Switch to indicate that the notification is supported.

Remarks

None

4.1.5.5 FlsTstTestIntervalIdEndValue

Description

Defines the end value of the test interval ID.

Remarks

None

4.1.5.6 FlsTstTestResultSignature

Description

Configures the result of the test in background mode:

True: Test result is a signature.

False: Test result is OK/Not OK.

Remarks

None

4.1.5.7 FlsTstErrorCalloutFunction

Description

Pointer to function, which shall be reported before reporting to DEM or DET.

Remarks

The header file containing the declarations of `FlsTstErrorCalloutFunction` must be included using the parameter `FlsTstIncludeFile`.

4.1.5.8 FlsTstUseFaultStructForECC

Description

Specify the "fault structure" to be used for ECC algorithm and ECC circuit test.

Remarks

None

4.1.5.9 FlsTstCodeFlashAddressToInsertEccError

Description

Code flash address where ECC error is inserted with `FlsTst_TestEcc()`.

Remarks

All lower 12 bits should be set to 0.

4.1.5.10 FlsTstWorkFlashAddressToInsertEccError

Description

Work flash address where ECC error is inserted with `FlsTst_TestEcc()`.

Remarks

All lower 12 bits should be set to 0.

4.1.5.11 FlsTstSram0AddressToInsertEccError

Description

Sram0 address where ECC error is inserted with `FlsTst_TestEcc()`.

Remarks

All lower 12 bits should be set to 0.

4.1.5.12 FlsTstSram1AddressToInsertEccError

Description

Sram1 address where ECC error is inserted with `FlsTst_TestEcc()`.

Remarks

All lower 12 bits should be set to 0.

4.1.5.13 FlsTstSram2AddressToInsertEccError

Description

Sram2 address where ECC error is inserted with `FlsTst_TestEcc()`.

Remarks

All lower 12 bits should be set to 0.

4.1.5.14 FlsTstITCMAddressToInsertEccError

Description

ITCM address where ECC error is inserted with `FlsTst_TestEcc()`.

Remarks

All lower 12 bits should be set to 0.

4.1.5.15 FlsTstDTCMAddressToInsertEccError

Description

DTCM address where ECC error is inserted with `FlsTst_TestEcc()`.

Remarks

All lower 12 bits should be set to 0.

4.1.5.16 FlsTstCodeFlash1AddressToInsertEccError

Description

Code flash1 address where ECC error is inserted with `FlsTst_TestEcc()`.

Remarks

All lower 12 bits should be set to 0.

4.1.5.17 FlsTstWorkFlash1AddressToInsertEccError

Description

Work flash1 address where ECC error is inserted with `FlsTst_TestEcc()`.

Remarks

All lower 12 bits should be set to 0.

4.1.6 FlsTstIncludeFile

4.1.6.1 FlsTstIncludeFile

Description

`FlsTstIncludeFile` is a list of the file names that shall be included within the driver. Any application-specific symbol (such as the error callout function) that is used by the FlsTst configuration should be included by configuring this parameter.

Remarks

`FlsTstIncludeFile` must be a file name with a .h extension and a unique name; otherwise some errors may occur during configuration.

4.1.7 FlsTstCustomFunction

4.1.7.1 FlsTstRegisterSettingCalloutFunction

Description

If the setting is enabled, a callout function is invoked during execution of the FlsTst_TestEcc() API, allowing the user to set values to specific registers. Informs the user of the address and value of a register that needs to be set through the arguments of the callout function.

True: This function is enabled.

False: This function is disabled. That is, the callout function is not called, and the FlsTst_TestEcc() API sets values in specific registers.

Remarks

Select False unless you need to use this feature. The default value is False. See **Adapting custom configuration** for details on how to adapt if set to True.

5 Functional description

5.1 Initialization

Flash Test provides the following features as an initialization function (`FlsTst_Init`).

- The function notifies the user of the error value `FLSTST_E_ALREADY_INITIALIZED`, if the execution state is not equal to FLSTST_UNINIT.
- The function notifies the user of the error value `FLSTST_E_INIT_FAILED`, if the configuration pointer is a NULL pointer.
- The function initializes all data structures, global variables, and registers with the default values.
- The function changes the execution state to `FLSTST_INIT`.
- The function clears the test results of the foreground and background and test ECC.

5.2 Deinitialization

Flash Test provides the following functions as a diagnostic end processing. (`FlsTst_DeInit`).

- The function initializes all data structures, global variables, and registers with the default values.
- The function changes the execution state to FLSTST_UNINIT.
- The function clears the test results of the foreground and background and test ECC.

5.3 Background test

Background test (`FlsTst_MainFunction`) is called periodically by a scheduler, and is interruptible. The test is split up over many scheduled tasks.

The background test performs the following functions:

- Diagnostic of memory.
- Acquisition of test results. (`FlsTst_GetTestResultBgnd`)
- Acquisition of signature. (`FlsTst_GetTestSignatureBgnd`)
- Acquisition of error information. (`FlsTst_GetErrorDetails`)
- Test completion notice. (FlsTst_TestCompletedNotification)

During background test, the suspend/resume/abort functions are possible. (`FlsTst_Suspend` / `FlsTst_Resume` / `FlsTst_Abort`)

Following is the state transition diagram. Confirmation of the state can be conducted in `FlsTst_GetCurrentState`.

