MCAL User Manual for Fls_17_Dmu

32-bit TriCore™ AURIX™ TC3xx microcontroller

About this document

Scope and purpose

This User Manual is intended to enable users to integrate the Microcontroller Abstraction Layer (MCAL) software for the TriCore™ AURIX™ family of 32-bit microcontrollers.

This document describes responsibilities of integrator in-charge of integrating MCAL software with the basic software (BSW) stack. This document also provides detailed information on safety, configuration and functions along with examples of usage of significant features.

***Note**: Detailed information about package installation, safety and other generic information that are common across all modules are provided in MCAL User Manual General.*

Intended audience

This document is intended for anyone using the Fls_17_Dmu module of the TC3xx MCAL software.

Document conventions

Table 1 Conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, table and figure captions, screen names, windows, dialog boxes, menus, sub-menus
Italics	Denotes variable(s) and reference(s)
`Courier New`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
>	Indicates that a cascading sub-menu opens when you select a menu item
[cover parentID=<alpha numeric value>]	Used for traceability completeness. Reader should ignore these.

Reference documents

This User Manual should be read in conjunction with the following documents:

- AURIX™ TC3xx MCAL User Manual General
- Specification of Flash Driver, AUTOSAR_SWS_Flash_Driver, AUTOSAR Release 4.2.2
- Specification of Flash Driver, AUTOSAR_SWS_Flash_Driver, AUTOSAR Release 4.4.0

Table of contents

Table of contents

Table of contents

Table of contents

1 Fls_17_Dmu driver

1.1 User information

1.1.1 Description

The FLS driver offers well-defined configuration and standard services as per AUTOSAR for the initialization, read, write and erase of DFlash0. Apart from this there are some non-AUTOSAR services provided as well for example Fls_17_Dmu_CompareWordsSync, Fls_17_Dmu_CancelNonEraseJobs, Fls_17_Dmu_VerifyErase, Fls_17_Dmu_VerifySectorErase, Fls_17_Dmu_GetNotifCaller and so on. User gets an encapsulated access to the underlying DFlash0 through the FLS driver. The scope of the FLS driver is limited only to the DFlash0 Bank. The module is delivered as Post-Build variant. Note:FLS module cannot result in an endless loop leading to a watchdog timeout.

1.1.2 Hardware-software mapping

This section describes the system view of the FLS driver and peripherals administered by it.

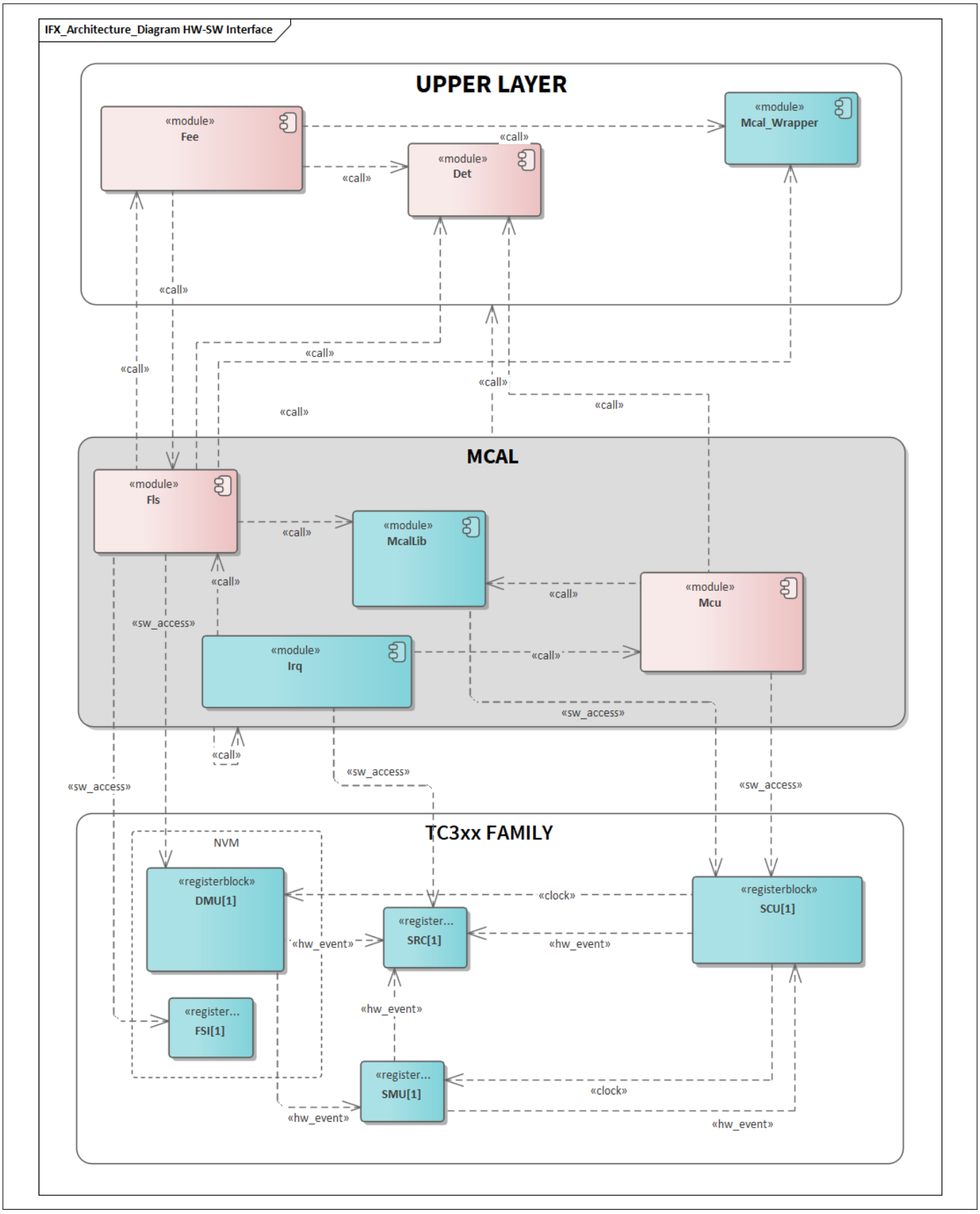

Figure 1 Mapping of hardware-software interfaces

1.1.2.1 DMU - DFlash0: primary hardware peripheral

DMU (DFlash0): primary hardware peripheral

Hardware functional features

The FLS driver uses the DMU for operations such as read, write, suspend, resume, user content count (hardening) and erase DFlash0 memory. The key hardware functional features used by the driver are:

- Single ended sensing mode support for DFlash 0
- Writing and erasing DFlash 0:

i. 8 bytes page programming and 32 bytes burst programming

ii. Erase by multi-sector erase commands

- Suspend, resume for erase operation
- Interrupt service requests for end of busy (EOBM bit) for erase and write operations in hardware

The unsupported features of the DMU are:

- Complement sensing mode for DFlash0
- ECC error reporting to safety management unit (SMU)
- Suspend, resume for write operation

Users of the hardware

The FLSLOADER and FLS drivers utilize the DMU IP module. FLS is used during runtime and FLSLOADER is used during the boot. Hence, the access to the DMU registers is not concurrent

Hardware diagnostic features

- The ECC is used for error detection. Dynamic correction of single, double and triple-bit errors and detection of quad-bit errors
- The SMU alarms configured for the DMU are not monitored by the FLS driver

Hardware events

The following hardware events notified by SFR flags are used in FLS driver:

- Error flags are raised upon occurrence of errors in programming, erasing, reading or erase suspend / resume operation
- Erase verify error (EVER): This flag is set by the erase commands when there is an erase verification error
- Program verify error (PVER): This flag is set by the program commands when there is a program verification error
- Protection error (PROER): This flag is set by the hardware when write or erase command executed on protected memory section
- Operation Error (OPER): This flag is set by the hardware when Flash standard interface (FSI) encounters any error
- Sequence Error (SQER): This flag is set by the hardware when improper DMU command sequences are executed
- End of busy (EOBM): This flag enables the interrupt to report the end of erase and program operations

1.1.2.2 FSI: dependent hardware peripheral

FSI: dependent hardware peripheral

Hardware functional features

Following are the features supported by FSI:

- DMU interfaces to the FSI for all flash operations
- The result of user content count is given by FSI as the number of logic 1 bits in the selected pages at the selected reference current.

Users of the hardware

FLS driver accesses the FSI for hardening check operation.

Hardware diagnostic features

The SMU flags related to FSI are not monitored by the FLS driver.

Hardware events

- Hardware events from FSI are used by the FLS driver for hardening (user content count) check operation:
- The result of user content count command is returned as 13-bit unsigned integer with bits [7:0] in FSI_COMM_1.COMM1 [7:0] and bits [12:8] in FSI_COMM_2.COMM2 [4:0]. The result is returned as 13-bit unsigned integer with bits [7:0] in FSI_COMM_1.COMM1 [7:0] and bits [12:8] in FSI_COMM_2.COMM2 [4:0].

1.1.2.3 SCU: dependent hardware peripheral

SCU: dependent hardware peripheral

Hardware functional features

The FLS driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the f_{SRI}, f_{FSI} and f_{SPB} clock signals for functioning. The system clock is set up through the MCU driver. It is mandatory for the user to set up an appropriate system clock.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the FLS driver.

Hardware events

Hardware events from the SCU are not used by the FLS driver.

1.1.2.4 SRC: dependent hardware peripherals

SRC: dependent hardware peripheral

Hardware functional features

The FLS driver depends on the interrupt router for raising an interrupt to the CPU based on the end of busy event, which indicates the end or finish of the ongoing erase or write job in the HW.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the user software. Interrupt mode is not available when FLS is used with Infineon FEE.

Hardware diagnostic features

The SMU alarms configured for interrupt router are not monitored by the FLS driver.

Hardware events

The interrupt events raised by the interrupt router are serviced by the CPU. The FLS driver provides interrupt handlers as software interfaces, which must be invoked from the ISR. The following hardware events/interrupts are notified for DMU DFlash0:

- Programming completion through end of busy (EOB)
- Erase completion through end of busy (EOB)

1.1.3 File structure

1.1.3.1 C file structure

This section provides details of the C files of the FLS driver.

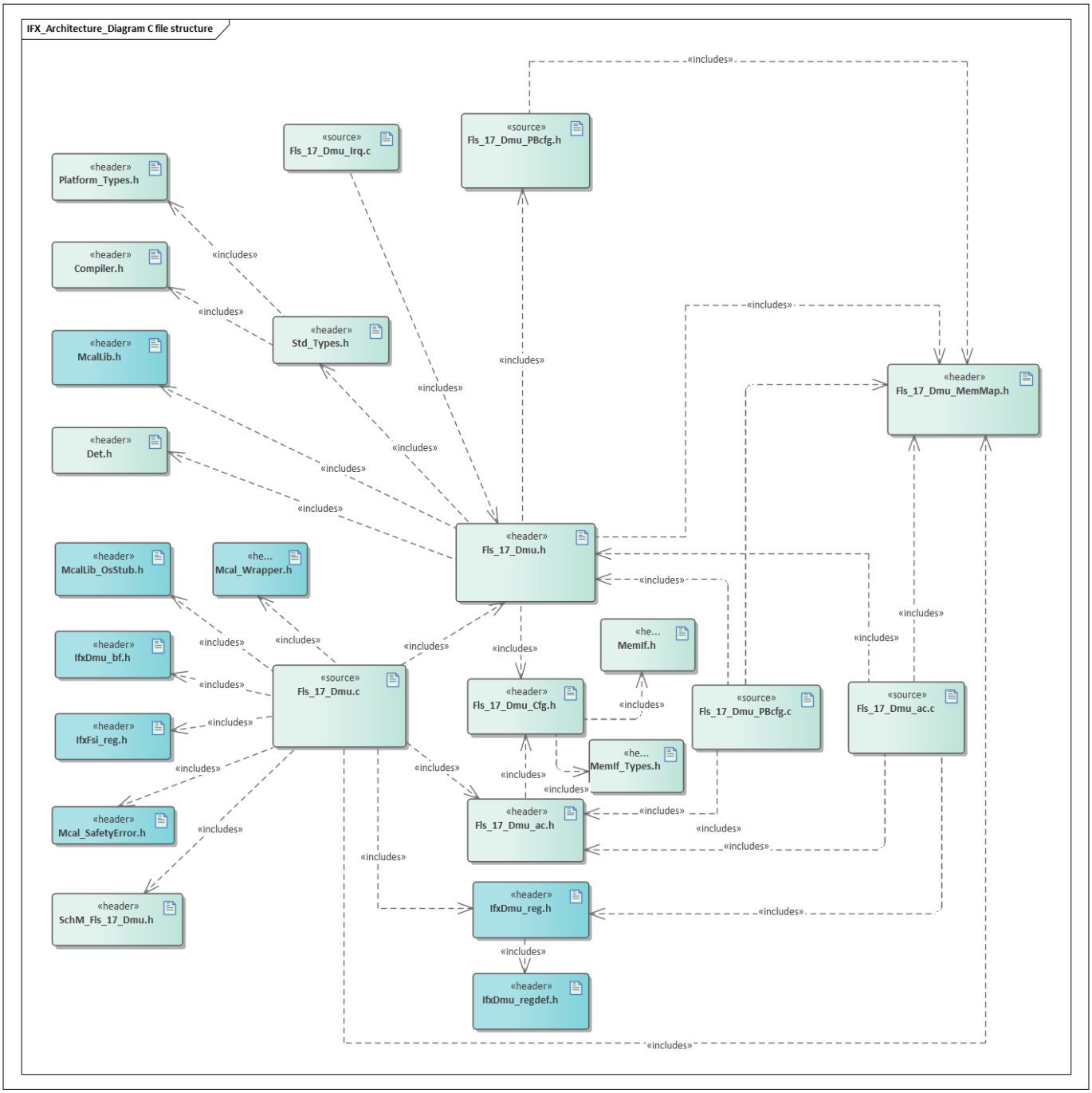

Figure 2 Fls_C_file_structure-1.png

Table 2 C file structure

File name	Description
`Compiler.h`	Provides abstraction from compiler-specific keywords
`Det.h`	Provides the exported interfaces of Development Error Tracer
`Fls_17_Dmu.c`	This file contains functionality of the FLS driver. Version checks are also done in this file.

(table continues...)

Table 2 (continued) C file structure

File name	Description
Fls_17_Dmu.h	This header file exports macros, type definitions, interrupt service routine and function prototypes for the Flash driver
Fls_17_Dmu_Cfg.h	Contains driver pre-compile configuration parameters Contain definitions for all pre-compile time configuration parameters defined as pre-processor directive which are specified for BSW module
Fls_17_Dmu_Irq.c	Interrupt handler file for FLS
Fls_17_Dmu_MemMap.h	File containing the memory section definitions used by the FLS driver
Fls_17_Dmu_PBcfg.c	Contains driver post-build configuration parameters
Fls_17_Dmu_PBcfg.h	File (generated) containing declaration of the post-build configuration data structures
Fls_17_Dmu_ac.c	Command cycles for Flash operations
Fls_17_Dmu_ac.h	Header file for macros used by Fls_17_Dmu_ac.c
IfxDmu_bf.h	SFR header file for Dmu
IfxDmu_reg.h	SFR header file for Dmu
IfxDmu_regdef.h	SFR header file for Dmu
IfxFsi_reg.h	SFR header file for FSI
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB.
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore. This shall be included by other drivers to call OS APIs.
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Mcal_Wrapper.h	Provides the exported interfaces for Production Error and Runtime Development Errors. Implemented by default to include functions of Dem.h and Det.h files. This file can be modified by the user but function prototype is not user modifiable.
MemIf.h	Header file containing exported interfaces and type definitions of MemIf module. *Note: This file is available only for AUTOSAR version 4.4.0.*
MemIf_Types.h	Header file containing the type declaration of MemIf. *Note: This file is available only for AUTOSAR version 4.2.2.*
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Fls_17_Dmu.h	Header file containing prototype of the scheduled function of the Fls driver
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

1.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the FLS driver.