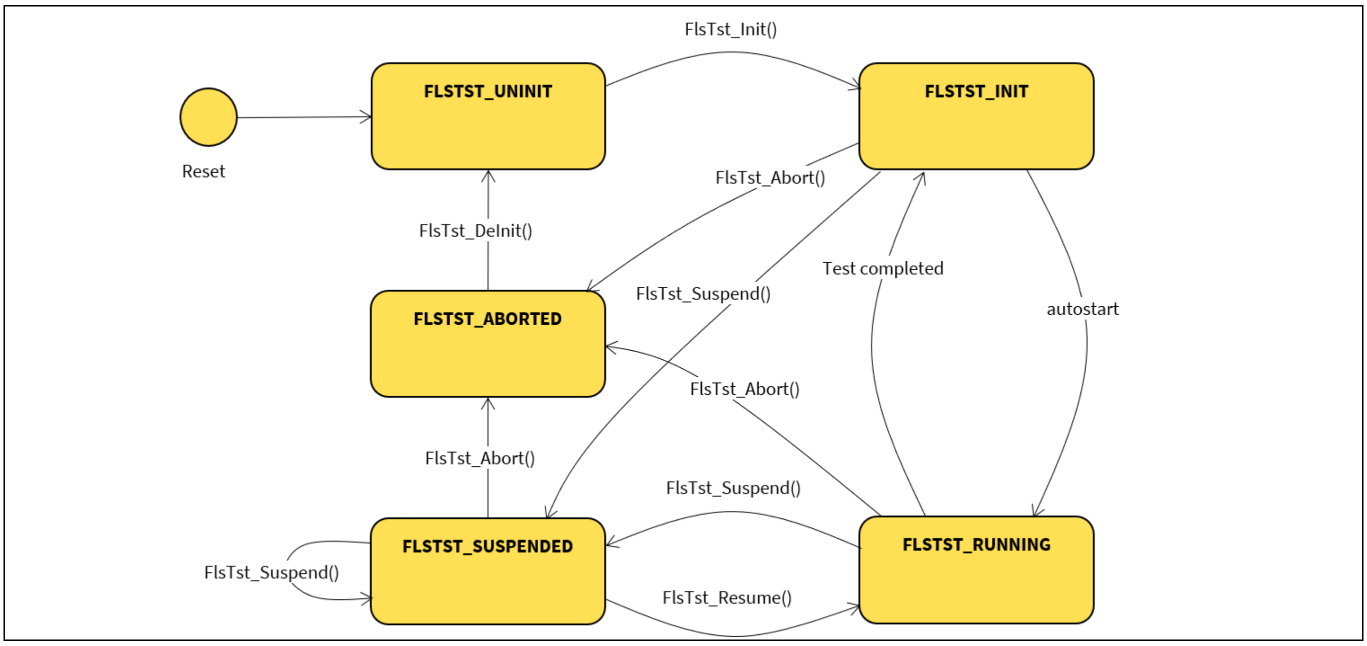

Figure 3 State diagram

5.4 Foreground test

Foreground test (`FlsTst_StartFgnd`) is called via users call. The foreground test performs the following functions:

- Diagnostic of memory.
- Acquisition of test results. (`FlsTst_GetTestResultFgnd`)
- Acquisition of signature. (`FlsTst_GetTestSignatureFgnd`)
- Acquisition of error information. (`FlsTst_GetErrorDetails`)

5.5 ECC test

The ECC test (`FlsTst_TestEcc`) performs the following functions as a diagnostic:

- Diagnostic of ECC circuit.
- Acquisition of error information. (`FlsTst_GetErrorDetails`)

5.6 API parameter checking

If default error detection is enabled, the driver's services perform development error checks. A development error is an error that shall be detected and fixed during the development phase. It shall not occur in production code.

All development errors are reported to the DET, a central error hook function within the AUTOSAR environment.

If an error occurs, the error hook routine will be called and the error code, as well as the service and the module ID, will be passed on as parameters. For more information about the DET and its API, see *specification of default error tracer* [5].

The `FlsTst_Init` function checks state, when being called. If it is in the unexpected state, the `FlsTst_Init` function returns the `FLSTST_E_ALREADY_INITIALIZED`.

The `FlsTst_Init` function checks the function argument, when being called. If it is a NULL pointer, the `FlsTst_Init` function returns the `FLSTST_E_INIT_FAILED`.

The `FlsTst_StartFgnd` function checks the function argument to check whether it is out of range. If it is outside the scope, the `FlsTst_StartFgnd` function returns `FLSTST_E_PARAM_INVALID`.

The functions of the following APIs check if the driver was already initialized. In case of an uninitialized driver, the error code `FLSTST_E_UNINIT` will be reported.

```
>FlsTst_DeInit
>FlsTst_StartFgnd
>FlsTst_Abort
>FlsTst_Suspend
>FlsTst_Resume
>FlsTst_GetTestResultBgnd
>FlsTst_GetTestResultFgnd
>FlsTst_GetTestSignatureBgnd
>FlsTst_GetTestSignatureFgnd
>FlsTst_GetErrorDetails
>FlsTst_TestEcc
>FlsTst_MainFunction
```

The `FlsTst_GetVersionInfo` function checks if the function is called with a NULL pointer. In case of a NULL pointer, the error code `FLSTST_E_PARAM_POINTER` will be reported.

The `FlsTst_Resume` function checks Flash Test execution state. In case of a Flash Test execution state other than FLSTST_SUSPENDED, the error code `FLSTST_E_STATE_FAILURE` will be reported.

5.7 Diagnostic algorithm

`FlsTst_StartFgnd` and `FlsTst_MainFunction` implement fault detection by the following diagnostic algorithms. A simple feature description for each is also given.

5.7.1 CRC (8-bit CRC/16-bit CRC/32-bit CRC)

Description

The algorithm calculates a signature of a memory block using the CRC algorithm. (8-bit CRC is not supported.)

5.7.2 Checksum

Description

The algorithm calculates a signature of a memory block using the checksum algorithm.

5.7.3 Duplicate

Description

This algorithm compares the following two memory blocks:

- Test target memory block
- The duplicate memory block

5.7.4 ECC

Description

This algorithm uses the ECC function of Hardware to detect a fault.

5.8 Supported memory

The supported memory is code flash, and work flash: SRAM, ITCM, DTCM.

5.9 Production error detection

If there is a failure, `FLSTST_E_FLSTST_FAILURE` is reported to the DEM.

When an error occurs, the error hook routine (configured via `FlsTstErrorCalloutFunction`) is also called as well as service ID, module ID, and instance ID are passed as parameters.

5.10 Reentrancy

The following functions are re-entrant to each other and to themselves. All other API functions of Flash Test are not re-entrant.

- FlsTst_GetTestResultBgnd
- FlsTst_GetTestResultFgnd
- FlsTst_GetVersionInfo
- FlsTst_GetTestSignatureBgnd
- FlsTst_GetTestSignatureFgnd
- FlsTst_GetErrorDetails

5.11 Debugging support

Flash Test does not support debugging.

6 Hardware resources

6.1 CPUSS

- CPUSS_RAM0_CTL
- CPUSS_RAM1_CTL
- CPUSS_RAM2_CTL
- CPUSS_CM7_0_CTL
- CPUSS_CM7_1_CTL
- CPUSS_IDENTITY
- CPUSS_ECC_CTL

6.2 FAULT

- FAULT_STRUCT_STATUS
- FAULT_STRUCT_DATA
- FAULT_STRUCT_MASK0
- FAULT_STRUCT_MASK1
- FAULT_STRUCT_MASK2
- FAULT_STRUCT_INTR
- FAULT_STRUCT_INTR_MASK

6.3 FLASHC

- FLASHC_FLASH_CTL
- FLASHC_FLASH_CMD
- FLASHC_ECC_CTL

6.4 DW

- DW_CRC_CTL
- DW_CRC_DATA_CTL
- DW_CRC_POL_CTL
- DW_CRC_LFSR_CTL
- DW_CRC_REM_CTL
- DW_CRC_REM_RESULT
- DW_CH_STRUCT_CH_CTL
- DW_CH_STRUCT_CH_IDX
- DW_CH_STRUCT_CH_CURR_PTR
- DW_CH_STRUCT_INTR
- DW_CH_STRUCT_INTR_MASK
- DW_CH_STRUCT_TR_CMD

6.5 Timer

- Flash Test does not use any hardware timers.

6.6 Interrupts

Flash Test does not use any occupied handler interrupt. However, the user handler is used for API calls of FlashTest.

7 Appendix

7.1 API reference

7.1.1 Data types

7.1.1.1 FlsTst_AlgorithmType

Type

```
typedef enum
{
  FLSTST_16BIT_CRC,                        /* 16 Bit CRC */
  FLSTST_32BIT_CRC,                        /* 32 Bit CRC */
  FLSTST_8BIT_CRC,                        /* 8 Bit CRC */
  FLSTST_CHECKSUM,                         /* Checksum */
  FLSTST_DUPLICATED_MEMORY,                /* Duplicated Memory */
  FLSTST_ECC                                /* ECC */
} FlsTst_AlgorithmType;
```

Description

Type of the test algorithm.

7.1.1.2 FlsTst_BlockConfigType

Type

```
typedef struct
{
  uint32 BlockBaseAddressUpper;
    /* Upper 32 bits of the base address of test block. */
  uint32 BlockBaseAddressLower;
    /* Lower 32 bits of the base address of test block. */
  uint32 BlockIndex;
    /* Foreground test: */
    /* Index identifies block to be tested by FlsTst_StartFgnd(). */
    /* Background test: */
    /* The scheduling for background test shall follow an order */
    /* defined by this index. '0' means highest priority. */
  uint32 BlockSize;
    /* Size of test block */
  uint32 SignatureAddressUpper;
    /* Upper 32 bits of address of the signature reference */
    /* value of the Flash Test block. */
  uint32 SignatureAddressLower;
    /* Lower 32 bits of address of the signature reference */
    /* value of the Flash Test block. */
  uint32 DuplicateAddressUpper;
    /* Upper 32 bits of start address of the */
    /* duplicate memory block. */
  uint32 DuplicateAddressLower;
    /* Lower 32 bits of start address of the */
    /* duplicate memory block. */
  FlsTst_AlgorithmType TestAlgorithm;
    /* Test algorithm for foreground mode and */
    /* background mode. */
  uint32  Resource;
} FlsTst_BlockConfigType;
```

Description

This structure contains the information of the test block.

7.1.1.3 FlsTst_BlockIdFgndType

Type

```
typedef uint32 FlsTst_BlockIdFgndType;
```

Description

This type specifies the identification (ID) for a Flash block to be tested in foreground mode, which is configured in the configuration structure.

7.1.1.4 FlsTst_StateType

Type

```
typedef enum
{
  FLSTST_UNINIT = 0,
    /* The Flash Test is not initialized or not usable. */
  FLSTST_INIT,
    /* The Flash Test is initialized and ready to be started. */
  FLSTST_RUNNING,
    /* The Flash Test is currently running. */
  FLSTST_ABORTED,
    /* The Flash Test is aborted. */
  FLSTST_SUSPENDED
    /* The Flash Test is waiting to be resumed or is waiting to */
    /* start foreground mode test. */
} FlsTst_StateType;
```

Description

This is a state value returned by the API service `FlsTst_GetCurrentState()`.

7.1.1.5 FlsTst_TestResultBgndType

Type

```
typedef struct
{
  uint32 IntervalId;
  FlsTst_TestResultType ResultData;
} FlsTst_TestResultBgndType;
```

Description

Return type of API service `FlsTst_GetTestResultBgnd()`.

7.1.1.6 FlsTst_TestResultFgndType

Type

```
typedef enum
{
  FLSTST_NOT_TESTED = 0,
    /* There is no result available. */
  FLSTST_OK,
    /* The last Flash Test has been tested with OK result */
  FLSTST_NOT_OK
    /* The last Flash Test has been tested with NOT_OK result. */
} FlsTst_TestResultFgndType;
```

Description

Return type of API service `FlsTst_GetTestResultFgnd()`.

7.1.1.7 FlsTst_TestResultType

Type

```
typedef enum
{
  FLSTST_RESULT_NOT_TESTED = 0,
    /* There is no test result available. */
  FLSTST_RESULT_OK,
    /* The last Flash Test interval has been tested with OK result. */
  FLSTST_RESULT_NOT_OK
    /* The last Flash Test interval has been tested with NOT-OK result. */
} FlsTst_TestResultType;
```

Description

Type of test result.

7.1.1.8 FlsTst_TestSignatureBgndType

Type

```
typedef struct
{
  uint32 IntervalId;
  uint32 SignatureData[FLSTST_BGND_BLOCKID_MAX];
} FlsTst_TestSignatureBgndType;
```

Description

Type for test signature in background mode. The signature of "FlsTstBlockIndex = n" is stored in the SignatureData [n]. (n = Maximum value in multiple config set.)

= The size of the [FLSTST_BGND_BLOCKID_MAX] is the number of background test block. If background test block number is 0, the size of the [FLSTST_BGND_BLOCKID_MAX] is 1.

7.1.1.9 FlsTst_TestSignatureFgndType

Type

```
typedef uint32 FlsTst_TestSignatureFgndType;
```

Description

Type for test signature in foreground mode.