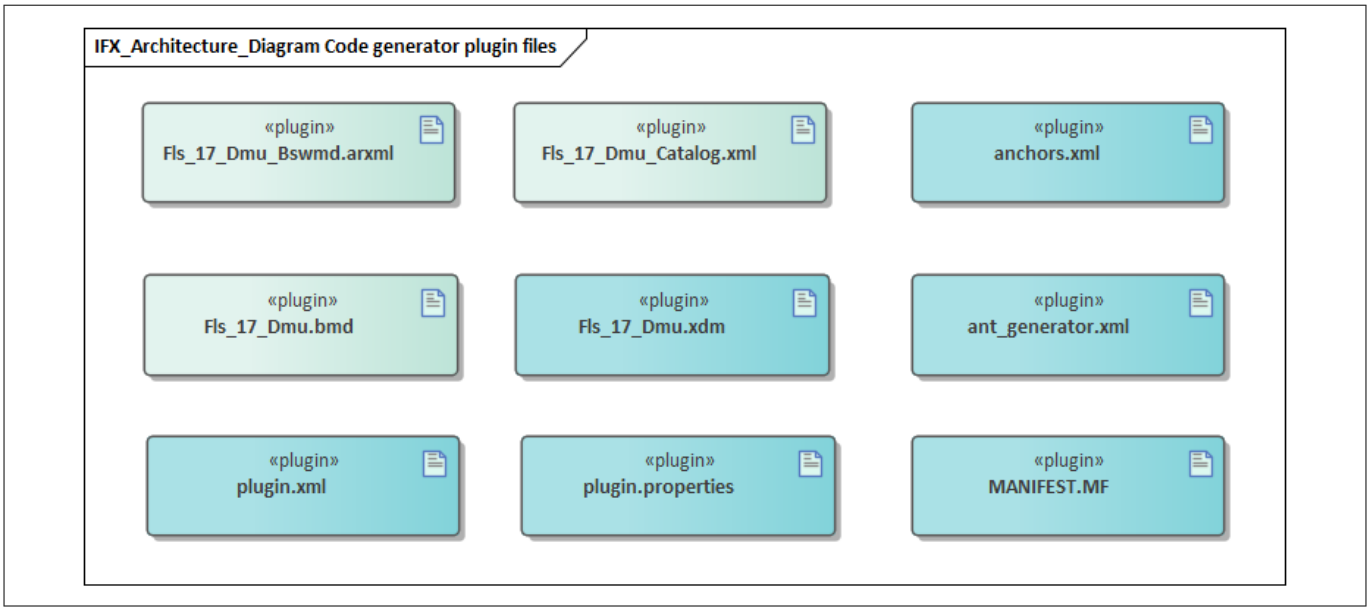

Figure 3 Fls_Code_generator_plugin_files-1.png

Table 3 Code generator plugin files

File name	Description
`Fls_17_Dmu.bmd`	AUTOSAR format XML data model schema file (for each device)
`Fls_17_Dmu.xdm`	Tresos format XML data model schema file
`Fls_17_Dmu_Bswmd.arxml`	AUTOSAR format module description file
`Fls_17_Dmu_Catalog.xml`	AUTOSAR format catalog file
`MANIFEST.MF`	Tresos plugin support file containing the meta-data for FLS driver
`anchors.xml`	Tresos anchors support file for the FLS driver
`ant_generator.xml`	Tresos support file to generate and rename multiple post-build configuration when using variation point.
`plugin.properties`	Tresos plugin support file for the FLS driver
`plugin.xml`	Tresos plugin support file for the FLS driver

1.1.4 Integration hints

This section lists the key points that an integrator or user of the FLS driver must consider.

1.1.4.1 Intergration with AUTOSAR stack

This section lists the modules, which are not part of MCAL, but are required to integrate the FLS driver.

- **EcuM**

  The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of the MCAL, the EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.
- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the re-locatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the file `Fls_17_Dmu_MemMap.h`.

The `Fls_17_Dmu_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows.

```
#if defined FLS_17_DMU_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_UNSPECIFIED
/*User pragma here*/
 #undef FLS_17_DMU_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined FLS_17_DMU_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_UNSPECIFIED
/*User pragma here*/
 #undef FLS_17_DMU_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined FLS_17_DMU_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
/*User pragma here*/
 #undef FLS_17_DMU_START_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
 #undef MEMMAP_ERROR
#elif defined FLS_17_DMU_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
/*User pragma here*/
 #undef FLS_17_DMU_STOP_SEC_VAR_CLEARED_ASIL_B_LOCAL_32
 #undef MEMMAP_ERROR
#elif defined FLS_17_DMU_START_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
/*User pragma here*/
 #undef FLS_17_DMU_START_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined FLS_17_DMU_STOP_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
/*User pragma here*/
 #undef FLS_17_DMU_STOP_SEC_CONFIG_DATA_ASIL_B_LOCAL_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined FLS_17_DMU_START_SEC_CODE_ASIL_B_LOCAL
/*User pragma here*/
 #undef FLS_17_DMU_START_SEC_CODE_ASIL_B_LOCAL
 #undef MEMMAP_ERROR
#elif defined FLS_17_DMU_STOP_SEC_CODE_ASIL_B_LOCAL
/*User pragma here*/
 #undef FLS_17_DMU_STOP_SEC_CODE_ASIL_B_LOCAL
 #undef MEMMAP_ERROR
#endif
```

- **DET**

  The DET module is a part of the AUTOSAR stack that handles all the development and transient faults reported by the BSW. The FLS driver reports all the development errors through the `Det_ReportError()` API and transient faults through the `Det_ReportTransientFault()` API to the DET module. The user of the FLS driver must process all the errors reported to the DET module through the `Det_ReportError()` and `Det_ReportTransientFault()` APIs.

The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and need to be replaced with a complete DET module during the integration phase.

- **Mcal Wrapper**

  This Driver performs reporting of the Runtime errors. The Handling of the reported errors shall be done by the user. The `Mcal_Wrapper_Det_ReportRuntimeError()` API is provided in the `Mcal_Wrapper.c` and `Mcal_Wrapper.h` files as a stub code, and can be updated by the integrator to handle the reported errors. The files `Mcal_Wrapper.c` and `Mcal_Wrapper.h` are user modifiable but function prototype is not user modifiable and by default the Mcal Wrapper function shall calls AUTOSAR DET Module.

  The user of the Fls driver shall process all the Runtime errors reported to the Mcal_Wrapper module. The interface used for reporting Runtime errors in both AUTOSAR is `Mcal_Wrapper_Det_ReportRuntimeError()` API. The `Mcal_Wrapper.c` and `Mcal_Wrapper.h` files are provided in the MCAL package as a stub code and can be replaced with a user specific Runtime error handling module/s during the integration phase.
- **SchM**

  The SchM is not required for the integration of FLS driver.
- **Safety error**

  The FLS driver will report all the detected safety errors through the `Mcal_ReportSafetyError()` API.

  The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError()` API is provided in the `Mcal_SafetyError.c` and `Mcal_SafetyError.h` files as a stub code, and must be updated by the integrator to handle the reported errors.
- **Notifications and callbacks**

  The FLS driver does not implement any notifications. However, the FLS driver reports the job end and error through notification function. These notification functions can be configured by the user in the EB Tresos.
- **Operating system(OS)**

  The OS or application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application.

  The OS files provided by MCAL package is only an example code and must be updated by the integrator with the actual OS files for the desired function.

1.1.4.2 Multicore and Resource Manager

The FLS driver does not support execution on multiple cores.

1.1.4.3 MCU support

The FLS driver is dependent on the MCU driver for the clock configuration. The initialization of the FLS driver must be started only after completing the MCU initialization.

1.1.4.4 Port support

The FLS driver does not use any services provided by the PORT driver.

1.1.4.5 DMA support

The FLS driver does not use any services provided by the DMA driver.

1.1.4.6 Interrupt connections

The following events can trigger an interrupt service request to the Interrupt Router (IR)

- End of BUSY(EOB): if DMU_HF_EER.EOBM = 1B and one of the DMU_HF_STATUS flags D0BUSY, D1BUSY or PFlash flags transitions from 1 to 0 then an interrupt service request is triggered (for example wake-up, erase sequences or program sequences)
- Operation Error (OPER): if DMU_HF_EER.OPERM = 1B and DMU_HF_ERRSR.OPER flag is set
- Protection Error (PROER): if DMU_HF_EER.PROERM = 1B and DMU_HF_ERRSR.PROER flag is set
- Sequence Error (SQER): if DMU_HF_EER.SQERM = 1B and DMU_HF_ERRSR.SQER flag is set
- Program Verify Error (PVER): if DMU_HF_EER.PVERM = 1B and DMU_HF_ERRSR.PVER flag is set
- Erase Verify Error (EVER): if DMU_HF_EER.EVERM = 1B and DMU_HF_ERRSR.EVER flag is set

The event that triggered the interrupt can be determined from the DMU_HF_STATUS and DMU_HF_ERRSR registers. An interrupt event must be triggered when the event appears again and the corresponding status flag is still set. The FLS driver enables and uses only EOB interrupt. Other interrupt mentioned are not used by FLS driver. End of BUSY interrupts are only generated after completion of start-up. The following diagram depicts the interrupt connections of DMU data Flash:

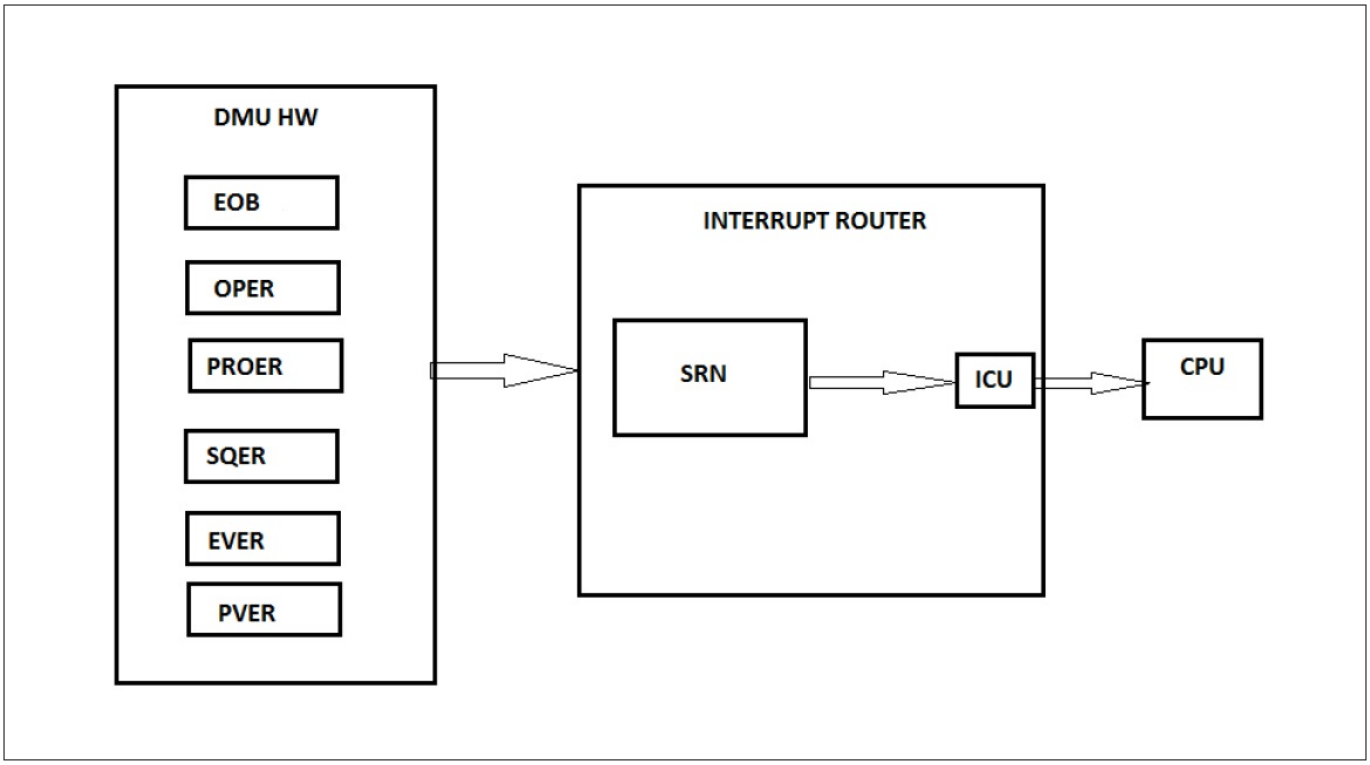

Figure 4 Interrupt mode

Invoking of interrupt handlers provided by the driver must be done by the user. A sample invocation of FLS driver interrupt handler is as follows:

```
ISR(DMUHOST_ISR)
 /* Enable Global Interrupts */
{
 ENABLE();

 /* Call to Flash Interrupt function */
 Fls_17_Dmu_Isr();
}
```

1.1.4.7 Example usage

This section explains an example usage scenario of the FLS driver for a nominal case. Applications usually adopt and modify the configuration and usage sequence as per their use-case.

Configuration of the driver

The configuration of the driver involves the following steps.

1. Configuration of the System Clock f_{SYS}. This configuration is done using the MCU driver.

2. Configuration of the FLS driver: The Flash driver is delivered as a post-build. The configuration of sectors should be done in the `FlsSectorList` container.

The `FlsSector` within the `FlsSectorList` container requires the following parameter: `FlsNumberOfSectors`(number of sectors), `FlsSectorSize`(sector size) and the `FlsSectorStartAddress`(Start/Begin address of the sector).

Note: This also has a dependency on whether the IFX FEE has been used or not.

Initialization of Flash driver

The following code snippet shows the steps involved in the initialization of the Flash driver.

```
#include "Std_Types.h"
#include "Mcu.h"
#include "Fls_17_Dmu.h"
#include "Irq.h"

extern const Mcu_ConfigType Mcu_Config;
extern const Fls_17_Dmu_ConfigType Fls_17_Dmu_Config;

/*Initialization of MCU*/
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock();

/* Initialization of flash module */
Fls_17_Dmu_Init(&Fls_17_Dmu_Config);
#if FLS_USE_INTERRUPTS == ON
/* Configure FLS Module Interrupt Priority.
Use only for FLS INTERRUPT Mode. */
IrqDmu_Init();
#endif
```

Flash operations

`Fls_17_Dmu_MainFunction()` is the only scheduled function provided by the FLS driver. This function should be called periodically, so that it can process the jobs without hardware interrupt support. This API is a service for performing the processing of the Flash read, write, erase and compare jobs. The timeout monitoring of erase or write operations is done based on the hardware(STM) timers. Timeout monitoring is not done for read or compare as the read times are considerably small to monitor through `Fls_17_Dmu_MainFunction()` cycle time.

The code snippet shows an example of the steps involved in erasing, writing and reading a data Flash bank after initialization of the Flash.

```
#define FLS_17_DMU_NVMSECTOR0_STARTADDRESS (0x00000000U)
#define FLS_17_DMU_NVMSECTOR_SIZE (0x20000U)
#define FLS_17_DMU_PAGE_SIZE (8U)
....
uint8 Test_ProgData[2 * FLS_17_DMU_PAGE_SIZE]; /*write buffer*/
uint8 Test_ReadData[2 * FLS_17_DMU_PAGE_SIZE]; /*read buffer*/
...


/*Demo erase*/
Std_ReturnType Fls_DemoErase(void)
{
 /* Erase DFLASH BANK 0 */
 ReturnVal = Fls_17_Dmu_Erase(FLS_17_DMU_NVMSECTOR0_STARTADDRESS,FLS_17_DMU_NVMSECTOR_SIZE);

 /*If erase job scheduled properly*/
 if(ReturnVal == E_OK)
 {
 /* Poll till Erase completed */
 while(Fls_17_Dmu_GetStatus() != MEMIF_IDLE)
 {
 Fls_17_Dmu_MainFunction();
 }
 if(Fls_17_Dmu_GetJobResult() != MEMIF_JOB_OK)
 {
 ReturnVal = E_NOT_OK;
 }
 }
}

/*Demo write*/
Std_ReturnType Fls_DemoWrite(void)
{
 /* Assuming the write bufferTest_ProgData is already filled with some data
 Write first 2 pages of DFLASH BANK 0.*/
 ReturnVal = Fls_17_Dmu_Write(FLS_BANK0_ADDR,Test_ProgData,(2*FLS_PAGESIZE));

 /*If the write job scheduled properly*/
 if(ReturnVal == E_OK)
 {
 /* Poll till Write completed */
 while(Fls_17_Dmu_GetStatus() != MEMIF_IDLE)
 {
 Fls_17_Dmu_MainFunction();
 }
 if(Fls_17_Dmu_GetJobResult() != MEMIF_JOB_OK)
 {
 ReturnVal = E_NOT_OK;
 }
 }
```

```
}

Std_ReturnType Fls_DemoRead(void)
{
/* Read the first two pages */
 ReturnVal = Fls_17_Dmu_Read(FLS_17_DMU_NVMSECTOR0_STARTADDRESS,Test_ReadData,(2 *
FLS_PAGESIZE));

if(ReturnVal == E_OK)
{
 while(Fls_17_Dmu_GetStatus() != MEMIF_IDLE)
 {
 /* Wait till Write is completed */
 Fls_17_Dmu_MainFunction();
 }

 if(Fls_17_Dmu_GetJobResult() != MEMIF_JOB_OK)
 {
 ReturnVal = E_NOT_OK;
 }
}
}
```

Configuration of FlsIllegalStateNotification

User shall configure an user defined function to handle the illegal state. Please refer the FLS demo application for example.