7.1.1.10 FlsTst_ResourceType

Type

```
typedef struct
{
  uint32  FlashMainStartAddress;     /* BaseAddressFlash(main) */
  uint32  FlashWorkStartAddress;     /* BaseAddressFlash(work) */
  uint32  Sram0StartAddress;         /* BaseAddressSram0       */
  uint32  Sram1StartAddress;         /* BaseAddressSram1       */
  uint32  Sram2StartAddress;         /* BaseAddressSram2       */
  uint32  CurrentITCMStartAddress;  /* BaseAddressITCM(current cpu) */
  uint32  CurrentDTCMStartAddress;  /* BaseAddressDTCM(current cpu) */
  uint32  ToCpu0ITCMStartAddress;   /* BaseAddressITCM(other=>cpu0) */
  uint32  ToCpu0DTCMStartAddress;   /* BaseAddressDTCM(other=>cpu0) */
  uint32  ToCpu1ITCMStartAddress;   /* BaseAddressITCM(other=>cpu1) */
  uint32  ToCpu1DTCMStartAddress;   /* BaseAddressDTCM(other=>cpu1) */
  uint32  FlashMainEccErrorInsertAddress;     /* Ecc Error Insert Address (main)  */
  uint32  FlashWorkEccErrorInsertAddress;     /* Ecc Error Insert Address (work)  */
  uint32  Sram0EccErrorInsertAddress;         /* Ecc Error Insert Address (Sram0) */
  uint32  Sram1EccErrorInsertAddress;         /* Ecc Error Insert Address (Sram1) */
  uint32  Sram2EccErrorInsertAddress;         /* Ecc Error Insert Address (Sram2) */
  uint32  ITCMEccErrorInsertAddress;          /* Ecc Error Insert Address
(ITCM)(current cpu) */
  uint32  DTCMEccErrorInsertAddress;          /* Ecc Error Insert Address
(DTCM)(current cpu) */
  uint32  Flash1MainStartAddress;   /* BaseAddressFlash1(flash1 main) */
  uint32  Flash1WorkStartAddress;   /* BaseAddressFlash1(flash1 work) */
  uint32  ToCpu2ITCMStartAddress;   /* BaseAddressITCM(other=>cpu2) */
  uint32  ToCpu2DTCMStartAddress;   /* BaseAddressDTCM(other=>cpu2) */
  uint32  ToCpu3ITCMStartAddress;   /* BaseAddressITCM(other=>cpu3) */
  uint32  ToCpu3DTCMStartAddress;   /* BaseAddressDTCM(other=>cpu3) */
  uint32  Flash1MainEccErrorInsertAddress;   /* Ecc Error Insert Address (flash1
main)  */
  uint32  Flash1WorkEccErrorInsertAddress;   /* Ecc Error Insert Address (flash1
work)  */
} FlsTst_ResourceType;
```

Description

Type for address of flash memory area.

7.1.1.11 FlsTst_RegisterType

Type

```
typedef struct
{
  volatile uint32 * CpussBaseAddress;     /* CPUSS.regBaseAddr  */
  volatile uint32 * FlashCBaseAddress;    /* FLASHC.regBaseAddr */
  volatile uint32 * FaultBaseAddress;     /* FAULT.regBaseAddr  */
  volatile uint32 * SystemM7BaseAddress; /* CM7.regBaseAddr     */
} FlsTst_RegisterType;
```

Description

Type for base address of registers.

7.1.1.12 FlsTst_NotifyFctType

Type

```
typedef P2FUNC(void, TYPEDEF, FlsTst_NotifyFctType)(void);
```

Description

Type of test completion notice function.

7.1.1.13 FlsTst_ConfigType

Type

```
typedef struct
{
  P2CONST(FlsTst_BlockConfigType, AUTOMATIC, AUTOMATIC)
  FlsTst_BlockConfigBgndPtr;
    /* Pointer to Test blocks of background test */
  P2CONST(FlsTst_BlockConfigType, AUTOMATIC, AUTOMATIC)
  FlsTst_BlockConfigFgndPtr;
    /* Pointer to Test blocks of foreground test */
  uint32 FlsTst_BlockNumberBgnd;
    /* The number of blocks of background test */
  uint32 FlsTst_BlockNumberFgnd;
    /* The number of blocks of foreground test */
  uint32  FlsTst_TestedCellNumber;
    /* Number of cells to be tested in background mode during one scheduled task */
  FlsTst_NotifyFctType FlsTst_NnotifyFunction;
    /* Notification function */
  P2CONST(FlsTst_ResourceType, AUTOMATIC, AUTOMATIC)
  FlsTst_ResourceAddressPtr;
    /* Pointer to base address of each flash resource */
  P2CONST(FlsTst_RegisterType, AUTOMATIC, AUTOMATIC)
  FlsTst_RegisterBaseDataPtr;
    /* Pointer to base address of the register */
  volatile uint32 * UseDWBaseAddress;   /* Base address of DW used in ConfigSet  */
  volatile uint32 * UseDWChBaseAddress; /* Address of DWCh used in ConfigSet  */
  uint8   FlsTst_SramNum;               /* Judge exist SRAM0/1/2 */
  uint8   FlsTst_CpuType;               /* CPU Type:M4 or M7 or .... */
} FlsTst_ConfigType;
```

Description

This type of external data structure shall contain the initialization data for Flash Test.

7.1.1.14 FlsTst_ErrorDetailsType

Type

```
typedef struct
{
  FlsTst_ErrorDetailsBgndType ErrorDetailsBgnd;
    /* Error details of background test. */
  FlsTst_ErrorDetailsFgndType ErrorDetailsFgnd;
    /* Error details of foreground test. */
  FlsTst_ErrorDetailsTestEccType ErrorDetailsTestEcc;
    /* Error details of ECC circuit test. */
} FlsTst_ErrorDetailsType;
```

Description

Error information is monitored in Flash Test module.

7.1.1.15 FlsTst_TestEccResultType

Type

```
typedef enum
{
  FLSTST_TEST_ECC_NOT_TESTED = 0,
    /* There is no result available. */
  FLSTST_TEST_ECC_OK,
    /* Test results normal */
    /* ECC circuit untestable. */
  FLSTST_TEST_ECC_1BIT_UNDETECTABLE,
    /* 1bit error detection is not possible */
  FLSTST_TEST_ECC_2BIT_UNDETECTABLE
    /* 2bit error detection is not possible */
} FlsTst_TestEccResultType;
```

Description

Error information of the ECC circuit.

7.1.1.16 FlsTst_ErrorDetailsBgndType

Type

```
typedef struct
{
  uint32 BgndIntervalId;    /* Interval ID */
  uint32 BgndBlockId;       /* Test block ID */
  uint32 BgndOccurAddress;  /* Address of error detection */
  uint8 BgndEccBitError;    /* Error information */
} FlsTst_ErrorDetailsBgndType;
```

Description

Error information monitored in the background Flash Test.

BgndEccBitError is either of the following values:

- FLSTST_ECC_NO_BIT_ERROR (0x00U) /* Not detect bit error */
- FLSTST_ECC_1BIT_ERROR (0x01U) /* Detect ECC 1 bit error */
- FLSTST_ECC_UNCORRECTABLE_ERROR (0x02U) /* Detect ECC uncorrectable error */

7.1.1.17 FlsTst_ErrorDetailsFgndType

Type

```
typedef struct
{
  uint32 FgndBlockId;       /* Test block ID */
  uint32 FgndOccurAddress;  /* Address of error detection */
  uint8 FgndEccBitError;    /* Error information */
} FlsTst_ErrorDetailsFgndType;
```

Description

Error information monitored in the foreground Flash Test.