Concurrent access to DFlash0 from FLS driver and user application

It is the responsibility of the integrator to prevent concurrent access to the data flash including the UCB. FLS driver is designed assuming exclusive access to DFlash0. If DFlash0 is shared by FLS driver and user implemented application, then the user needs to take care of the handshaking between FLS driver and user application to avoid concurrent access to DFlash0.

1.1.5 Key architectural considerations

1.1.5.1 API Naming Convention

To meet AUTOSAR specification for the module with an upper multiplicity greater than 1, all the external interfaces in the FLS module are named in the following manner:

<Module Short Name>_<VendorId>_<VendorApiInfix>_<ServiceName>()

The same is followed for error handling. For instance, the name for the Development error "Timeout exceeded" shall be formed in the following way: <MIP>_E_TIMEOUT where <MIP>is the Module implementation prefix of the BSW Module.

1.1.5.2 Error reporting in case of erase and write verification failures for ASR422 and ASR440 versions

In case of ASR422, if erase verification failure (EVER) occurs, the following errors will be reported.

- FLS_17_DMU_E_VERIFY_ERASE_FAILED DET if DET/Safety is enabled

- FLS_17_DMU_E_ERASE_FAILED RTE if runtime error detection is enabled

Similarly, in case of write verification failure, the following errors will be reported.

- FLS_17_DMU_E_VERIFY_WRITE_FAILED DET if DET/Safety is enabled
- FLS_17_DMU_E_WRITE_FAILED RTE if runtime error detection is enabled

In case of ASR440, if erase verification failure (EVER) occurs, the following errors will be reported.

- FLS_17_DMU_E_VERIFY_ERASE_FAILED RTE if runtime error detection and `FlsEraseVerificationEnabled` are enabled
- FLS_17_DMU_E_ERASE_FAILED transient fault

Similarly, in case of write verification failure, the following errors will be reported.

- FLS_17_DMU_E_VERIFY_WRITE_FAILED if runtime error detection and `FlsWriteVerificationEnabled` are enabled
- FLS_17_DMU_E_WRITE_FAILED transient fault

1.2 Assumptions of Use (AoU)

The AoU for the FLS driver are as follows.

- **Check for initialization**

The integrator shall ensure that proper initialization is done by calling the Fls_17_Dmu_Init() API before invoking any other service of the FLS driver.

[cover parentID FLS={78C52790-FD02-4374-ABC5-1E94933BAAAA}]

- **FLS initialization and Initcheck**

The integrator shall verify the correctness of initialization by calling the Fls_17_Dmu_InitCheck() API after the initialization is completed by the Fls_17_Dmu_Init() API.

The Fls_17_Dmu_InitCheck() API checks if the initialized fixed global SFRs and fixed global variables of the FLS driver are initialized according to the configuration.

[cover parentID FLS={4E3B5CD0-694B-410c-A6B1-EDEAE53603CB}]

- **Working of suspend in standalone mode**

When FLS driver is used in standalone mode, the Fls_17_Dmu_SuspendErase() API shall be invoked by the application only when the previous job requested was an erase operation. This is to ensure that any operation other than erase is not suspended unintentionally.

[cover parentID FLS={F516B301-0F41-4864-B0E7-F92DAABC0EEA}]

- **Clock set-up**

Clocks are not set up by the FLS driver. The integrator shall ensure that the clocks needed for the flash operations on DFLASH0 are correctly set up using the MCU driver.

[cover parentID FLS={F34583B5-3E53-4bf2-8CCC-E64FC399B03B}]

- **Non-reentrant APIs**

The FLS driver's APIs are non-reentrant and therefore, the integrator shall ensure that multiple invocation of the FLS API(s) does not occur from different contexts, threads or cores.

[cover parentID FLS={71BD2EA3-E26F-44e3-ADCB-C2F0D64080D2}]

- **Non-usage of DFlash1**

When the FLS driver is being used for operations on the DFlash0, the integrator shall ensure that the DFlash1 is not used independently by any other driver, except for the HSM operations.

[cover parentID FLS={8CC1F5A8-581B-4c82-8364-12E90AF1E1DA}]

- **Using FLS for DFlash1 operation**

The integrator shall not use the FLS driver to perform operations on the DFlash1 hardware.

[cover parentID FLS={8D39DF0E-A919-4762-838D-4B9E9A90650C}]

- **ADER and bus access error behaviour**

For bus access monitoring over SRI, the following errors are reported:

- SRI access address phase error:

If an ECC error occurs during the address phase of an SRI access, then the DMU_HF_ERRSR.ADER bit will be set and an error will be signaled to the SMU. The SRI access will terminate with an error. This error shall not be handled in the FLS driver and shall be handled by the user.

- SRI access write data phase error:

If an ECC error occurs on the data phase of an SRI write access, then an error will be signaled to the SMU. This error shall not be handled in the FLS driver and shall be handled by the user.

[cover parentID FLS={D5F895FF-AD5F-4337-88E3-B5FC8116ADFF}]

- **Write address**

The integrator shall ensure the correctness of the TargetAddress for write operation and also ensure that this address is not protected against writes by the Flash driver.

[cover parentID FLS={4E1CFF64-D76E-440a-8B68-1EC5C9E9B28E}]

- **Access to FLS SFRs from CPU core**

Integrator shall ensure that the FLS driver is invoked from the CPU core that has access to the FLS SFR(s).

[cover parentID FLS={8120AD04-68B1-4eff-AE48-AD14FD6CCD14}]

- **Correctness of 'config pointer'**

The user shall ensure that the config pointer passed is correct.
[cover parentID FLS={95C5FF4A-CBB7-4c18-A117-B754402C4D2C}]

- **Correctness of DFlash0 size configuration**

The integrator shall ensure that the total size of DFlash0 in the hardware is greater than or equal to the size of the data flash (DFlash0) mentioned in the configuration.

[cover parentID FLS={EF72308E-CE84-46eb-9B83-D79951DB6D74}]

- **Invocation of Fls_17_Dmu_GetNotifCaller() API**

The integrator shall ensure that Fls_17_Dmu_GetNotifCaller() is called only from inside the callback notification functions invoked by the FLS.

The Fls_17_Dmu_GetNotifCaller() is needed to identify the notification so that the caller can take appropriate action.

[cover parentID FLS={BE3D0479-4FB5-48e3-B995-10FE8AC2E49B}]

- **No multicore support**

Integrator shall ensure that all the FLS services are executed from one core only. The FLS does not support multicore capability.

[cover parentID FLS={49337170-0313-49f8-91CC-EE972E4A91FA}]

- **Precaution during read operation**

The integrator shall ensure that the source address given for read is not protected against reads.

[cover parentID FLS={F30D701D-9250-4fa0-A700-C8AB627D30A5}]

1.3 Reference information

1.3.1 Configuration interfaces

Supported configuration variant: Post-Build

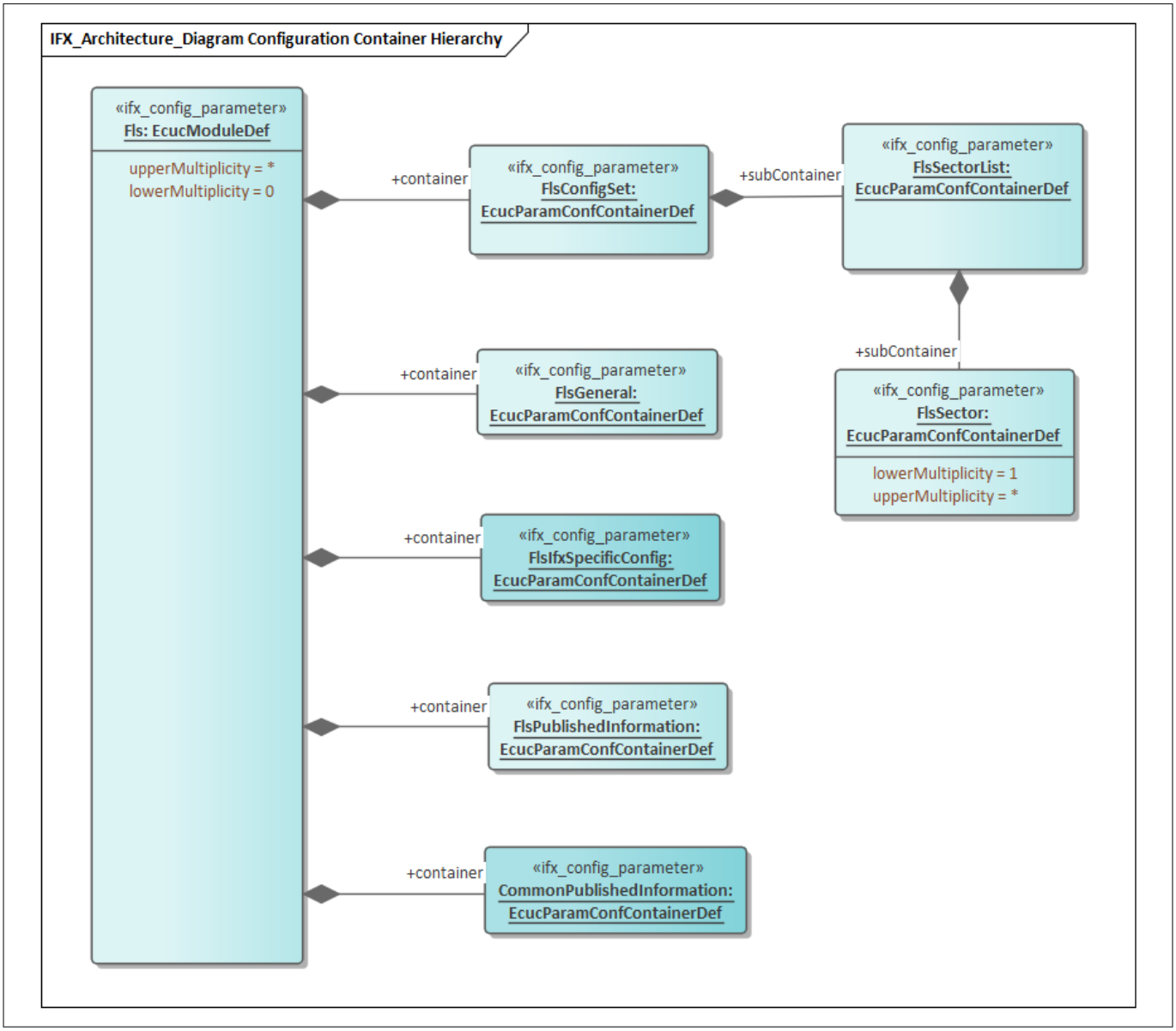

Figure 5 Container hierarchy along with their configuration parameters

1.3.1.1 Container: CommonPublishedInformation

This section describes the information about the module published by the FLS driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.1.1 ArMajorVersion

Table 4 Specification for ArMajorVersion

Name	`ArMajorVersion`		
Description	Major version number of AUTOSAR specification on which the driver implementation is based on.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.2 ArMinorVersion

Table 5 Specification for ArMinorVersion

Name	`ArMinorVersion`		
Description	Minor version number of AUTOSAR specification on which the driver implementation is based on.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per selected Autosar version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.3 ArPatchVersion

Table 6 Specification for ArPatchVersion

Name	`ArPatchVersion`

(table continues...)

Table 6 (continued) Specification for ArPatchVersion

Description	Patch version number of AUTOSAR specification on which the driver implementation is based on.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	As per selected Autosar version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.4 ModuleId

Table 7 Specification for ModuleId

Name	`ModuleId`		
Description	Provides the module ID of the flash driver module ID as described by AUTOSAR : Wp1.1.2 Basic Software Module List		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	92		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.5 Release

Table 8 Specification for Release

Name	`Release`		
Description	Specifies the derivate for which the configuration project is created.		
Multiplicity	1..1	**Type**	EcucStringParamDef

(table continues...)

Table 8 (continued) Specification for Release

Range	String		
Default value	As per the configuration		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.6 SwMajorVersion

Table 9 Specification for SwMajorVersion

Name	`SwMajorVersion`		
Description	Major version number of the vendor specific implementation of the driver.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per driver version.		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.7 SwMinorVersion

Table 10 Specification for SwMinorVersion

Name	`SwMinorVersion`		
Description	Minor version number of the vendor specific implementation of the driver.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per driver version.		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 10 (continued) Specification for SwMinorVersion

Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.8 SwPatchVersion

Table 11 Specification for SwPatchVersion

Name	`SwPatchVersion`		
Description	Patch version number of the vendor specific implementation of the driver.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per driver version.		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.9 VendorApiInfix

Table 12 Specification for VendorApiInfix

Name	`VendorApiInfix`		
Description	The parameter is used to specify the vendor specific name. Default value is set to Dmu, as this is the unique name of the Fls module provided by IFX.		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	Dmu		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-

(table continues...)

Table 12 (continued) Specification for VendorApiInfix

Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.10 VendorId

Table 13 Specification for VendorId

Name	`VendorId`		
Description	Specifies the vendor Id for Infineon		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2 Container: Fls

This container holds the configuration of the FLS (internal or external) driver module.

The multiplicity describes the number of Flash drivers present, therefore, there will be one container for each Flash driver in the ECUC template. When no Flash driver is present, the multiplicity is 0.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: -

1.3.1.3 Container: FlsConfigSet

This container is for the runtime configuration parameters of the Flash driver.

Implementation Type: Fls_17_Dmu_ConfigType.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.3.1 FlsAcErase

Table 14 Specification for FlsAcErase

Name	`FlsAcErase`

(table continues...)

Table 14 (continued) Specification for FlsAcErase

Description	Address offset in RAM to which the erase flash access code shall be loaded. Used as function pointer to access the erase flash access code. This parameter is not used and hence not supported. In TC3xx, Pflash and Dflash can be read in parallel and hence there is no need to load Dflash access code into RAM.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.2 FlsAcWrite

Table 15 Specification for FlsAcWrite

Name	`FlsAcWrite`		
Description	Address offset in RAM to which the write flash access code shall be loaded. Used as function pointer to access the write flash access code. This parameter is not used and hence not supported. In TC3xx, Pflash and Dflash can be read in parallel and hence there is no need to load Dflash access code into RAM.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.3 FlsCallCycle

Table 16 Specification for FlsCallCycle

Name	`FlsCallCycle`		
Description	Cycle time of calls of the main function for the Flash driver(in seconds). This parameter is used in the timeout monitoring for the write/erase jobs. A value of 10 ms is selected as default assuming that this duration would be a reasonable frequency to check the status of scheduled user jobs.		
Multiplicity	1..1	**Type**	EcucFloatParamDef
Range	0.0001 - 1		
Default value	0.01		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.4 FlsDefaultMode

Table 17 Specification for FlsDefaultMode

Name	`FlsDefaultMode`		
Description	This parameter is the default read mode of the data flash(DFLASH0) on the device after initialization. The default value has been selected assuming that a read in MEMIF_MODE_SLOW mode(32 bytes) would be reasonable for the user.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	MEMIF_MODE_FAST: driver is working in the fast(burst) mode. MEMIF_MODE_SLOW: driver is working in the slow mode.		
Default value	MEMIF_MODE_SLOW		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.5 FlsEraseVerifyErrNotif

Table 18 Specification for FlsEraseVerifyErrNotif

Name	`FlsEraseVerifyErrNotif`		
Description	User defined notification function pointer of type 'void fn_name (void)'. This notification function is called by the FLS driver for giving notification of the EVER bit error during the erase job. If the FlsEraseVerifyErrNotif is configured as NULL, the notification functions are not called. This parameter is valid only if the Infineon FEE is used and should be configured as Fee_17_JobEraseErrorNotification. The Fee_17_JobEraseErrorNotification is the name of the Infineon FEE erase verification error notification function and therefore has been given as the default value. If the Infineon FEE is not used, then this parameter is not supported. The post build variant value is false for this parameter since the default value is non-editable.		
Multiplicity	0..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	Fee_17_JobEraseErrorNotification		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	IFX	**Scope**	LOCAL
Dependency	FlsIfxFeeUse		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.6 FlsJobEndNotification

Table 19 Specification for FlsJobEndNotification

Name	`FlsJobEndNotification`		
Description	User defined notification function pointer of type void fn_name (void). This notification function is called by the FLS driver on successful completion of the job. If the FlsJobEndNotification is configured as NULL, the notification functions are not called. If the Infineon FEE is used, it should be configured as Fee_JobEndNotification. Assuming the usage is with Infineon FEE, the default value has been set as Fee_JobEndNotification. The integrator or user has to verify the function address if numerical value is provided.		
Multiplicity	0..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	Fee_JobEndNotification		

(table continues...)