`FgndEccBitError` is either of the following values:

- FLSTST_ECC_NO_BIT_ERROR (0x00U) /* Not detect bit error */
- FLSTST_ECC_1BIT_ERROR (0x01U) /* Detect ECC 1 bit error */
- FLSTST_ECC_UNCORRECTABLE_ERROR (0x02U) /* Detect ECC uncorrectable error */

7.1.1.18 FlsTst_ErrorDetailsTestEccType

Type

```
typedef struct
{
  uint8 TargetResource;
    /* Test target resource */
  FlsTst_TestEccResultType EccCircuitResult;
    /* Error information of the ECC circuit */
} FlsTst_ErrorDetailsTestEccType;
```

Description

Error information monitored in the ECC circuit Test.

TargetResource is either of the following values:

- FLSTST_NONE (0x00U) /* Nothing */
- FLSTST_FLASH_MAIN (0x01U) /* Code Flash */
- FLSTST_FLASH_WORK (0x02U) /* Work Flash */
- FLSTST_SRAM0 (0x03U) /* SRAM0 */
- FLSTST_SRAM1 (0x04U) /* SRAM1 */
- FLSTST_SRAM2 (0x05U) /* SRAM2 */
- FLSTST_ITCM0 (0x06U) /* ITCM0 */
- FLSTST_DTCM0 (0x07U) /* DTCM0 */
- FLSTST_ITCM1 (0x08U) /* ITCM1 */
- FLSTST_DTCM1 (0x09U) /* DTCM1 */
- FLSTST_FLASH1_MAIN (0x11U) /* Code Flash1 */
- FLSTST_FLASH1_WORK (0x12U) /* Work Flash1 */
- FLSTST_ITCM2 (0x13U) /* ITCM2 */
- FLSTST_DTCM2 (0x14U) /* DTCM2 */
- FLSTST_ITCM3 (0x15U) /* ITCM3 */
- FLSTST_DTCM3 (0x16U) /* DTCM3 */

7.1.1.19 FlsTst_EccErrorDetectType

Type

```
typedef enum
{
  FLSTST_NONDETECT,
  FLSTST_DETECT
} FlsTst_EccErrorDetectType;
```

Description

This is the return value of the "FlsTst_EccFaultJudgment" API.

FLSTST_NONDETECT:It is a fault detected during testing of FlashTest's ECC algorithm.

FLSTST_DETECT:It is not a fault detected during testing of FlashTest's ECC algorithm.

7.1.2 Constants

7.1.2.1 Error codes

The service might return the following error codes, if default error detection is enabled:

Table 4 Error codes

Name	Value	Description
`FLSTST_E_STATE_FAILURE`	0x01	Failure within Flash Test execution state
`FLSTST_E_PARAM_INVALID`	0x02	API parameter out of specified range
`FLSTST_E_UNINIT`	0x03	API service used without module initialization
`FLSTST_E_ALREADY_INITIALIZED`	0x04	Flash Test module is already initialized
`FLSTST_E_INIT_FAILED`	0x05	For Variant PB: Configuration pointer is a NULL pointer
`FLSTST_E_PARAM_POINTER`	0x06	Parameter versioninfo is a NULL pointer
`FLSTST_E_FAILURE_FOR_CALLOUT`	0x70	This error ID is used to call the error callout handler.
`FLSTST_E_FLSTST_FAILURE`	External allocation by DEM	At least one block test result is not ok

7.1.2.2 Version information

The following version information is published in the driver's header file:

Table 5 Version information

Name	Value	Description
`FLSTST_SW_MAJOR_VERSION`	See the release notes	Major version number
`FLSTST_SW_MINOR_VERSION`	See the release notes	Minor version number
`FLSTST_SW_PATCH_VERSION`	See the release notes	Patch version number

7.1.2.3 Module information

Table 6 Module information

Name	Value	Description
`FLSTST_MODULE_ID`	104	Module ID
`FLSTST_VENDOR_ID`	66	Vendor ID

7.1.2.4 API service IDs

The following API service IDs are published in the driver's header file:

Table 7 API service IDs

Name	Value	Description
`FlsTst_Init`	0x00	Service ID of `FlsTst_Init`
`FlsTst_DeInit`	0x01	Service ID of `FlsTst_DeInit`
`FlsTst_StartFgnd`	0x02	Service ID of `FlsTst_StartFgnd`

Name	Value	Description
`FlsTst_Abort`	0x03	Service ID of `FlsTst_Abort`
`FlsTst_Suspend`	0x04	Service ID of `FlsTst_Suspend`
`FlsTst_Resume`	0x05	Service ID of `FlsTst_Resume`
`FlsTst_GetCurrentState`	0x06	Service ID of `FlsTst_GetCurrentState`
`FlsTst_GetTestResultBgnd`	0x07	Service ID of `FlsTst_GetTestResultBgnd`
`FlsTst_GetTestResultFgnd`	0x0f	Service ID of `FlsTst_GetTestResultFgnd`
`FlsTst_GetVersionInfo`	0x08	Service ID of `FlsTst_GetVersionInfo`
`FlsTst_GetTestSignatureBgnd`	0x09	Service ID of `FlsTst_GetTestSignatureBgnd`
`FlsTst_GetTestSignatureFgnd`	0x0a	Service ID of `FlsTst_GetTestSignatureFgnd`
`FlsTst_GetErrorDetails`	0x0b	Service ID of `FlsTst_GetErrorDetails`
`FlsTst_TestEcc`	0x0c	Service ID of `FlsTst_TestEcc`
`FlsTst_MainFunction`	0x0d	Service ID of `FlsTst_MainFunction`
`FlsTst_TestCompletedNotification`	0x0e	Service ID of `FlsTst_TestCompletedNotification`

7.1.3 Functions

Note: Watchdog timer is not controlled in each FlashTest API.

7.1.3.1 FlsTst_Init

Syntax

```
void FlsTst_Init
(
const FlsTst_ConfigType* ConfigPtr
)
```

Service ID

0x00

Parameters (in)

ConfigPtr - NULL pointer only.

Parameters (out)

None

Return value

None

DET errors

- FLSTST_E_ALREADY_INITIALIZED - The routine FlsTst_Init is called while Flash Test driver is already initialized.
- FLSTST_E_INIT_FAILED - The routine FlsTst_Init is called while function argument is NULL pointer.

DEM errors

None

Description

Service for Flash Test initialization.

7.1.3.2 FlsTst_DeInit

Syntax

```
void FlsTst_DeInit
(
void
)
```

Service ID

0x01

Parameters (in)

None

Parameters (out)

None

Return value

None

DET errors

- `FLSTST_E_UNINIT` - Flash Test has not been initialized yet.

DEM errors

None

Description

Service for Flash Test DeInitialization.