Table 19 (continued) Specification for FlsJobEndNotification

Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.7 FlsJobErrorNotification

Table 20 Specification for FlsJobErrorNotification

Name	`FlsJobErrorNotification`		
Description	User defined notification function pointer of type void fn_name (void). This notification function is called by the FLS driver on cancellation of the job or a failure in executing the job. If the FlsJobErrorNotification is configured as NULL, the notification functions is not called. If the Infineon FEE is used, it should be configured as Fee_JobErrorNotification. Assuming the usage with Infineon FEE, the default value has been given as Fee_JobErrorNotification. The integrator/user has to verify the function address if numerical value is provided.		
Multiplicity	0..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	Fee_JobErrorNotification		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.8 FlsMaxReadFastMode

Table 21 Specification for FlsMaxReadFastMode

Name	`FlsMaxReadFastMode`

(table continues...)

Table 21 (continued) Specification for FlsMaxReadFastMode

Description	The maximum number of bytes to read in one cycle of the job processing of the Flash driver in fast mode. This configuration of this parameter will affect Compare and Blank check operation as well. The value configured for FlsMaxReadFastMode should be more than the value configured for FlsMaxReadNormalMode. Therefore, the default value has been set assuming a word aligned read address from data flash(DFLASH0) and more than the value of FlsMaxReadNormalMode.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - FlsTotalSize		
Default value	64		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	FlsTotalSize		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.9 FlsMaxReadNormalMode

Table 22 Specification for FlsMaxReadNormalMode

Name	`FlsMaxReadNormalMode`		
Description	The maximum number of bytes to read in one cycle of the job processing of the Flash driver in normal mode. This configuration of this parameter will affect Compare and Blank check operation as well. The default value has been given assuming the read address from DFLASH0 is word aligned and is less than the value of FlsMaxReadFastMode.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - FlsTotalSize		
Default value	32		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	FlsTotalSize		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.10 FlsMaxWriteFastMode

Table 23 Specification for FlsMaxWriteFastMode

Name	`FlsMaxWriteFastMode`		
Description	The maximum number of bytes to write in one cycle of the job processing of the Flash driver. In Aurix the write can be either page write (1 page = 8 bytes) or burst write (4 pages = 32 bytes). This parameter is not supported as the burst mode for write is used by default and if the length of data to be written is less than or equal to 24 bytes (that is less than or equal to 4 pages) then page write is used for these remaining bytes.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	32 - 32		
Default value	32		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.11 FlsMaxWriteNormalMode

Table 24 Specification for FlsMaxWriteNormalMode

Name	`FlsMaxWriteNormalMode`		
Description	The maximum number of bytes to write in one cycle of the job processing of the Flash driver. In Aurix the write can be either page write (1 page = 8 bytes) or burst write (4 pages = 32 bytes). This parameter is not supported as the burst mode for write is used by default and if the length of data to be written is less than or equal to 24 bytes (that is less than or equal to 4 pages) then page write is used for these remaining bytes.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	32 - 32		
Default value	32		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL

(table continues...)

Table 24	(continued) Specification for FlsMaxWriteNormalMode
Dependency	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.3.12 FlsProgVerifyErrNotif

Table 25	Specification for FlsProgVerifyErrNotif		
Name	`FlsProgVerifyErrNotif`		
Description	User defined notification function pointer of type void fn_name (void). This notification function is called by the FLS Driver for giving notification of the PVER error during write/programming job. If the FlsProgVerifyErrNotif is configured as NULL then the notification function is not called. This parameter is valid only if IFX FEE is used and should be configured as Fee_17_JobProgErrorNotification. The Fee_17_JobProgErrorNotification() is the Infineon FEE programming error notification and therefore has been given as the default value. If Infineon FEE is not used, then this parameter is not supported. The post build variant value is false for this parameter since the default value is non-editable.		
Multiplicity	0..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	Fee_17_JobProgErrorNotification		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	IFX	**Scope**	LOCAL
Dependency	FlsIfxFeeUse		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.13 FlsProtection

Table 26	Specification for FlsProtection		
Name	`FlsProtection`		
Description	This parameter is not supported as the protection is best handled by the FlsLoader. This parameter is unused and hence disabled.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		

(table continues...)

Table 26 (continued) Specification for FlsProtection

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.14 FlsWaitStateErrorCorrection

Table 27 Specification for FlsWaitStateErrorCorrection

Name	`FlsWaitStateErrorCorrection`		
Description	Defines wait state configuration for error correction. Minimum value for the ECC cycles : Ceiling(tDFECC * fFSI) The wait cycles to be programmed in the DMU_HF_DWAIT register is ECC cycles - 1. For example, if the tDFECC = 20 ns, with fFSI = 100 MHz. The number of error correction cycles equals 2 therefore program values are: DMU_HF_DWAIT.RECC = 1 So for error correction cycles of 2, the value to be entered here is 1.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	FLS_17_DMU_WAITSTATE_ERRCOREC_0 - FLS_17_DMU_WAITSTATE_ERRCOREC_7		
Default value	FLS_17_DMU_WAITSTATE_ERRCOREC_1		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.15 FlsWaitStateRead

Table 28 Specification for FlsWaitStateRead

Name	`FlsWaitStateRead`

(table continues...)

Table 28 (continued) Specification for FlsWaitStateRead

Description	Defines wait state configuration for read access. Minimum value for the DFlash0 read cycles : Ceiling (tDF * fFSI) The wait cycles to be programmed in the DMU_HF_DWAIT register is DFLASH read cycles - 1. For example, if the tDF = 100 ns and fFSI = 100 MHz. The number of DFlash read cycles equals 10, therefore program values are: DMU_HF_DWAIT.RFLASH = 9 So for read cycles of 10, the value to be entered here is 9.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	FLS_17_DMU_WAITSTATE_READ_0 - FLS_17_DMU_WAITSTATE_READ_255		
Default value	FLS_17_DMU_WAITSTATE_READ_9		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4 Container: FlsDemEventParameterRefs

Container for the references to DemEventParameter elements which shall be invoked using the Mcal_Wrapper_Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.

Note: This container is not applicable and made non-editable. This configuration container is not used in the code but it is listed for AUTOSAR compatibility.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

1.3.1.4.1 FLS_E_COMPARE_FAILED

Table 29 Specification for FLS_E_COMPARE_FAILED

Name	`FLS_E_COMPARE_FAILED`		
Description	Reference to the DemEventParameter which shall be issued when the error "Flash compare failed (HW)" has occurred. *Note: This parameter is not applicable and made non-editable. This configuration parameter is not used in the code but it is listed for AUTOSAR compatibility.*		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef

(table continues...)

Table 29 (continued) Specification for FLS_E_COMPARE_FAILED

Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.2 FLS_E_ERASE_FAILED

Table 30 Specification for FLS_E_ERASE_FAILED

Name	`FLS_E_ERASE_FAILED`		
Description	Reference to the DemEventParameter which shall be issued when the error "Flash erase failed (HW)" has occurred. *Note: This parameter is not applicable and made non-editable. This configuration parameter is not used in the code but it is listed for AUTOSAR compatibility.*		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.3 FLS_E_READ_FAILED

Table 31 Specification for FLS_E_READ_FAILED

Name	`FLS_E_READ_FAILED`

(table continues...)

Table 31 (continued) Specification for FLS_E_READ_FAILED

Description	Reference to the DemEventParameter which shall be issued when the error "Flash read failed (HW)" has occurred. *Note: This parameter is not applicable and made non-editable. This configuration parameter is not used in the code but it is listed for AUTOSAR compatibility.*		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.4 FLS_E_UNEXPECTED_FLASH_ID

Table 32 Specification for FLS_E_UNEXPECTED_FLASH_ID

Name	`FLS_E_UNEXPECTED_FLASH_ID`		
Description	Reference to the DemEventParameter which shall be issued when the error "Expected hardware ID not matched" has occurred. *Note: This parameter is not applicable and made non-editable. This configuration parameter is not used in the code but it is listed for AUTOSAR compatibility.*		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.5 FLS_E_WRITE_FAILED

Table 33 Specification for FLS_E_WRITE_FAILED

Name	FLS_E_WRITE_FAILED		
Description	Reference to the DemEventParameter which shall be issued when the error "Flash write failed (HW)" has occurred. *Note: This container is not applicable and made non-editable. This configuration container is not used in the code but it is listed for AUTOSAR compatibility.*		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node:		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.5 Container: FlsExternalDriver

This container is present for external Flash drivers only. Internal Flash drivers do not use the parameter listed in this container, hence its multiplicity is 0 for internal drivers.

This container is not supported since FLS is an internal flash driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.5.1 FlsSpiReference

Table 34 Specification for FlsSpiReference

Name	FlsSpiReference		
Description	Reference to SPI sequence (required for external Flash drivers). This is not supported as external drivers are not supported.		
Multiplicity	1..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: SpiSequence		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE

(table continues...)

Table 34 (continued) Specification for FlsSpiReference

Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6 Container: FlsGeneral

This container holds the for general parameters of the FLS driver. These parameters are always pre-compile.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.6.1 FlsAcLoadOnJobStart

Table 35 Specification for FlsAcLoadOnJobStart

Name	`FlsAcLoadOnJobStart`		
Description	If this parameter is enabled, then the erase access code is loaded in the RAM during Fls_17_Dmu_Erase() API call and unloaded after the completion or cancellation of the job. Similarly, the write access code is loaded in the RAM during the Fls_17_Dmu_Write() API call and unloaded after the completion or cancellation of the job. If this parameter is disabled, then the write and erase access code of the FLS driver are executed from the program flash. This parameter shall be non-editable. The FLS driver access code executes from program flash(PFlash). In TC3xx, Pflash and Dflash can be read in parallel and hence there is no need to load Dflash access code into RAM.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.2 FlsBaseAddress

Table 36 Specification for FlsBaseAddress

Name	`FlsBaseAddress`		
Description	The Flash memory start address (also see SWS_Fls_00208 and SWS_Fls_00209). This parameter defines the lower boundary for the read / write / erase/compare and blank check jobs. This parameter is fixed and not editable.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	Based on the target device -		
Default value	0xAF000000		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.3 FlsBlankCheckApi

Table 37 Specification for FlsBlankCheckApi

Name	`FlsBlankCheckApi`		
Description	This parameter is used to enable/disable the Fls_17_Dmu_BlankCheck() API. The default value is set as FALSE for the optional features to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.4 FlsCancelApi

Table 38 Specification for FlsCancelApi

Name	`FlsCancelApi`		
Description	This parameter is used to enable/disable the Fls_17_Dmu_Cancel() API. The default value is set as FALSE for the optional features to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.5 FlsCompareApi

Table 39 Specification for FlsCompareApi

Name	`FlsCompareApi`		
Description	This parameter is used to enable/disable the Fls_17_Dmu_Compare() API. The default value is set as FALSE for the optional features to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		

(table continues...)

Table 39	(continued) Specification for FlsCompareApi
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.6.6 FlsDevErrorDetect

Table 40 Specification for FlsDevErrorDetect

Name	`FlsDevErrorDetect`		
Description	Parameter enables or disables the Default Error Tracer (DET) detection and reporting. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.7 FlsDriverIndex

Table 41 Specification for FlsDriverIndex

Name	`FlsDriverIndex`		
Description	This parameter is used to assign an index to the FLS driver. The default value is set to minimum.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 254		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.8 FlsEccErrorInfoApi

Table 42 Specification for FlsEccErrorInfoApi

Name	`FlsEccErrorInfoApi`		
Description	This parameter is used to enable or disable the service/API to get the page address of the most recent ECC error that occurred. TRUE: Service to get ECC error information is available. FALSE: Service to get ECC error information is not available. The default value is set as FALSE for the optional features to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	None	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsIfxFeeUse		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.9 FlsEcucPartitionRef

Table 43 Specification for FlsEcucPartitionRef

Name	`FlsEcucPartitionRef`		
Description	Parameter maps the Flash driver to zero or one ECUC partition to make the driver API available in this partition. *Note: Parameter support is added only for AUTOSAR schema compliance. This parameter is not used in code generation logic, hence this parameter is made editable false.*		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: EcucPartition		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	ECU

(table continues...)

Table 43 (continued) Specification for FlsEcucPartitionRef

Dependency	-
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.1.6.10 FlsEraseVerificationEnabled

Table 44 Specification for FlsEraseVerificationEnabled

Name	`FlsEraseVerificationEnabled`		
Description	Compile switch to enable erase verification TRUE: memory region is checked to be erased FALSE: memory region is not checked to be erased		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.6.11 FlsGetJobResultApi

Table 45 Specification for FlsGetJobResultApi

Name	`FlsGetJobResultApi`		
Description	This parameter is used to enable/disable the Fls_17_Dmu_GetJobResult() API. The default value is set as FALSE for the optional features to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 45 (continued) Specification for FlsGetJobResultApi

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.12 FlsGetStatusApi

Table 46 Specification for FlsGetStatusApi

Name	`FlsGetStatusApi`		
Description	This parameter is used to enable/disable the Fls_17_Dmu_GetStatus() API. The default value is set as FALSE for the optional features to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.13 FlsIfxFeeUse

Table 47 Specification for FlsIfxFeeUse

Name	`FlsIfxFeeUse`		
Description	This parameter is used to enable/disable the use of Infineon FEE specific APIs. The default value is set TRUE assuming that FLS driver is used with Infineon FEE.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		

(table continues...)

Table 47 (continued) Specification for FlsIfxFeeUse

Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.14 FlsInitApiMode

Table 48 Specification for FlsInitApiMode

Name	`FlsInitApiMode`		
Description	This parameter is used for configuring the 'User' or 'Supervisor' mode for initialization in the FLS driver. By default access level of all the APIs is set to supervisor so that, there is no dependency on the OS functions to write into the access protected SFRs.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	FLS_17_DMU_MCAL_SUPERVISOR: mode used is SUPERVISOR FLS_17_DMU_MCAL_USER1: operating mode used is USER1		
Default value	FLS_17_DMU_MCAL_SUPERVISOR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsRuntimeApiMode		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.15 FlsInitCheckApi

Table 49 Specification for FlsInitCheckApi

Name	`FlsInitCheckApi`
Description	Switch to enable the safety check for initialization using Fls_17_Dmu_InitCheck() API. The default value is set to FALSE for the optional features to minimize the executable code size.

(table continues...)

Table 49 (continued) Specification for FlsInitCheckApi

Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.16 FlsRunTimeErrorDetect

Table 50 Specification for FlsRunTimeErrorDetect

Name	`FlsRunTimeErrorDetect`		
Description	The activation of the runtime errors is configurable (ON / OFF) at the pre-compile time. FlsRunTimeErrorDetect should also be configured to true if FlsSafetyEnable is enabled. The default value is set as TRUE to ensure that the error detection is enabled and relevant issues are handled during product life cycle.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.17 FlsRuntimeApiMode

Table 51 Specification for FlsRuntimeApiMode

Name	`FlsRuntimeApiMode`		
Description	This configuration parameter gives the mode in which the runtime API is used. By default access level of all the APIs is set to supervisor so that, there is no dependency on the OS functions to write into the access protected SFRs.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	FLS_17_DMU_MCAL_SUPERVISOR: The mode used is SUPERVISOR FLS_17_DMU_MCAL_USER1: operating mode used is USER1		
Default value	FLS_17_DMU_MCAL_SUPERVISOR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.18 FlsSafetyEnable

Table 52 Specification for FlsSafetyEnable

Name	`FlsSafetyEnable`		
Description	This parameter is used to enable/disable the safety notifications for the FLS module. The default value is set to TRUE to ensure that the safety issues are addressed.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.19 FlsSetModeApi

Table 53 Specification for FlsSetModeApi

Name	`FlsSetModeApi`		
Description	This parameter is used to enable/disable the Fls_17_Dmu_SetMode() API. The default value is set to FALSE for the optional feature to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.20 FlsTimeoutSupervisionEnabled

Table 54 Specification for FlsTimeoutSupervisionEnabled

Name	`FlsTimeoutSupervisionEnabled`		
Description	Compile switch to enable/disable timeout supervision TRUE: timeout supervision for erase and write jobs is enabled FALSE: timeout supervision for erase and write jobs is disabled		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.6.21 FlsTotalSize

Table 55 Specification for FlsTotalSize

Name	`FlsTotalSize`		
Description	This parameter is used to calculate the upper boundary for the read/write/erase/compare and blank check jobs. Entire DFlash 0 area is used only by FEE because the unused area (other than the area used for EEPROM emulation) in DFLASH0 data flash becomes unusable for any other purpose as it would incur too many disturbs from the cycled EEPROM area. If the FEE operates in the double sector mode only, then the minimum size should be 8kb, because the logical sector size of the DFLASH0 data flash is 4kb and FEE needs minimum 2 sectors (double sector algorithm), the minimum value that can be configured for this configuration parameter is limited to 8kb. If the FEE operates in quasi only mode then the minimum size would be 4kb as per the minimum logical block size of the DFLASH0 data flash. If the FEE operates in both double sector and quasi state, the minimum value to be used for quasi would be 4kb and the remaining would be divided into 2 sectors of equal size. Similarly, if the minimum size for double sector(8kb) is used, when both double sector and quasi has to operate, then the remaining area could be used for quasi, in multiples of 4kb. While configuring this parameter, user has to take care of the total DFLASH0 size available on a variant.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	4096 - 1048576		
Default value	Based on Target Device		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.22 FlsUseInterrupts

Table 56 Specification for FlsUseInterrupts

Name	`FlsUseInterrupts`
Description	Job processing triggered by hardware interrupt. True: Job processing triggered by interrupt (hardware controlled) False: Job processing not triggered by interrupt (software controlled) This parameter is non-editable and set to false when Infineon FEE is used.