7.1.3.3 FlsTst_StartFgnd

Syntax

```
Std_ReturnType FlsTst_StartFgnd
(
FlsTst_BlockIdFgndType FgndBlockId
)
```

Service ID

0x02

Parameters (in)

`FgndBlockId` - Number of foreground tests to be executed, which is dependent on configuration.

Parameters (out)

None

Return value

`Std_ReturnType` - E_OK: Foreground test processed. E_NOT_OK: Foreground test not accepted.

DET errors

- `FLSTST_E_PARAM_INVALID` - The parameter FgndBlockId is out of range.
- `FLSTST_E_UNINIT` - Flash Test has not been initialized yet.

DEM errors

None

Description

Service for executing foreground Flash Test.

7.1.3.4 FlsTst_Abort

Syntax

```
void FlsTst_Abort
(
void
)
```

Service ID

0x03

Parameters (in)

None

Parameters (out)

None

Return value

None

DET errors

- `FLSTST_E_UNINIT` - Flash Test has not been initialized yet.

DEM errors

None

Description

Service for aborting Flash Test.

7.1.3.5 FlsTst_Suspend

Syntax

```
void FlsTst_Suspend
(
void
)
```

Service ID

0x04

Parameters (in)

None

Parameters (out)

None

Return value

None

DET errors

- `FLSTST_E_UNINIT` - Flash Test has not been initialized yet.

DEM errors

None

Description

Service for suspending current operation of Flash Test, until `FlsTst_Resume` is called.

7.1.3.6 FlsTst_Resume

Syntax

```
void FlsTst_Resume
(
void
)
```

Service ID

0x05

Parameters (in)

None

Parameters (out)

None

Return value

None

DET errors

- `FLSTST_E_STATE_FAILURE` - The execution state of Flash Test module is not `FLSTST_SUSPENDED`.
- `FLSTST_E_UNINIT` - Flash Test has not been initialized yet.

DEM errors

None

Description

Service for continuing Flash Test at the point it was suspended.

7.1.3.7 FlsTst_GetCurrentState

Syntax

```
FlsTst_StateType FlsTst_GetCurrentState
(
void
)
```

Service ID

0x06

Parameters (in)

None

Parameters (out)

None

Return value

`FlsTst_StateType`

DET errors

None

DEM errors

None

Description

Service returns the current Flash Test execution state.

7.1.3.8 FlsTst_GetTestResultBgnd

Syntax

```
FlsTst_TestResultBgndType FlsTst_GetTestResultBgnd
(
void
)
```

Service ID

0x07

Parameters (in)

None

Parameters (out)

None

Return value

`FlsTst_TestResultBgndType`

DET errors

- `FLSTST_E_UNINIT` - Flash Test has not been initialized yet.

DEM errors

None

Description

Service returns the background Flash Test result.

7.1.3.9 FlsTst_GetTestResultFgnd

Syntax

```
FlsTst_TestResultFgndType FlsTst_GetTestResultFgnd
(
void
)
```

Service ID

0x0f

Parameters (in)

None

Parameters (out)

None

Return value

`FlsTst_TestResultFgndType`

DET errors

- `FLSTST_E_UNINIT` - Flash Test has not been initialized yet.

DEM errors

None

Description

Service returns the foreground Flash Test result.

7.1.3.10 FlsTst_GetVersionInfo

Syntax

```
void FlsTst_GetVersionInfo
(
Std_VersionInfoType* versioninfo
)
```

Service ID

0x08

Parameters (in)

None

Parameters (out)

`versioninfo` - Pointer to where to store the version information of this module.

Return value

None

DET errors

- `FLSTST_E_PARAM_POINTER` - parameter `versioninfo` is a NULL pointer.

DEM errors

None

Description

Service returns the version information of this module.

7.1.3.11 FlsTst_GetTestSignatureBgnd

Syntax

```
FlsTst_TestSignatureBgndType FlsTst_GetTestSignatureBgnd
(
void
)
```

Service ID

0x09

Parameters (in)

None

Parameters (out)

None

Return value

`FlsTst_TestSignatureBgndType`

DET errors

- `FLSTST_E_UNINIT` - Flash Test has not been initialized yet.

DEM errors

None

Description

Service returns Flash Test result in background mode.

7.1.3.12 FlsTst_GetTestSignatureFgnd

Syntax

```
FlsTst_TestSignatureFgndType FlsTst_GetTestSignatureFgnd
(
void
)
```

Service ID

0x0a

Parameters (in)

None

Parameters (out)

None

Return value

FlsTst_TestSignatureFgndType

DET errors

- `FLSTST_E_UNINIT` - Flash Test has not been initialized yet.

DEM errors

None

Description

Service returns Flash Test result in foreground mode.

7.1.3.13 FlsTst_GetErrorDetails

Syntax

```
FlsTst_ErrorDetailsType FlsTst_GetErrorDetails
(
void
)
```

Service ID

0x0b

Parameters (in)

None

Parameters (out)

None

Return value

`FlsTst_ErrorDetailsType`

DET errors

- `FLSTST_E_UNINIT` - Flash Test has not been initialized yet.

DEM errors

None

Description

Service returns error details monitored from the Flash module.

7.1.3.14 FlsTst_TestEcc

Syntax

```
Std_ReturnType FlsTst_TestEcc
(
void
)
```

Service ID

0x0c

Parameters (in)

None

Parameters (out)

None

Return value

`Std_ReturnType` - E_OK: The test result of the ECC circuit is normal.

E_NOT_OK: The test result of the ECC circuit is abnormal.

DET errors

- `FLSTST_E_UNINIT` - Flash Test has not been initialized yet.

DEM errors

None

Description

Service executes a test of ECC hardware. This is only applicable in case the hardware provides such functionality.

Test the target resource's ECC circuit in the following order:

Code Flash > Work Flash > SRAM0 > SRAM1 > SRAM2 > ITCM > DTCM > Code Flash1 > Work Flash1

(If the target resource does not exist on the device being tested, it will not be tested.)

For example, if it is detected in work flash that the ECC circuit is not functioning properly, subsequent target resources will not be tested.

7.1.3.15 FlsTst_MainFunction

Syntax

```
void FlsTst_MainFunction
(
void
)
```

Service ID

0x0d

Parameters (in)

None

Parameters (out)

None

Return value

None

DET errors

- `FLSTST_E_UNINIT` - Flash Test has not been initialized yet.

DEM errors

- `FLSTST_E_FLSTST_FAILURE` - At least one block test result is not ok.

Description

Service for executing Flash Test in background mode.