(table continues...)

Table 56 (continued) Specification for FlsUseInterrupts

Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.23 FlsVersionInfoApi

Table 57 Specification for FlsVersionInfoApi

Name	`FlsVersionInfoApi`		
Description	This parameter is used to enable/disable the Fls_17_Dmu_GetVersionInfo() API.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.24 FlsWriteVerificationEnabled

Table 58 Specification for FlsWriteVerificationEnabled

Name	`FlsWriteVerificationEnabled`

(table continues...)

Table 58 (continued) Specification for FlsWriteVerificationEnabled

Description	Compile switch to enable/disable write verification. TRUE: written data is compared directly after write FALSE: written data is not compared directly after write		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.7 Container: FlsIfxSpecificConfig

This container lists all the Infineon specific pre-compile configuration parameters
Post-Build Variant Multiplicity: -
Multiplicity Configuration Class: -

1.3.1.7.1 FlsEraseSuspendTimeout

Table 59 Specification for FlsEraseSuspendTimeout

Name	`FlsEraseSuspendTimeout`		
Description	Timeout parameter for the erase suspend feature (number of loops).		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	12000 - 65535		
Default value	12000		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsUseEraseSuspend		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.2 FlsIllegalStateNotification

Table 60 Specification for FlsIllegalStateNotification

Name	`FlsIllegalStateNotification`		
Description	This parameter is a pointer to a notification function, which is called when the FLS driver reaches an illegal state. The illegal state here signifies that the FLS driver is not able to proceed. No more FLS request is triggered. In such a case, system reset is recommended.		
Multiplicity	1..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.3 FlsStateVarStruct

Table 61 Specification for FlsStateVarStruct

Name	`FlsStateVarStruct`		
Description	This parameter is used to provide the name of the structure containing the entire global variables specific to the Flash driver.		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	FlsStateVar		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.4 FlsUseEraseSuspend

Table 62 Specification for FlsUseEraseSuspend

Name	`FlsUseEraseSuspend`		
Description	Compile switch to enable or disable the FLS erase suspend and erase resume features. STD_ON : FLS suspend/resume feature for erase is enabled STD_OFF: FLS suspend/resume feature for erase is disabled		
Multiplicity	0..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Post-Build
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8 Container: FlsPublishedInformation

Additional published parameters not covered by CommonPublishedInformation container.

Note that these parameters do not have any configuration class setting, because they are published information.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.8.1 FlsAcLocationErase

Table 63 Specification for FlsAcLocationErase

Name	`FlsAcLocationErase`		
Description	Position in RAM, to which the erase flash access code has to be loaded. Only relevant if the erase flash access code is not position independent. If this information is not provided it is assumed that the erase flash access code is position independent and that therefore the RAM position can be freely configured. This parameter is not applicable as the flash driver access code executes from program flash. Therefore, this parameter is not supported.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		

(table continues...)

Table 63 (continued) Specification for FlsAcLocationErase

Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.2 FlsAcLocationWrite

Table 64 Specification for FlsAcLocationWrite

Name	`FlsAcLocationWrite`		
Description	Position in RAM, to which the write flash access code has to be loaded. Only relevant if the write flash access code is not position independent. If this information is not provided it is assumed that the write flash access code is position independent and that therefore the RAM position can be freely configured. This parameter is not relevant as flash driver access code executes from program flash. Therefore, this parameter is not supported.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.3 FlsAcSizeErase

Table 65 Specification for FlsAcSizeErase

Name	`FlsAcSizeErase`
Description	Number of bytes in the RAM needed for the erase Flash access code. This parameter is not relevant as the flash driver access code executes from program flash. Therefore, this is not supported.

(table continues...)

Table 65 (continued) Specification for FlsAcSizeErase

Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.4 FlsAcSizeWrite

Table 66 Specification for FlsAcSizeWrite

Name	`FlsAcSizeWrite`		
Description	Number of bytes in the RAM needed for the write Flash access code. This parameter is not relevant as the Flash access code executes from program flash. Therefore, this is not supported.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.5 FlsEraseTime

Table 67 Specification for FlsEraseTime

Name	`FlsEraseTime`

(table continues...)

Table 67 (continued) Specification for FlsEraseTime

Description	Maximum time to erase one logical sector in microseconds. The default value is given as per datasheet considering simultaneous access of DF0 and DF1 from TriCore and HSM respectively. Tolerance of 10% is considered additionally on the actual erase time required for one sector (1.5 seconds). Erase time: The actual erase time is 1.5s and additional tolerance of 10% is considered. Also considering simultaneous access of DF0 by Fls and DF1 by HSM Fls, the timeout value is increased by 15% Erase time = 1.5s + 0.15s(10% of 1.5s) + 0.2475 (15 % of total erase time, 1.65s) = 1897500us		
Multiplicity	1..1	**Type**	EcucFloatParamDef
Range	0.0 - 4294967295.0		
Default value	1897500		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.6 FlsErasedValue

Table 68 Specification for FlsErasedValue

Name	`FlsErasedValue`		
Description	The contents of an erased Flash memory cell. The default value is selected as 0 as this is the value on DFLASH0 after erase.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.7 FlsExpectedHwId

Table 69 Specification for FlsExpectedHwId

Name	`FlsExpectedHwId`		
Description	Unique identifier of the hardware device that is expected by the driver (the device for which the driver has been implemented). This parameter is not used as it is applicable only for external flash drivers.		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.8 FlsSpecifiedEraseCycles

Table 70 Specification for FlsSpecifiedEraseCycles

Name	`FlsSpecifiedEraseCycles`		
Description	Number of erase cycles specified for the Flash device (usually given in the device data sheet). The default value is selected based on the datasheet.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	125000		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.9 FlsWriteTime

Table 71 Specification for FlsWriteTime

Name	`FlsWriteTime`		
Description	Maximum time for one write operation, in microseconds, that is, burst write (32 bytes). The default value has been given based on the target parameter of the hardware DFLASH0 for burst write considering simultaneous access of DF0 and DF1. Tolerance of 10% is considered additionally on the actual write time required for one burst operation (140 microseconds). Write time: The actual write time is 140us and additional tolerance of 10% is considered. Additional 5ms is considered for simultaneous access of DF0 by Fls and DF1 by HSM Fls. Write time = 140us + 14us (10% of 140us) + 5000us (Additional 5ms) = 5154us		
Multiplicity	1..1	**Type**	EcucFloatParamDef
Range	0.0 - 4294967295.0		
Default value	5154		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9 Container: FlsSector

This container contains configuration description of a flashable sector.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: -

1.3.1.9.1 FlsNumberOfSectors

Table 72 Specification for FlsNumberOfSectors

Name	`FlsNumberOfSectors`		
Description	Number of continuous sectors with identical values for FlsSectorSize and FlsPageSize(in bytes). The FlsSectorStartAddress parameter denotes the start address of the first sector. The maximum and the default value for this parameter is '2' as it is used with the double sector algorithm.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef

(table continues...)

Table 72 (continued) Specification for FlsNumberOfSectors

Range	1 - 2		
Default value	2		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.2 FlsPageSize

Table 73 Specification for FlsPageSize

Name	`FlsPageSize`		
Description	Size of one FLS age in bytes. This parameter is fixed, therefore, not configurable.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	8 - 8		
Default value	8		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.3 FlsSectorSize

Table 74 Specification for FlsSectorSize

Name	`FlsSectorSize`

(table continues...)

Table 74 (continued) Specification for FlsSectorSize

Description	Size of the FLS Sector (in bytes). For double sector data, this parameter will of the size of one of the sectors. If no quasi-static data is used, then the value of this parameter will typically be half of FlsTotalsize and should be in the multiple of 4 Kbytes. For quasi-static data this contains the quasi region and should be in the multiple of 4K bytes. If both double sector and quasi-static data are used then two containers should be used to specify the sector size appropriately such that the total size is justified. For example, the minimum size for quasi would be 4 kb and the rest could be dedicated for using the double sector algorithm. For more details, refer to FlsNumberOfSectors.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	4096 - 1048576		
Default value	DFLASH0 total size divided by 2 (DFLASH0 total size varies)		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	FlsTotalSize		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.4 FlsSectorStartaddress

Table 75 Specification for FlsSectorStartaddress

Name	`FlsSectorStartaddress`		
Description	Start address offset of the DFlash0 sector from the configured Flash base address to access a certain Flash memory area. FLS base address is always added to this address to arrive at the correct address.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 1044480		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	FlsTotalSize		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10 Container: FlsSectorList

List of flashable sectors and pages.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.2 Functions - Type definitions

1.3.2.1 Fls_17_Dmu_AddressType

Table 76 Specification for Fls_17_Dmu_AddressType

Syntax	`Fls_17_Dmu_AddressType`	
Type	uint32	
File	`Fls_17_Dmu.h`	
Range	0 – 4294967295	Size depends on target platform and DFLASH0 data flash memory on the flash device.
Description	Used as an address offset from the configured Flash base address to access a certain Flash memory area. The Fls_17_Dmu_AddressType type has the lower limit as 0 and the FLS base address is always added to it to arrive at the correct address.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.2 Fls_17_Dmu_ConfigType

Table 77 Specification for Fls_17_Dmu_ConfigType

Syntax	`Fls_17_Dmu_ConfigType`	
Type	Structure	
File	`Fls_17_Dmu.h`	
Range	HW dependent structure	Structure to hold the Flash driver configuration set. The contents of the initialization data structure are specific to the Flash memory hardware.
Description	A pointer to such a structure is provided to the Flash driver initialization routine for configuration of the driver and Flash memory hardware.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.3 Fls_17_Dmu_HardenType

Table 78 Specification for Fls_17_Dmu_HardenType

Syntax	`Fls_17_Dmu_HardenType`	
Type	uint8	
File	`Fls_17_Dmu.h`	
Range	0 - FLS_17_DMU_HARDENCHK_NOTREQD	Hardening not required
	2 - FLS_17_DMU_HARDENCHK_ERROR	Hardening failed due to some error.
	1 - FLS_17_DMU_HARDENCHK_REQRD	Hardening required
Description	Used to specify the hardening update (whether hardening is required or not or any failure occurred during the hardening check).	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.4 Fls_17_Dmu_Job_Type

Table 79 Specification for Fls_17_Dmu_Job_Type

Syntax	`Fls_17_Dmu_Job_Type`	
Type	uint8	
File	`Fls_17_Dmu.h`	
Range	0 - FLS_NO_JOB	No notification was called
	1 - FLS_WRITE_JOB	Notification for the write job
	2 - FLS_ERASE_JOB	Notification for the erase job
	3 - FLS_READ_JOB	Notification for the read job
	4 - FLS_COMPARE_JOB	Notification for the compare job
	6 - FLS_CANCEL_JOB	Notification for the canceled job
	9 - FLS_BLANKCHECK_JOB	Notification for the blank check
Description	Specifies the type of job for which the notification was called.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.5 Fls_17_Dmu_LengthType

Table 80 Specification for Fls_17_Dmu_LengthType

Syntax	`Fls_17_Dmu_LengthType`
Type	uint32
File	`Fls_17_Dmu.h`

(table continues...)

Table 80 (continued) Specification for Fls_17_Dmu_LengthType

Range	0 – 4294967295	Should be the same type as Fls_AddressType because of arithmetic operations. Size depends on the target platform and the DFLASH0 data flash memory on the device.
Description	Specifies the number of bytes to read/write/erase/compare.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.6 Fls_17_Dmu_NotifFunctionPtrType

Table 81 Specification for Fls_17_Dmu_NotifFunctionPtrType

Syntax	`Fls_17_Dmu_NotifFunctionPtrType`
Type	Pointer to a function of type void Function_Name (void)
File	`Fls_17_Dmu.h`
Description	Function pointer type for callback functions. Used for job end, job error and illegal functions.
Source	IFX
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3 Functions - APIs

This section lists all the APIs of the FLS driver.

1.3.3.1 Fls_17_Dmu_BlankCheck

Table 82 Specification for `Fls_17_Dmu_BlankCheck` API

Syntax	`Std_ReturnType  Fls_17_Dmu_BlankCheck`
	`(`
	`  const Fls_17_Dmu_AddressType TargetAddress,`
	`  const Fls_17_Dmu_LengthType Length`
	`)`

Service ID	0x0A
Sync/Async	Asynchronous
Safety Level	Refer to the release notes for the safety related info
Re-entrancy	Non reentrant

(table continues...)

Table 82 (continued) Specification for `Fls_17_Dmu_BlankCheck` API

Parameters (in)	TargetAddress Length	Address in the DFlash0 data flash memory from which the blank check should be started. Min.: 0 Max.: FLS_17_DMU_TOTAL_SIZE - 1 Number of bytes to be checked for erase pattern. Min.: 1 Max.: FLS_17_DMU_TOTAL_SIZE - TargetAddress
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: request for blank checking has been accepted by the module E_NOT_OK: request for blank checking has not been accepted by the module
Description	The Fls_17_Dmu_BlankCheck should verify, whether a given memory area has been erased but not (yet) programmed. The function should limit the maximum number of checked Flash cells per main function cycle to the configured value FlsMaxReadNormalMode or FlsMaxReadFastMode, respectively.	
Source	AUTOSAR	
Error handling	FLS_17_DMU_SE_HW_BUSY, FLS_17_DMU_E_UNINIT, FLS_17_DMU_E_PARAM_LENGTH, FLS_17_DMU_E_PARAM_ADDRESS, FLS_17_DMU_E_BUSY	
Configuration dependencies	FlsBlankCheckApi	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.2 Fls_17_Dmu_Cancel

Table 83 Specification for `Fls_17_Dmu_Cancel` API

Syntax	`void Fls_17_Dmu_Cancel ( void )`
Service ID	0x03
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info
Re-entrancy	Non reentrant

(table continues...)

Table 83 (continued) Specification for `Fls_17_Dmu_Cancel` API

Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	void
Description	Cancels an ongoing job. *Note: Fls_17_Dmu_Cancel() shall not be invoked from interrupt context.*	
Source	AUTOSAR	
Error handling	FLS_17_DMU_E_UNINIT	
Configuration dependencies	FlsCancelApi	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.3 Fls_17_Dmu_CancelNonEraseJobs

Table 84 Specification for `Fls_17_Dmu_CancelNonEraseJobs` API

Syntax	`void Fls_17_Dmu_CancelNonEraseJobs ( void )`	
Service ID	0x23	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	void

(table continues...)

Table 84 (continued) Specification for `Fls_17_Dmu_CancelNonEraseJobs` API

Description	This is an Infineon specific API and not listed in the SWS. Service for canceling the ongoing flash jobs except the erase job. This function aborts the pending jobs (except the erase job), so that directly after returning from this function, a new job can be accepted by the driver. The function resets the internal job processing variables of the driver(such as address, length and data pointer) and sets the driver state to idle. The routine sets the job result to MEMIF_JOB_CANCELED, if the job result currently has the following value: MEMIF_JOB_PENDING. Otherwise, it leaves the job result unchanged.
Source	IFX
Error handling	-
Configuration dependencies	FlsIfxFeeUse
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.4 Fls_17_Dmu_Compare

Table 85 Specification for `Fls_17_Dmu_Compare` API

Syntax	`Std_ReturnType  Fls_17_Dmu_Compare`
	`(`
	`  const Fls_17_Dmu_AddressType SourceAddress,`
	`  const uint8 * const TargetAddressPtr,`
	`  const Fls_17_Dmu_LengthType Length`
	`)`	

Service ID	0x08	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non reentrant	
Parameters (in)	SourceAddress TargetAddressPtr Length	Source address in the DFLASH0 data flash memory. This address offset is added to the data flash memory base address. Min.: 0 Max.: FLS_17_DMU_TOTAL_SIZE - 1 Pointer to the target data buffer Number of bytes to compare Min.: 1 Max.: FLS_17_DMU_TOTAL_SIZE - SourceAddress
Parameters (out)	-	-

(table continues...)

Table 85 (continued) Specification for `Fls_17_Dmu_Compare` API

Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: compare command is accepted E_NOT_OK: compare command is not accepted
Description	Compares the contents of an area of the DFLASH0 data flash memory with that of an application data buffer.	
Source	AUTOSAR	
Error handling	FLS_17_DMU_SE_HW_BUSY, FLS_17_DMU_E_BUSY, FLS_17_DMU_E_UNINIT, FLS_17_DMU_E_PARAM_DATA, FLS_17_DMU_E_PARAM_LENGTH, FLS_17_DMU_E_PARAM_ADDRESS	
Configuration dependencies	FlsCompareApi	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.5 Fls_17_Dmu_CompareWordsSync

Table 86 Specification for `Fls_17_Dmu_CompareWordsSync` API

Syntax	`Std_ReturnType  Fls_17_Dmu_CompareWordsSync`
	`(`
	`  const Fls_17_Dmu_AddressType SourceAddress,`
	`  const uint32 * const TargetAddressPtr,`
	`  const uint32 Length`
	`)`	

Service ID	0x22	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non reentrant	
Parameters (in)	SourceAddress TargetAddressPtr Length	Source address in the DFLASH0 data flash memory. This address offset is added to the DFLASH0 data flash memory base address. Pointer to the target data buffer. Number of words to be compared. It takes the value from 1 to DLASH0 total size.
Parameters (out)	-	-
Parameters (in - out)	-	-

(table continues...)

Table 86 (continued) Specification for `Fls_17_Dmu_CompareWordsSync` API

Return	Std_ReturnType	E_OK: compare is successful E_NOT_OK: compare is not successful
Description	This is an IFX specific API and not listed in the SWS. It is a service for comparing the contents on the DFLASH0 data flash memory synchronously. *Note: The range check is performed only when 'FlsSafetyEnable' is enabled.*	
Source	IFX	
Error handling	FLS_17_DMU_SE_BUSY, FLS_17_DMU_SE_PARAM_DATA, FLS_17_DMU_SE_PARAM_ADDRESS, FLS_17_DMU_E_COMPARE_FAILED, FLS_17_DMU_SE_HW_BUSY, FLS_17_DMU_SE_PARAM_LENGTH	
Configuration dependencies	FlsIfxFeeUse	
User hints	-	
SFR accessed	DMU_HF_ECCC(rw), DMU_HF_ECCS(r), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.6 Fls_17_Dmu_Erase

Table 87 Specification for `Fls_17_Dmu_Erase` API

Syntax	`Std_ReturnType  Fls_17_Dmu_Erase`
	`(`
	`  const Fls_17_Dmu_AddressType TargetAddress,`
	`  const Fls_17_Dmu_LengthType Length`
	`)`	

Service ID	0x01	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non reentrant	
Parameters (in)	TargetAddress Length	Target address in the DFLASH0 data flash memory. This address offset is added to the DFLASH0 data flash memory base address. Min.: 0 Max.: FLS_17_DMU_TOTAL_SIZE - 1 Number of bytes to erase Min.: 1 Max.: FLS_17_DMU_TOTAL_SIZE - TargetAddress

(table continues...)

Table 87 (continued) Specification for `Fls_17_Dmu_Erase` API

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: erase command accepted E_NOT_OK: erase command not accepted
Description	This API is a service for erasing one or more complete Flash sectors.	
Source	AUTOSAR	
Error handling	FLS_17_DMU_E_ERASE_FAILED, FLS_17_DMU_SE_ILLGL_OPERTN, FLS_17_DMU_E_UNINIT, FLS_17_DMU_SE_HW_BUSY, FLS_17_DMU_E_BUSY, FLS_17_DMU_E_PARAM_LENGTH, FLS_17_DMU_E_PARAM_ADDRESS	
Configuration dependencies	-	
User hints	-	
SFR accessed	DMU_HF_ERRSR(r), DMU_HF_OPERATION(r), DMU_HF_SUSPEND(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.7 Fls_17_Dmu_GetEccErrorPageAddress

Table 88 Specification for `Fls_17_Dmu_GetEccErrorPageAddress` API

Syntax	`Std_ReturnType  Fls_17_Dmu_GetEccErrorPageAddress`
	`(`
	`  uint32 * const PageAddressPtr`
	`)`	

Service ID	0x2E	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	PageAddressPtr	Pointer to store the page address where the last ECC error occurred
Parameters (in - out)	-	-

(table continues...)

Table 88 (continued) Specification for `Fls_17_Dmu_GetEccErrorPageAddress` API

Return	Std_ReturnType	E_OK: The requested job has been executed. E_NOT_OK: The requested job has not been executed due to any of the following reasons: - Driver is busy - Passed pointer is null
Description	This is an IFX specific API and not listed in the SWS. Service to get the address of the flash page where last Multi-bit ECC error is detected. If there is no Multi-bit ECC error detected in the current power cycle, then the value of PageAddressPtr parameter will be 0xFFFFFFFF. *Note: The PageAddress of the last occurred ECC error is not maintained across power cycles.*	
Source	IFX	
Error handling	FLS_17_DMU_SE_BUSY, FLS_17_DMU_SE_PARAM_DATA	
Configuration dependencies	FlsIfxFeeUse,FlsEccErrorInfoApi	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.8 Fls_17_Dmu_GetJobResult

Table 89 Specification for `Fls_17_Dmu_GetJobResult` API

Syntax	`MemIf_JobResultType  Fls_17_Dmu_GetJobResult`
	`(`
	`  void`
	`)`	

Service ID	0x05	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	MemIf_JobResultType	The result of the last job

(table continues...)

Table 89 (continued) Specification for `Fls_17_Dmu_GetJobResult` API

Description	Returns the result of the last job. *Note: When the Infineon FEE is present, for the Fls_17_Dmu_CompareWordsSync(), Fls_17_Dmu_ReadWordsSync(), Fls_17_Dmu_VerifyErase() and Fls_17_dmu_verifySectorErase() APIs, the job result is not updated. Therefore, the job result returned for the mentioned APIs are of the previous jobs.*
Source	AUTOSAR
Error handling	FLS_17_DMU_E_UNINIT
Configuration dependencies	FlsGetJobResultApi
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.9 Fls_17_Dmu_GetNotifCaller

Table 90 Specification for `Fls_17_Dmu_GetNotifCaller` API

Syntax	`Fls_17_Dmu_Job_Type  Fls_17_Dmu_GetNotifCaller`
	`(`
	`  void`
	`)`	

Service ID	0x29	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Fls_17_Dmu_Job_Type	FLS job that raised the notification
Description	Returns the FLS job that raised the notification. It should be called only from the callback notification functions of the upper layers. This is an Infineon specific API and not listed in the SWS.	
Source	IFX	
Error handling	-	
Configuration dependencies	FlsIfxFeeUse	

(table continues...)

Table 90 (continued) Specification for `Fls_17_Dmu_GetNotifCaller` API	
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.10 Fls_17_Dmu_GetOperStatus

Table 91 Specification for `Fls_17_Dmu_GetOperStatus` API		
Syntax	`Std_ReturnType Fls_17_Dmu_GetOperStatus ( void )`	
Service ID	0x26	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: no OPER error E_NOT_OK: OPER error occurred
Description	This is an Infineon specific API and not listed in the SWS. Returns whether the OPER error had occurred or not.	
Source	IFX	
Error handling	-	
Configuration dependencies	-	
User hints	-	
SFR accessed	DMU_HF_ERRSR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.11 Fls_17_Dmu_GetStatus

Table 92 Specification for `Fls_17_Dmu_GetStatus` API

Syntax	`MemIf_StatusType Fls_17_Dmu_GetStatus ( void )`	
Service ID	0x04	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	MemIf_StatusType	The state of the driver
Description	Returns the driver state.	
Source	AUTOSAR	
Error handling	-	
Configuration dependencies	FlsGetStatusApi	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.12 Fls_17_Dmu_GetVersionInfo

Table 93 Specification for `Fls_17_Dmu_GetVersionInfo` API

Syntax	`void Fls_17_Dmu_GetVersionInfo ( Std_VersionInfoType * const VersionInfoPtr )`	
Service ID	0x10	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-

(table continues...)

Table 93 (continued) Specification for `Fls_17_Dmu_GetVersionInfo` API

Parameters (out)	VersionInfoPtr	Pointer to where to store the version information of this module.
Parameters (in - out)	-	-
Return	void	-
Description	Returns the version information of this module.	
Source	AUTOSAR	
Error handling	FLS_17_DMU_E_PARAM_POINTER	
Configuration dependencies	FlsVersionInfoApi	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.13 Fls_17_Dmu_Init

Table 94 Specification for `Fls_17_Dmu_Init` API

Syntax	`void Fls_17_Dmu_Init ( const Fls_17_Dmu_ConfigType * const ConfigPtr )`	
Service ID	0x00	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non reentrant	
Parameters (in)	ConfigPtr	Pointer to the FLS driver configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	void
Description	Initializes the Flash driver.	
Source	AUTOSAR	
Error handling	FLS_17_DMU_E_PARAM_CONFIG, FLS_17_DMU_E_BUSY, FLS_17_DMU_E_INIT_FAILED, FLS_17_DMU_SE_INIT_FAILED, FLS_17_DMU_SE_HW_BUSY	
Configuration dependencies	-	

(table continues...)

Table 94 (continued) Specification for `Fls_17_Dmu_Init` API

User hints	-
SFR accessed	DMU_HF_CCONTROL(w), DMU_HF_DWAIT(rw), DMU_HF_ECCC(rw), DMU_HF_ECCW(w), DMU_HF_EER(rw), DMU_HF_ERRSR(r), DMU_HF_MARGIN(rw), DMU_HF_PCONTROL(w), DMU_HF_PROCONDF(r), DMU_HF_PROCONUSR(r), DMU_HF_SUSPEND(rw), FSI_COMM_1(w), FSI_COMM_2(w), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.14 Fls_17_Dmu_InitCheck

Table 95 Specification for `Fls_17_Dmu_InitCheck` API

Syntax	`Std_ReturnType  Fls_17_Dmu_InitCheck`
	`(`
	`  const Fls_17_Dmu_ConfigType ConfigPtr`
	`)`	

Service ID	0x2B	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non reentrant	
Parameters (in)	ConfigPtr	None
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: if initialization comparison is success E_NOT_OK: if initialization comparison fails
Description	This API checks the initialization values.	
Source	IFX	
Error handling	-	
Configuration dependencies	FlsInitCheckApi	
User hints	-	

(table continues...)

Table 95 (continued) Specification for `Fls_17_Dmu_InitCheck` API

SFR accessed	DMU_HF_CCONTROL(r), DMU_HF_DWAIT(r), DMU_HF_ECCC(r), DMU_HF_ECCS(r), DMU_HF_ECCW(r), DMU_HF_EER(r), DMU_HF_ERRSR(r), DMU_HF_MARGIN(r), DMU_HF_PCONTROL(r), DMU_HF_PROCONDF(r), DMU_HF_PROCONUSR(r), DMU_HF_SUSPEND(r), FSI_COMM_1(r), FSI_COMM_2(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.15 Fls_17_Dmu_IsHardeningRequired

Table 96 Specification for `Fls_17_Dmu_IsHardeningRequired` API

Syntax	`Fls_17_Dmu_HardenType  Fls_17_Dmu_IsHardeningRequired`
	`(`
	`  const Fls_17_Dmu_AddressType TargetAddress,`
	`  const uint8 AlignChk`
	`)`	

Service ID	0x28	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	TargetAddress AlignChk	Target address in Flash memory. This address offset is added to the Flash memory base address This parameter signifies whether the hardening is to be done at the page level or WL level. The following are the values which will be used for indication: - hardening is done at the page level if the value of this parameter is: FLS_17_DMU_PAGE_HARDEN(0x55) - hardening is done at the 'Word-line level' if the value of this parameter is: FLS_17_DMU_WORDLINE_HARDEN(0xAA)
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Fls_17_Dmu_HardenType	0 - FLS_17_DMU_HARDEN_NOTREQRD: Hardening is not required. 1 - FLS_17_DMU_HARDEN_REQRD: Hardening is required. 2 - FLS_17_DMU_HARDEN_ERROR: Hardening failed due to error.
Description	This is an Infineon specific API and not listed in the SWS. The function checks whether the contents of the DFLASH0 data flash memory at requested Page or WL address need hardening or not.	

(table continues...)

Table 96 (continued) Specification for `Fls_17_Dmu_IsHardeningRequired` API

Source	IFX
Error handling	FLS_17_DMU_E_HARDENCHK_FAIL, FLS_17_DMU_SE_PARAM_INVLD, FLS_17_DMU_SE_HW_BUSY, FLS_17_DMU_SE_PARAM_ADDRESS, FLS_17_DMU_SE_BUSY, FLS_17_DMU_SE_HW_TIMEOUT
Configuration dependencies	FlsIfxFeeUse
User hints	-
SFR accessed	DMU_HF_CLRE(w), DMU_HF_CONTROL(rw), DMU_HF_ERRSR(r), FSI_COMM_1(rw), FSI_COMM_2(r), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.16 Fls_17_Dmu_Read

Table 97 Specification for `Fls_17_Dmu_Read` API

Syntax	`Std_ReturnType Fls_17_Dmu_Read ( const Fls_17_Dmu_AddressType SourceAddress, uint8 * const TargetAddressPtr, const Fls_17_Dmu_LengthType Length )`	
Service ID	0x07	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non reentrant	
Parameters (in)	SourceAddress Length	Source address in the DFlash0 data flash memory. This address offset will be added to the DFlash0 data flash memory base address. Min.: 0 Max.: FLS_17_DMU_TOTAL_SIZE - 1 Number of bytes to read Min.: 1 Max.: FLS_17_DMU_TOTAL_SIZE - SourceAddress
Parameters (out)	TargetAddressPtr	Pointer to the target data buffer
Parameters (in - out)	-	-

(table continues...)

Table 97 (continued) Specification for `Fls_17_Dmu_Read` API

Return	Std_ReturnType	E_OK: read command has been accepted E_NOT_OK: read command has not been accepted
Description	Reads from flash memory.	
Source	AUTOSAR	
Error handling	FLS_17_DMU_E_PARAM_ADDRESS, FLS_17_DMU_E_PARAM_DATA, FLS_17_DMU_E_UNINIT, FLS_17_DMU_E_BUSY, FLS_17_DMU_E_PARAM_LENGTH, FLS_17_DMU_SE_HW_BUSY	
Configuration dependencies	-	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.17 Fls_17_Dmu_ReadWordsSync

Table 98 Specification for `Fls_17_Dmu_ReadWordsSync` API

Syntax	`Std_ReturnType  Fls_17_Dmu_ReadWordsSync`
	`(`
	`  const Fls_17_Dmu_AddressType SourceAddress,`
	`  uint32 * const TargetAddressPtr,`
	`  const uint32 Length`
	`)`	

Service ID	0x21	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	SourceAddress Length	Source address in the DFLASH0 data flash memory. This address offset is added to the DFLASH0 base address. Number of words to be read. It takes the value from 1 to DFLASH0 data flash size.
Parameters (out)	TargetAddressPtr	Pointer to target data buffer
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: read command is accepted E_NOT_OK: read command is not accepted
Description	This is an Infineon specific API and not listed in the SWS. It is a service to read synchronously from the DFLASH0 data flash memory. *Note: The range check is performed for the input parameters only when the 'FlsSafetyEnable' configuration parameter is enabled.*	

(table continues...)

Table 98 (continued) Specification for `Fls_17_Dmu_ReadWordsSync` API

Source	IFX
Error handling	FLS_17_DMU_E_READ_FAILED, FLS_17_DMU_SE_PARAM_LENGTH, FLS_17_DMU_SE_BUSY, FLS_17_DMU_SE_PARAM_DATA, FLS_17_DMU_SE_PARAM_ADDRESS, FLS_17_DMU_SE_HW_BUSY
Configuration dependencies	FlsIfxFeeUse
User hints	-
SFR accessed	DMU_HF_ECCC(rw), DMU_HF_ECCS(r), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.18 Fls_17_Dmu_ResumeErase

Table 99 Specification for `Fls_17_Dmu_ResumeErase` API

Syntax	`Std_ReturnType Fls_17_Dmu_ResumeErase ( void )`	
Service ID	0x2A	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Erase resume command was accepted and passed or Erase was not suspended when this API was called E_NOT_OK: Erase resume command was not accepted or failed
Description	This is an IFX specific API and not listed in the SWS. It is a service for resuming a suspended erase of a sector.	
Source	IFX	
Error handling	FLS_17_DMU_E_RESUME_FAIL, FLS_17_DMU_SE_HW_BUSY, FLS_17_DMU_SE_BUSY, FLS_17_DMU_SE_HW_TIMEOUT	

(table continues...)