7.1.3.16 FlsTst_TestCompletedNotification

Syntax

```
void FlsTst_TestCompletedNotification
(
void
)
```

Service ID

0x0e

Parameters (in)

None

Parameters (out)

None

Return value

None

DET errors

None

DEM errors

None

Description

The `FlsTst_TestCompletedNotification` function shall be called every time when a complete test cycle had been tested. The implementation of this function is the user.

The name of the function to be called can be configured by parameter `FlsTstTestCompletedNotification`.

7.1.3.17 FlsTst_EccFaultJudgement

Syntax

```
FlsTst_EccErrorDetectType FlsTst_EccFaultJudgement
(
void
)
```

Service ID

None

Parameters (in)

None

Parameters (out)

None

Return value

`FlsTst_EccErrorDetectType`

DET errors

None

DEM errors

None

Description

Determine whether fault occurred in FlashTest's ECC algorithm.

This function is called from the fault handler.

8 Appendix

8.1 Access register table

8.1.1 CPUSS

Table 8 CPUSS register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
`CPUSS_RAMx_CTL x:0,1,2`	31:0	Word (32 bits)	0x00040000 ECC_INJ_EN[18] 0x00080000 ECC_CHECK_DIS[19]	This register is for the CPUSS system SRAM controller.	`FlsTst_TestEcc` *After API completion, the value of register is set to the value before the API starts.	0x000C0000	Monitoring is not needed.
`CPUSS_CM7_x_CTL x:0,1,2,3`	31:0	Word (32 bits)	0x0000000E/0x0000000D PPB_LOCK[3:0] 0x000200000 ITCM_ECC_INJ_EN [17] 0x000800000 ITCM_ECC_CHECK_DIS[19] 0x002000000 DTCM_ECC_INJ_EN[21]	This register is for the CPUSS system CM7 controller.	`FlsTst_TestEcc` *After API completion, the value of register is set to the value before the API starts.	0x002A000F	Monitoring is not needed.
`CPUSS_ECC_CTL`	31:0	Word (32 bits)	0xXXXXXXXX *X: This value depends on the resource and data to insert error.	ECC control	`FlsTst_TestEcc`	-	Monitoring is not needed.
`CPUSS_IDENTITY`	31:0	Word (32 bits)	- (Read only)	This register is typically used by SW that is executed on different bus masters or with different protection contexts.	`FlsTst_TestEcc` `FlsTst_StartFgnd` `FlsTst_MainFunction` (Read only)	- (Read only)	Monitoring is not needed.

8.1.2 FAULT

Table 9 **FAULT register table**

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value

`FAULT_STRUCT_STATUS`	31:0	Word (32 bits)	0x0000000	Fault status
				Write the value when clearing the value.
				Otherwise use the Read value.
					`FlsTst_StartFgnd`
					`FlsTst_MainFunction`
					*When using ECC algorithm
					`FlsTst_TestEcc`
						0x00000000	Monitoring is not needed.

`FAULT_STRUCT_DATA`	31:0	Word (32 bits)	0x0000000	Fault DATA0/1
				Write the value when clearing the value.
				Otherwise use the Read value.
					`FlsTst_StartFgnd`
					`FlsTst_MainFunction`
					*When using ECC algorithm
					`FlsTst_TestEcc`
						0x00000000	Monitoring is not needed.

`FAULT_STRUCT_MASK0`
`FAULT_STRUCT_MASK1`
`FAULT_STRUCT_MASK2`
	31:0	Word (32 bits)	-
			(Read only)
				-	`FlsTst_TestEcc`	-	Monitoring is not needed.

`FAULT_STRUCT_INTR`	31:0	Word (32 bits)	0x0000001	Clear of interrupt	`FlsTst_StartFgnd`
					`FlsTst_MainFunction`
					*When using ECC algorithm
					`FlsTst_TestEcc`
						0x00000000	Monitoring is not needed.

`FAULT_STRUCT_INTR_MASK`	31:0	Word (32 bits)	0x0000000	Interrupt mask	`FlsTst_TestEcc` *After API completion, the value of register is set to the value before the API starts.	0x00000000	Monitoring is not needed.

8.1.3 FLASHC

Table 10 FLASHC register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
`FLASHC_FLASH_CTL`	31:0	Word (32 bits)	0x0066000 MAIN_ECC_INJ_EN[17] MAIN_ERR_SILENT[18] WORK_ECC_INJ_EN[21] WORK_ERR_SILENT[22]	Flash control make settings related to ECC.	`FlsTst_TestEcc` *After API completion, the value of register is set to the value before the API starts.	0x00000000	Monitoring is not needed.

`FLASHC_FLASH_CMD`	31:0	Word (32 bits)	0x0000001	Flash command
				clear cache and buffer
					`FlsTst_StartFgnd`
					`FlsTst_MainFunction`
					*When using ECC algorithm
					`FlsTst_TestEcc`
						0x00000000	Monitoring is not needed.

`FLASHC_ECC_CTL`	31:0	Word (32 bits)	0xXXXXXXXX *X: This value depends on the resource and data to insert error	ECC control Set the value to insert ECC error.	`FlsTst_TestEcc`	0x00000000	Monitoring is not needed.

If there are multiple flash controllers, access registers belonging to each flash controller.

8.1.4 DW

Table 11 DW register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
`DW_CRC_CTL`	31:0	Word (32 bits)	0x00000000 DATA_REVERSE[0] REM_REVERSE[8]	CRC control	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC(16) algorithm	0x00000000	0x00000000
	31:0	Word (32 bits)	0x00000101 DATA_REVERSE[0] REM_REVERSE[8]	CRC control	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC(32) algorithm	0x00000101	0x00000101
`DW_CRC_DATA_CTL`	31:0	Word (32 bits)	0x00000000	CRC data control	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC algorithm	0x00000000	0x00000000
`DW_CRC_POL_CTL`	31:0	Word (32 bits)	0x10210000	CRC polynomial control	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC(16) algorithm	0x10210000	0x10210000
	31:0	Word (32 bits)	0x04C11DB7	CRC polynomial control	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC(32) algorithm	0x04C11DB7	0x04C11DB7
`DW_CRC_LFSR_CTL`	31:0	Word (32 bits)	- *Depend on data	CRC LFSR control	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC algorithm	0x00000000	Monitoring is not needed.
`DW_CRC_REM_CTL`	31:0	Word (32 bits)	0x00000000	CRC remainder control	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC(16) algorithm	0x00000000	0x00000000
	31:0	Word (32 bits)	0xFFFFFFFF	CRC remainder control	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC(32) algorithm	0xFFFFFFFF	0xFFFFFFFF
`DW_CRC_REM_RESULT`	31:0	Word (32 bits)	- *Read Only	CRC remainder result	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC(32) algorithm	0x00000000	Monitoring is not needed.