Table 99 (continued) Specification for `Fls_17_Dmu_ResumeErase` API

Configuration dependencies	FlsUseEraseSuspend
User hints	-
SFR accessed	DMU_HF_CLRE(w), DMU_HF_ERRSR(r), DMU_HF_STATUS(r), DMU_HF_SUSPEND(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.19 Fls_17_Dmu_SetMode

Table 100 Specification for `Fls_17_Dmu_SetMode` API

Syntax	`void Fls_17_Dmu_SetMode ( const MemIf_ModeType Mode )`	
Service ID	0x09	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non reentrant	
Parameters (in)	Mode	MEMIF_MODE_SLOW: slow read access MEMIF_MODE_FAST: fast read access
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	void
Description	Sets the flash operation mode of the driver.	
Source	AUTOSAR	
Error handling	FLS_17_DMU_SE_PARAM_INVLD, FLS_17_DMU_E_BUSY, FLS_17_DMU_E_UNINIT	
Configuration dependencies	FlsSetModeApi	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.20 Fls_17_Dmu_SuspendErase

Table 101 Specification for `Fls_17_Dmu_SuspendErase` API

Syntax	`Std_ReturnType Fls_17_Dmu_SuspendErase ( void )`	
Service ID	0x25	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Erase suspend successful or erase is already suspended E_NOT_OK: erase suspend failed or this API is called when erase is not on-going
Description	This is an IFX specific API and not listed in the SWS. It is a service for suspending an ongoing erase of a sector.	
Source	IFX	
Error handling	FLS_17_DMU_SE_SUSPNDERASE_FAIL, FLS_17_DMU_SE_HW_TIMEOUT	
Configuration dependencies	FlsUseEraseSuspend	
User hints	-	
SFR accessed	DMU_HF_STATUS(r), DMU_HF_SUSPEND(rw) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.21 Fls_17_Dmu_VerifyErase

Table 102 Specification for `Fls_17_Dmu_VerifyErase` API

Syntax	`Std_ReturnType Fls_17_Dmu_VerifyErase ( const Fls_17_Dmu_AddressType TargetAddress, uint32 * const UnerasedWordlineAddressPtr, uint8 * const UnerasedWordlineCountPtr )`	
Service ID	0x24	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	TargetAddress	Target offset address in the DFLASH0 data flash memory. This address offset is added to the DFLASH0 data flash memory base address. The input value for TargetAddress can only be the start address of either of the sectors used by the Infineon FEE double sector algorithm.
Parameters (out)	UnerasedWordlineAddressPtr UnerasedWordlineCountPtr	Pointer to the first un-eraseable WL address. Pointer to the un-eraseable WL count.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Erase verification command was accepted and passed E_NOT_OK: Erase verification command was not accepted or failed with more than two un-erasable WL
Description	This is an Infineon specific API and not listed in the SWS. It is a synchronous service to verify the erase operation performed on one of the two sectors(as per double sector algorithm used by Infineon FEE).	
Source	IFX	
Error handling	FLS_17_DMU_SE_HW_BUSY, FLS_17_DMU_SE_PARAM_DATA, FLS_17_DMU_SE_BUSY, FLS_17_DMU_SE_PARAM_ADDRESS	
Configuration dependencies	FlsIfxFeeUse	
User hints	-	
SFR accessed	DMU_HF_ECCC(rw), DMU_HF_MARGIN(rw), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.22 Fls_17_Dmu_VerifySectorErase

Table 103 Specification for `Fls_17_Dmu_VerifySectorErase` API

Syntax	`Std_ReturnType Fls_17_Dmu_VerifySectorErase`
	`(`
	`  const Fls_17_Dmu_AddressType TargetAddress,`
	`  uint32 * const UnerasedWordlineAddressPtr,`
	`  uint8 * const UnerasedWordlineCountPtr,`
	`  const uint8 Sector`
	`)`	

Service ID	0x2C	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	TargetAddress Sector	Target offset address in the DFLASH0 data flash memory. This parameter is an address offset and is added to the DFLASH0 data flash memory base address.The TargetAddress can only be the value of the start address of either of the sectors of the Infineon FEE double sector algorithm. Logical sub sector number (of the corresponding NVM sector) to be verified
Parameters (out)	UnerasedWordlineAddressPtr UnerasedWordlineCountPtr	Pointer to the first un-erased WL address. Pointer to the un-erased WL count.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Erase verification operation is accepted and passed. E_NOT_OK: Erase verification operation is not accepted or failed with more than two un-erasable WLs.
Description	This is an Infineon specific API and not listed in the SWS. It is a service for verifying the erase of a logical sub sector synchronously.	
Source	IFX	
Error handling	FLS_17_DMU_SE_PARAM_DATA, FLS_17_DMU_SE_BUSY, FLS_17_DMU_SE_PARAM_INVLD, FLS_17_DMU_SE_HW_BUSY, FLS_17_DMU_SE_PARAM_ADDRESS	
Configuration dependencies	FlsIfxFeeUse	
User hints	-	
SFR accessed	DMU_HF_ECCC(rw), DMU_HF_MARGIN(rw), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	

(table continues...)

Table 103 (continued) Specification for `Fls_17_Dmu_VerifySectorErase` API

Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.23 Fls_17_Dmu_Write

Table 104 Specification for `Fls_17_Dmu_Write` API

Syntax	`Std_ReturnType Fls_17_Dmu_Write ( const Fls_17_Dmu_AddressType TargetAddress, const uint8 * const SourceAddressPtr, const Fls_17_Dmu_LengthType Length )`	
Service ID	0x02	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non reentrant	
Parameters (in)	TargetAddress SourceAddressPtr Length	Target address in the DFlash0 hardware memory. This address offset is be added to the DFlash0 base address. Min.: 0 Max.: FLS_17_DMU_TOTAL_SIZE - 1 Pointer to the source data buffer. Number of bytes to write Min.: 1 Max.: FLS_17_DMU_TOTAL_SIZE - TargetAddress
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: write operation accepted E_NOT_OK: write operation not accepted
Description	Writes one or more complete flash pages.	
Source	AUTOSAR	
Error handling	FLS_17_DMU_E_PARAM_DATA, FLS_17_DMU_E_BUSY, FLS_17_DMU_E_PARAM_ADDRESS, FLS_17_DMU_SE_HW_BUSY, FLS_17_DMU_E_VERIFY_ERASE_FAILED, FLS_17_DMU_E_UNINIT, FLS_17_DMU_E_WRITE_FAILED, FLS_17_DMU_E_PARAM_LENGTH	
Configuration dependencies	-	
User hints	-	

(table continues...)

Table 104 (continued) Specification for `Fls_17_Dmu_Write` API

SFR accessed	DMU_HF_ECCC(rw), DMU_HF_ECCS(r), DMU_HF_ERRSR(r), DMU_HF_OPERATION(r), DMU_HF_STATUS(r), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.4 Notifications and Callbacks

The FLS driver does not provide any notification or callbacks.

1.3.5 Scheduled functions

This section lists all the scheduled functions of the FLS driver.

1.3.5.1 Fls_17_Dmu_MainFunction

Table 105 Specification for `Fls_17_Dmu_MainFunction` API

Syntax	`void Fls_17_Dmu_MainFunction ( void )`	
Service ID	0x06	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This API is a service for performing the read, write, erase, compare and blank check jobs on the DFLASH0 hardware.	
Source	AUTOSAR	
Error handling	FLS_17_DMU_E_ERASE_FAILED, FLS_17_DMU_E_READ_FAILED, FLS_17_DMU_E_COMPARE_FAILED, FLS_17_DMU_E_UNINIT, FLS_17_DMU_E_TIMEOUT, FLS_17_DMU_E_VERIFY_WRITE_FAILED, FLS_17_DMU_E_VERIFY_ERASE_FAILED, FLS_17_DMU_E_WRITE_FAILED, FLS_17_DMU_E_BLANKCHECK_FAILED	

(table continues...)

Table 105 (continued) Specification for `Fls_17_Dmu_MainFunction` API	
Configuration dependencies	-
User hints	-
SFR accessed	DMU_HF_CLRE(w), DMU_HF_ECCC(rw), DMU_HF_ECCS(r), DMU_HF_ERRSR(rw), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.6 Interrupt service routines

This section lists all the interrupt handlers of the FLS driver.

1.3.6.1 Fls_17_Dmu_Isr

Table 106 Specification for `Fls_17_Dmu_Isr` API		
Syntax	`void Fls_17_Dmu_Isr ( void )`	
Service ID	0x2D	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This interrupt is mapped to the node: SRC_DMU0. This services the Write and Erase Jobs.	
Source	IFX	
Error handling	FLS_17_DMU_E_WRITE_FAILED, FLS_17_DMU_E_VERIFY_ERASE_FAILED, FLS_17_DMU_SE_INVALID_ISR, FLS_17_DMU_E_VERIFY_WRITE_FAILED, FLS_17_DMU_E_ERASE_FAILED	
Configuration dependencies	FlsUseInterrupts	

(table continues...)

Table 106 (continued) Specification for `Fls_17_Dmu_Isr` API

User hints	-
SFR accessed	DMU_HF_CLRE(w), DMU_HF_ECCC(rw), DMU_HF_ECCS(r), DMU_HF_ERRSR(rw), DMU_HF_SUSPEND(r), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.7 Callout

The driver does not support any callout functions.

1.3.8 Errors Handling

This section describes the various error types reported by the FLS driver.

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
FLS_17_DMU_E_BLANKCHECK_FAILED: Reported when the blank-check operation fails.	IFX	0x1E	RUNTIME	0x1E	RUNTIME
FLS_17_DMU_E_BUSY: Reported when the any FLS driver API service is called while the driver is still busy executing previous operation.	AUTOSAR	0x06	DET_SAFETY	0x06	DET_SAFETY
FLS_17_DMU_E_COMPARE_FAILED: Reported when the compare operation fails.	AUTOSAR	0x04	RUNTIME	0x04	TRANSIENT
FLS_17_DMU_E_ERASE_FAILED : Reported when the erase operation on DFLASH0 fails.	AUTOSAR	0x01	RUNTIME	0x01	TRANSIENT
FLS_17_DMU_E_HARDENCHK_FAIL: This is reported when the hardening check fails due to the hardware error.	IFX	0x37	RUNTIME	0x37	RUNTIME
FLS_17_DMU_E_INIT_FAILED: This runtime error is reported if OPER error is detected during initialization.	IFX	0x39	RUNTIME	0x39	RUNTIME

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
FLS_17_DMU_E_PARAM_ADDRESS: Reported when the FLS driver API service is called with the target/source address that is out of the range or when the passed address is not sector or page aligned.	AUTOSAR	0x02	DET_SAFETY	0x02	DET_SAFETY
FLS_17_DMU_E_PARAM_CONFIG: Reported when the FLS driver API service is called with a wrong parameter.	AUTOSAR	0x01	DET_SAFETY	0x01	DET_SAFETY
FLS_17_DMU_E_PARAM_DATA: Reported when the FLS driver API service is called with the value of source/target address as NULL pointer.	AUTOSAR	0x04	DET_SAFETY	0x04	DET_SAFETY
FLS_17_DMU_E_PARAM_LENGTH: Reported when the FLS driver API service is called with wrong length.	AUTOSAR	0x03	DET_SAFETY	0x03	DET_SAFETY
FLS_17_DMU_E_PARAM_POINTER: Reported when the FLS driver's Fls_17_Dmu_GetVersionInfo() API service is called with a NULL pointer as argument.	AUTOSAR	0x0a	DET_SAFETY	0x0a	DET_SAFETY
FLS_17_DMU_E_READ_FAILED: Reported when the read operation on DFLASH0 fails.	AUTOSAR	0x03	RUNTIME	0x03	TRANSIENT
FLS_17_DMU_E_RESUME_FAIL: This is reported when the resume of the erase operation fails due to the hardware error.	IFX	0x38	RUNTIME	0x38	RUNTIME
FLS_17_DMU_E_TIMEOUT: Reported when the timeout limit is exceeded during the execution of an FLS driver job.	AUTOSAR	0x09	DET_SAFETY	0x09	RUNTIME
FLS_17_DMU_E_UNINIT: Reported when any of the FLS driver's API service is called without properly initializing the driver.	AUTOSAR	0x05	DET_SAFETY	0x05	DET_SAFETY
FLS_17_DMU_E_VERIFY_ERASE_FAILED: Reported when the erase verification(blank check) fails.	AUTOSAR	0x07	DET_SAFETY	0x07	RUNTIME

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
FLS_17_DMU_E_VERIFY_WRITE_FAILED: Reported when the write verification (compare) fails.	AUTOSAR	0x08	DET_SAFETY	0x08	RUNTIME
FLS_17_DMU_E_WRITE_FAILED: Reported when write operation on DFLASH0 fails.	AUTOSAR	0x02	RUNTIME	0x02	TRANSIENT
FLS_17_DMU_SE_BUSY: This safety error is raised when the API service is called while the FLS driver is still busy.	IFX	0x06	SAFETY	0x06	SAFETY
FLS_17_DMU_SE_HW_BUSY: This is reported if the DFLASH0 flash bank is still busy with the operation.	IFX	0x6E	SAFETY	0x6E	SAFETY
FLS_17_DMU_SE_HW_TIMEOUT: This safety error is raised when the wait time for the execution of the suspend/ resume operation expires.	IFX	0x73	SAFETY	0x73	SAFETY
FLS_17_DMU_SE_ILLGL_OPERTN: This safety error is raised when the erase operation is suspended and a new erase operation is initiated.	IFX	0x64	SAFETY	0x64	SAFETY
FLS_17_DMU_SE_INIT_FAILED: This safety error is reported when the FLS erase operation is suspended and Fls_17_Dmu_Init() is invoked or the DFLASH0 emulation mode is not set to single ended sensing mode.	IFX	0x5F	SAFETY	0x5F	SAFETY
FLS_17_DMU_SE_INVALID_ISR: Error is reported as a safety error when there are spurious(not valid) interrupts.	IFX	0x78	SAFETY	0x78	SAFETY
FLS_17_DMU_SE_PARAM_ADDRESS: Reported when the API service is called with the target/ source address that is out of the range or when the passed address is not sector or page aligned.	IFX	0x02	SAFETY	0x02	SAFETY

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
FLS_17_DMU_SE_PARAM_DATA: Reported when the API service is called, with the source/target address as NULL pointer.	IFX	0x04	SAFETY	0x04	SAFETY
FLS_17_DMU_SE_PARAM_INVLD: This safety error is reported when the parameter passed as argument of the function is not valid.	IFX	0x5A	SAFETY	0x5A	SAFETY
FLS_17_DMU_SE_PARAM_LENGTH: Reported when the API service is called with wrong length.	IFX	0x03	SAFETY	0x03	SAFETY
FLS_17_DMU_SE_SUSPNDERASE_FAIL: This safety error is raised when the suspend error(ERR) in the suspend register(HF_SUSPEND) is set.	IFX	0x50	SAFETY	0x50	SAFETY

1.3.9 Deviations and limitations

This section describes the deviations and limitations of the FLS driver.

1.3.9.1 Deviations

This section describes the deviations of the FLS driver.

1.3.9.1.1 Software specification deviations

This section describes the deviations from software specification.

Table 107 Known deviations

Reference	Deviation
Protection setting is not used	Protection setting is not used in the FLS driver as it is more relevant for the FlsLoader driver and therefore the parameter `FlsProtection` is not supported.
`FlsMaxWriteFastMode` and `FlsMaxWriteNormalMode` configuration parameters are not supported	`FlsMaxWriteFastMode` / `FlsMaxWriteNormalMode` configuration parameters are not supported since write is performed for 8 or 32 bytes depending on the data size and page start address.
Runtime error	The runtime error reporting is configurable, if user disables the runtime error reporting, this is a deviation to AUTOSAR.
External flash driver	External flash driver is not supported.

(table continues...)

Table 107 (continued) Known deviations

Reference	Deviation
Unexpected flash ID error	FLS_E_UNEXPECTED_FLASH_ID error is not supported as external flash driver is not configured.
`FlsAcLoadOnJobStart`	`FlsAcLoadOnJobStart` configuration parameter is not supported because write and erase flash access code is executed from flash.
`FlsAcLocationWrite`	`FlsAcLocationWrite` configuration parameter is not supported because the write access code is executed from flash.
`FlsAcLocationErase`	`FlsAcLocationErase` configuration parameter is not supported because the erase access code is executed from flash.
Availability of `Fls_17_Dmu_Compare` API	For ASR440, `Fls_17_Dmu_Compare` API is not made available by Fls_Com.h. Instead, it is made available by Fls_17_Dmu.h.
For all requirements related to Runtime errors	Reporting of Runtime error: `Det_ReportRuntimeError` is done through `Mcal_Wrapper_Det_ReportRuntimeError` interface. All runtime error related datatypes and modified interfaces inclusion shall be done via `Mcal_Wrapper.h`

1.3.9.1.2 AMDC Violations

The FLS driver does not have any AMDC violations.

1.3.9.1.3 VSMD Violations

This section describes the violations reported by the EB VSMD checker tool with respect to AUTOSAR.

Table 108 Violations Reported by VSMD checker tool for EB03

Rule ID:	EB03

(table continues...)

Table 108 (continued) Violations Reported by VSMD checker tool for EB03

VSMD Node(s):	/AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsDemEventParameterRefs /AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsDemEventParameterRefs/FLS_E_COMPARE_FAILED /AURIX2G/EcAURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsDemEventParameterRefs /AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsDemEventParameterRefs/FLS_E_COMPARE_FAILED /AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsDemEventParameterRefs/FLS_E_ERASE_FAILED /AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsDemEventParameterRefs/FLS_E_ERASE_FAILED /AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsDemEventParameterRefs/FLS_E_READ_FAILED /AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsDemEventParameterRefs/ FLS_E_UNEXPECTED_FLASH_ID /AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsDemEventParameterRefs/FLS_E_WRITE_FAILED /AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsExternalDriver/AURIX2G/EcucDefs/Fls/ FlsConfigSet/ FlsJobEndNotification /AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsJobErrorNotification
Description	The StMD node has LOWER-MULTIPLICITY=0 and UPPER-MULTIPLICITY=1. The VSMD-node shall get theOPTIONAL-attribute instead of creating a list!
Additional Information:	

Table 109 Violations Reported by VSMD checker tool for EB09

Rule ID:	EB09
VSMD Node(s):	/AURIX2G/EcucDefs/Fls
Description	EB specific rule to check consistency of parameterpostBuildVariantUsed.
Additional Information:	

Table 110 Violations Reported by VSMD checker tool for EcuSws_1014

Rule ID:	EcuSws_1014
VSMD Node(s):	/AURIX2G/EcucDefs/Fls/AURIX2G/EcucDefs/Fls/ FlsConfigSet/AURIX2G/EcucDefs/Fls/FlsGeneral

(table continues...)

Table 110 (continued) Violations Reported by VSMD checker tool for EcuSws_1014

Description	Additional vendor specific parameter definitions(using ParameterTypes), container definitions andreferences shall be added to the VSMD according tothe alphabetical order.
Additional Information:	

Table 111 Violations Reported by VSMD checker tool for EcuSws_1035

Rule ID:	EcuSws_1035

(table continues...)

Table 111 (continued) Violations Reported by VSMD checker tool for EcuSws_1035

VSMD Node(s):	/AURIX2G/EcucDefs/Fls/AURIX2G/EcucDefs/Fls/ FlsConfigSet /AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsAcErase/AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsAcWrite/AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsCallCycle/AURIX2G/EcucDefs/Fls/FlsConfigSet/ lsDefaultMode/AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsDemEventParameterRefs/AURIX2G/EcucDefs/Fls/ FlsConfigSet/FlsDemEventParameterRefs/ FLS_E_COMPARE_FAILED/AURIX2G/EcucDefs/Fls/ FlsConfigSet/FlsDemEventParameterRefs/ FLS_E_ERASE_FAILED/AURIX2G/EcucDefs/Fls/ FlsConfigSet/FlsDemEventParameterRefs/ FLS_E_READ_FAILED/AURIX2G/EcucDefs/Fls/ FlsConfigSet/FlsDemEventParameterRefs/ FLS_E_UNEXPECTED_FLASH_ID/AURIX2G/ EcucDefs/Fls/FlsConfigSet/ FlsDemEventParameterRefs/FLS_E_WRITE_FAILED/ AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsExternalDriver/AURIX2G/EcucDefs/Fls/ FlsConfigSet/FlsExternalDriver/FlsSpiReference/ AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsJobEndNotification/AURIX2G/EcucDefs/Fls/ FlsConfigSet/FlsJobErrorNotification/AURIX2G/ EcucDefs/Fls/FlsConfigSet/FlsMaxReadFastMode/ AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsMaxReadNormalMode/AURIX2G/EcucDefs/Fls/ FlsConfigSet/FlsMaxWriteFastMode/AURIX2G/ cucDefs/Fls/FlsConfigSet/FlsMaxWriteNormalMode/ AURIX2G/EcucDefs/Fls/FlsConfigSet/FlsProtection/ AURIX2G/EcucDefs/Fls/FlsConfigSet/FlsSectorList/ AURIX2G/EcucDefs/Fls/FlsConfigSet/FlsSectorList/ FlsSector/AURIX2G/EcucDefs/Fls/FlsConfigSet/ FlsSectorList/FlsSector/FlsNumberOfSectors/ AURIX2G/EcucDefs/Fls/FlsConfigSet/FlsSectorList/ FlsSector/FlsPageSize/AURIX2G/EcucDefs/Fls/ FlsConfigSet/FlsSectorList/FlsSector/FlsSectorSize/ AURIX2G/EcucDefs/Fls/FlsConfigSet/FlsSectorList/ FlsSector/FlsSectorStartaddress/AURIX2G/ EcucDefs/Fls/FlsGeneral /AURIX2G/EcucDefs/Fls/ FlsGeneral/FlsAcLoadOnJobStart/AURIX2G/ EcucDefs/Fls/FlsGeneral/FlsBaseAddress/AURIX2G/ EcucDefs/Fls/FlsGeneral/FlsBlankCheckApi/AURIX2G/ EcucDefs/Fls/FlsGeneral/FlsCancelApi/AURIX2G/ EcucDefs/Fls/FlsGeneral/FlsCompareApi/AURIX2G/ EcucDefs/Fls/FlsGeneral/FlsDevErrorDetect/AURIX2G/ EcucDefs/Fls/FlsGeneral/FlsDriverIndex/AURIX2G/ EcucDefs/Fls/FlsGeneral/FlsGetJobResultApi/ AURIX2G/EcucDefs/Fls/FlsGeneral/FlsGetStatusApi/ AURIX2G/EcucDefs/Fls/FlsGeneral/FlsSetModeApi/

(table continues...)

Table 111 (continued) Violations Reported by VSMD checker tool for EcuSws_1035

	AURIX2G/EcucDefs/Fls/FlsGeneral/FlsTotalSize/ AURIX2G/EcucDefs/Fls/FlsGeneral/FlsUseInterrupts/ AURIX2G/EcucDefs/Fls/FlsGeneral/FlsVersionInfoApi/ AURIX2G/EcucDefs/Fls/FlsPublishedInformation/ AURIX2G/EcucDefs/Fls/FlsPublishedInformation/ FlsAcLocationErase/AURIX2G/EcucDefs/Fls/ FlsPublishedInformation/FlsAcLocationWrite/ AURIX2G/EcucDefs/Fls/FlsPublishedInformation/ FlsAcSizeErase/AURIX2G/EcucDefs/Fls/ FlsPublishedInformation/FlsAcSizeWrite/AURIX2G/ EcucDefs/Fls/FlsPublishedInformation/FlsEraseTime/ AURIX2G/EcucDefs/Fls/FlsPublishedInformation/ FlsErasedValue/AURIX2G/EcucDefs/Fls/ FlsPublishedInformation/FlsExpectedHwId/AURIX2G/ EcucDefs/Fls/FlsPublishedInformation/ FlsSpecifiedEraseCycles/AURIX2G/EcucDefs/Fls/ FlsPublishedInformation/FlsWriteTime
Description	For Containers, Parameters and References elementsUUID must be unique (also between StMD and VSMD).
Additional Information:	

Table 112 Violations Reported by VSMD checker tool for EcuSws_2101

Rule ID:	EcuSws_2101
VSMD Node(s):	/AURIX2G/EcucDefs/Fls/POST_BUILD_VARIANT_USED
Description	For each ConfigurationVariant supported bythe ModuleDef, there must be oneImplementationConfigClass element. In VSMD, theImplementationConfigClass is mandatory.
Additional Information:	

Table 113 Violations Reported by VSMD checker tool for EcuSws_6003

Rule ID:	EcuSws_6003
VSMD Node(s):	/AURIX2G/EcucDefs/Fls
Description	The SHORT-NAME of the AR-PACKAGEs of StMD andVSMD must be diffeent to ensure a unique SHORT-NAME-path.
Additional Information:	

Table 114 Violations Reported by VSMD checker tool for Tps_Ecuc_06051_ASR41

Rule ID:	Tps_Ecuc_06051_ASR41
VSMD Node(s):	/AURIX2G/EcucDefs/Fls/POST_BUILD_VARIANT_USED

(table continues...)

Table 114 (continued) Violations Reported by VSMD checker tool for Tps_Ecuc_06051_ASR41

Description	The implementationConfigClass of anEcucParameterDef or EcucAbstractReferenceDef inVSMD shall be the same or higher (where PreCompileconfiguration class is considered to be the lowest andPostBuild the highest) as in StMD with respect to theselected subset defined by the actually implementedsupportedConfigVariant.
Additional Information:	

1.3.9.2 Limitations

This section describes the limitations of the FLS driver.

Table 115 Known limitations

Reference	Limitation
`FlsMaxWriteNormalMode`, `FlsMaxWriteFastMode`	These parameters are fixed to 32 bytes.
`Fls_17_Dmu_Cancel`	Although the API is synchronous, hardware may be still busy after returning from `Fls_17_Dmu_Cancel` API due to already issued flash erase or write command sequence. In such scenarios, any new job issued may get rejected with return value as `E_NOT_OK` and safety error as `FLS_17_DMU_E_HW_BUSY`. The user may choose to retry or re-issue the same job again.
Timeout of flash operations	All timeout values used by the FLS module are calculated assuming the FSI operation at 100MHz.
Erase-suspend feature of FLS driver not to be used during simultaneous access of DFlash0 and DFlash1	When the command to resume erase is initiated on DFlash0 by the FLS driver on the TriCore side and DFlash1is already being accessed by FLS driver on the HSM side, then FSI gets into time-sliced mode of operation to cater to both the requests. Hence, the resume erase operation takes longer time than expected and may lead to timeout. In a scenario where FLS is used with IFX FEE, FEE retries the resume erase operation in case of a failed resume erase operation (due to timeout). During retry, there can be a situation where FLS resume erase operation is successful but the erase job end notification is never raised by the FLS driver. In this situation, FEE driver will hang. Hence, it is recommended to not use the erase-suspend feature during simultaneous access of DFlash0 and DFlash1. [cover parentID FLS={B5E62EDC-1205-401c-B511-6FF0F2C45C39}]

(table continues...)

Table 115 (continued) Known limitations

Reference	Limitation
When FLS is used with IFX FEE, QS and NVM features of FEE not to be used together during simultaneous access of DFlash0 and DFlash1	When user content count command is initiated on DFlash0 by the FLS driver on the TriCore side and DFlash1 is already being accessed by FLS driver on the HSM side, then FSI gets into time-sliced mode of operation to cater to both the requests. Hence, the user content count command sequence will take longer time than expected and may lead to timeout. In a scenario where FLS is used with IFX FEE, if timeout occurs, FLS will return hardening error. In turn, FEE will not perform hardening check and hardening of the current wordline or page. The data in this wordline or page may be lost since hardening was not done when needed. The probability of occurrence of this situation is low. Hence, it is recommended to not use both QS and NVM features of FEE together during simultaneous access of DFlash0 and DFlash1. [cover parentID FLS={4CD1AAE1-25D9-43b5-9629-0AFEC4D7FF8F}]
Write and erase functionality - Impact of parallel operations on DFlash0 and DFlash1	In the case of concurrent operations on DFlash0 and DFlash1(i.e. active time slicing), the erase time increases by about 15% for CPU erase commands and the write times are prolonged by 5ms. The increased values are considered for timeout calculations for asynchronous operations only. For synchronous operations (resume erase and hardening check), the increased values cause higher execution times and hence are not considered for timeout calculations.

Revision history

Table 116 Revision history

Date	Version	Description
2024-08-20	7.0	Released
2024-08-09	6.1	- SFR accessed field is updated for following API's under section 1.3.3: Fls_17_Dmu_CompareWordsSync, Fls_17_Dmu_Init, Fls_17_Dmu_IsHardeningRequired, Fls_17_Dmu_ReadWordsSync, Fls_17_Dmu_VerifyErase, Fls_17_Dmu_VerifySectorErase, Fls_17_Dmu_Write, Fls_17_Dmu_MainFunction, Fls_17_Dmu_Isr - Section 1.1.3.1 is updated to hide the connector between Mcal_Wrapper.h to Det.h
2023-12-05	6.0	Released
2023-12-05	5.1	Formula added for FlsEraseTime and FlsWriteTime in section 1.3.1.8.5 and 1.3.1.8.9.
2023-07-06	5.0	Released Review comments fixed •Updated "release plan" to "release notes" in revision history. • Mcal Wrapper module description updated in section 1.1.4.1.
2023-06-14	4.1	•ASIL level field changed to Safety level with value as "refer to release notes" for all APIs under 1.3.3 Functions - APIs •DEM module removed and Mcal_Wrapper module added in "1.1.4.1 Intergration with AUTOSAR stack" section • Runtime error information are removed in DET module and added in Mcal_Wrapper module in "1.1.4.1 Intergration with AUTOSAR stack" section •Mcal_Wrapper.h added in the "1.1.3.1 C file structure" section “Figure2 Fls_C_file_structure-1.png" and “Table 2 C file structure” •Note added in "1.1.1 Description" section. •DEM module removed and Mcal_Wrapper module added in 1.1.2 Hardware -Software mapping section “Figure1 Mapping of hardware-software interfaces”. •Updated the section 1.3.9.1.1: Software Specification Deviations for Autosar requirements. - Added the Reference "For all requirements related to Runtime errors". - Updated Description to add Mcal_Wrapper Module Information.
2022-07-07	4.0	Released

(table continues...)

Table 116 (continued) Revision history

2022-07-04	3.1	For JIRA 0000053912-18277, corrected NVM Sector0 Start address in Example Usage
2021-11-18	3.0	Released
2021-11-12	2.1	Updated config variant info, Removed delay() from example usage erase and write demo code section. Removed MemIf.h related deviation from software specific deviations section.
2020-12-10	2.0	Released
2020-12-08	1.1	- Removed limitation - Fls_17_Dmu_Write API - Regarding passing input parameter SourceAddressPtr to be word-aligned - Removed AMDC violations after ASR440 updates - Removed Software Specification deviations 1. Error handling - FLS_17_DMU_E_ERASE_FAILED, FLS_17_DMU_E_WRITE_FAILED, FLS_17_DMU_E_READ_FAILED FLS_17_DMU_E_COMPARE_FAILED 2. Error handling - FLS_17_DMU_E_VERIFY_ERASE_FAILED, FLS_17_DMU_E_VERIFY_WRITE_FAILED, FLS_17_DMU_E_TIMEOUT 3. Fls_17_Dmu_Write API - SourceAddressPtr (data buffer) alignment 4. Configuration Parameter- FlsEcucPartitionRef 5. Configuration Parameter- FlsWriteVerificationEnabled 6. Configuration Parameter- FlsEraseVerificationEnabled 7. Configuration Parameter- FlsTimeoutSupervisionEnabled 8. Behavior of timeouts for erase and write jobs - Updated VSMD violations after ASR440 updates - Added 'Handling of errors when IFX FEE is used' under Key Architectural Considerations section - Updated Example usage regarding timeout handling of erase and write jobs - Added information regarding 'Configuration of FlsIllegalStateNotification' under Example usage section - Removed limitation - FLS_17_DMU_E_TIMEOUT Error - Added deviation - Availability of Fls_17_Dmu_Compare API
2020-08-14	1.0	Released

(table continues...)

Table 116 (continued) Revision history

2020-08-06	0.1	- Initial Version - Fls_17_Dmu driver chapter moved from MC-ISAR_TC3xx_UM_Basic to this document - For 0000053912-11337, added AMDC violations - For 0000053912-11626, added VSMD violations - For 0000053912-12575, unsupported HW features removed. All information captured under 'Hardware-Software mapping' section - For 0000053912-12506, example usage section is corrected - For 0000053912-12373, the type of FlsSpiReference is corrected - For 000053912-10907, added limitations related to timeout handling, usage of Erase-suspend and hardening features during parallel access of DFlash0 and DFlash1 by Tricore and HSM respectively, updated the timeout values for the write and erase operations considering parallel access of DFlash0 and DFlash1 by Tricore and HSM respectively, updated deviation - 'Behavior of timeouts for erase and write jobs' - For 0000053912-12329, spell check corrected under Hardware-Software mapping section - For 0000053912-12502 and 0000053912-12477, harmonization and format update in all the section - For 0000053912-12452, limitation regarding forcing the minimum value of FlsCallCycle to be 200 microseconds is removed - For 0000053912-13020, listed ASR440 deviations - Removed deviation - 'FLS_E_VERIFY_ERASE_FAILED in Interrupt mode'

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2024-08-20

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-ocr1484806431059

Important notice

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, 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.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.