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
`DW_CH_STRUCT_CH_CTL`	31:0	Word (32 bits)	0x80000000 ENABLED[31]	DW chanel control	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC algorithm	0x80000000	0x80000000
`DW_CH_STRUCT_CH_CURR_PTR`	31:0	Word (32 bits)	- *Depend on data	DW chanel current descriptor pointer	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC algorithm	0x00000000	Monitoring is not needed.
`DW_CH_STRUCT_INTR`	31:0	Word (32 bits)	0x00000001	DW chanel interrupt	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC algorithm	0x00000000	Monitoring is not needed.
`DW_CH_STRUCT_INTR_MASK`	31:0	Word (32 bits)	0x00000000	DW chanel interrupt mask	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC algorithm	0x00000000	0x00000000
`DW_CH_STRUCT_CH_IDX`	31:0	Word (32 bits)	0x00000000 (clear value)	DW chanel Index	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC algorithm	0x00000000	0x00000000
`DW_CH_STRUCT_TR_CMD`	31:0	Word (32 bits)	0x00000001	DW chanel software trigger	`FlsTst_StartFgnd` `FlsTst_MainFunction` *When using CRC algorithm	0x00000000	Monitoring is not needed.

8.1.5 Core

Table 12 Arm core register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
`CM7_ITCMCR`	31:0	Word (32 bits)	0x00000007 (for ITCM 1bit error) 0x00000003 (for ITCM 2bit error)	Instruction and data tightly-coupled memory control registers	`FlsTst_TestEcc` *After API completion, the value of register is set to the value before the API starts.	0x00000000	Monitoring is not needed.
`CM7_DTCMCR`	31:0	Word (32 bits)	0x00000007 (for DTCM 1bit error) 0x00000003 (for DTCM 2bit error)	Instruction and data tightly-coupled memory control registers	`FlsTst_TestEcc` *After API completion, the value of register is set to the value before the API starts.	0x00000000	Monitoring is not needed.

Revision history

Revision	Issue date	Description of change
**	2018-01-25	New spec.
*A	2018-06-06	- Related documentation Updated data sheet name and number - 2.6 Memory mapping Changed file name from BswImplemation.bmd to FlsTst_Bswmd.arxml. - 3.3 Generated files Changed file name from BswImplemation.bmd to FlsTst_Bswmd.arxml. - A.1.1 Data types Changed "FlsTst_ErrorDetailsTestEccType".
*B	2018-12-20	- 2.2.1 Architecture specifics Added extended configuration. - 2.4.1 FlsTst_StartFgnd/FlsTst_MainFunction Updated "Using CRC algorithm". Updated "Using ECC algorithm". - 2.4.2 FlsTst_TestEcc Added a description. - 4.1.1 FlsTstConfigSet Added "FlsTstUseDWTriggerMuxGroupSelectForCRC". Added "FlsTstUseDWUnitForCRC". Added "FlsTstUseDWChSelectForCRC". - 4.1.5 FlsTstGeneral Added "FlsTstUseFaultStructForECC". - 5.7.4 ECC Updated "Description". - 5.8 Supported memory Updated description.(added ITCM/DTCM) - 6.2 CPUSS - 6.3 FAULT - 6.5 DW Updated resources to use. - 6.7 Interrupts Updated description. - A.1.1 Data types Updated "FlsTst_BlockConfigType" Updated "FlsTst_ResourceType" Updated "FlsTst_RegisterType" Updated "FlsTst_ConfigType" Updated "FlsTst_ErrorDetailsTestEccType" Added "FlsTst_EccErrorDetectType" - A.1.3 Functions Added "FlsTst_EccFaultJudgement"

Revision history

Revision	Issue date	Description of change
		- B.1.1PERI - B.1.2 CPUSS - B.1.3 FAULT - B.1.4 FLASHC - B.1.5 DW Information of each register was updated. - B.1.6 Core Added this chapter
*C	2019-06-14	- Related documentation Updated hardware documentation information.
*D	2019-11-08	- 2.2.1 Architecture specifics Added extended configuration. - 2.4.2 FlsTst_TestEcc Added a description. - 4.1.5 FlsTstGeneral Added "FlsTstCodeFlashAddressToInsertEccError" Added "FlsTstWorkFlashAddressToInsertEccError" Added "FlsTstSram0AddressToInsertEccError" Added "FlsTstSram1AddressToInsertEccError" Added "FlsTstSram2AddressToInsertEccError" Added "FlsTstITCMAddressToInsertEccError" Added "FlsTstDTCMAddressToInsertEccError"
*E	2020-03-23	- 2.6.1 Memory allocation keyword Updated description of memory section.
*F	2020-09-05	- 2.6 Memory mapping Updated description - 2.6.2 Restriction on memory allocation Added this chapter
*G	2020-11-20	MOVED TO INFINEON TEMPLATE.
*H	2021-04-06	- 2.2.1 Architecture specifics - 2.4.1 FlsTst_StartFgnd/FlsTst_MainFunction Deleted about "FlsTstUseDWTriggerMuxGroupSelectForCRC" - 2.4.2 FlsTst_TestEcc Updated description. - 4.1.1.4 FlsTstUseDWTriggerMuxGroupSelectForCRC - 6.1 PERI Deleted. - 6.4 DW Added "DW_CH_STRUCT_TR_CMD". - 7.1.1.11 FlsTst_RegisterType - 7.1.1.13 FlsTst_ConfigType Updated type. - 8.1.1 PERI

Revision history

Revision	Issue date	Description of change
		Deleted. - 8.1.4 DW Added "DW_CH_STRUCT_TR_CMD".
*I	2021-12-23	Updated to Infineon style.
*J	2023-03-03	- 2.2.1 Architecture specifics Added extended configuration. - 2.4.2 FlsTst_TestEcc Updated description. - 2.4.2.1 Adapting custom configuration Added this chapter - 4.1.5 FlsTstGeneral Added "FlsTstCodeFlash1AddressToInsertEccError" Added "FlsTstWorkFlash1AddressToInsertEccError" - 4.1.7 FlsTstCustomFunction - 4.1.7.1 FlsTstRegisterSettingCalloutFunction Added this chapter - 6.3 FLASHC Deleted "FLASHC_CM4_CTL0". - 7.1.1.10 FlsTst_ResourceType Updated type. - 7.1.1.18 FlsTst_ErrorDetailsTestEccType Updated description. - 8.1.1 CPUSS - 8.1.3 FLASHC Updated access tables -8.1.2 FAULT Typo correction
*K	2023-12-08	Web release. No content updates.

Edition 2023-12-08

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email:

erratum@infineon.com

Document reference

002-22287 Rev. *K