MCAL User Manual for FlsLoader

32-bit TriCore^{TM} AURIX^{TM} 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^{TM} AURIX^{TM} 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 FlsLoader 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^{TM} TC3xx MCAL User Manual General

Table of contents

Table of contents

Table of contents

1 FlsLoader driver

1.1 User information

1.1.1 Description

Two types of non-volatile memory (NVM) are instantiated in the 2nd Generation AURIX (TC3xx) microcontroller.

- Program Flash (PFlash) stores the program code and constant data
- Data Flash (DFlash) stores the application-specific data

The FLSLOADER driver provides the following services:

- Initialization and de-initialization of the Flash
- Writing the program and data to the Flash
- Erasing the contents of the Flash
- Locking and unlocking the Flash

These services of the driver are operable on DFlash bank 0 and all PFlash banks of the microcontroller. All references to DFlash and PFlash, in this section, are meant for bank 0 of the DFlash and all banks of the PFlash, respectively. The driver is delivered as a pre-compile variant. Therefore, the driver supports configuration parameters with only pre-compile configuration class.

1.1.2 Hardware-software mapping

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

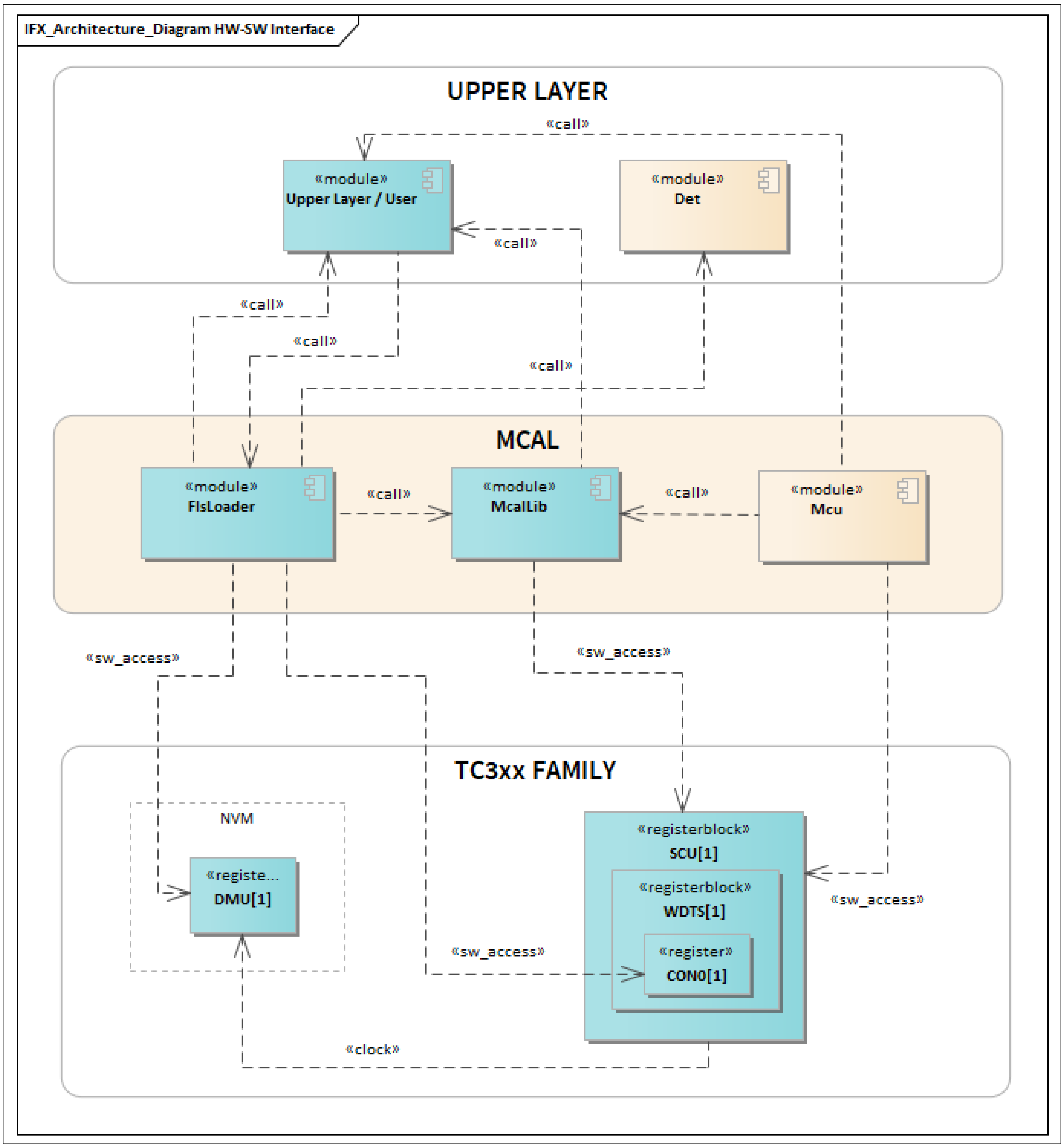


Figure 1 **Mapping of hardware-software interfaces**

1.1.2.1 DMU: primary hardware peripheral

Hardware functional features

The FLSLOADER driver uses the DMU for various operations on the PFlash/DFlash memories. The key hardware functional features used by the driver are:

- Write and erase to PFlash:

i. 32 bytes page programming and 256 bytes burst. programming.

ii. Erase by multi-sector erase commands.

- Write and erase to DFlash0 and UCB:

i. 8 bytes page programming and 32 bytes burst programming.

ii. Erase by multi-sector erase commands.

- Single ended mode support for DFlash0
- Password based protection of PFlash/DFlash0 banks through UCBs

The unsupported features of the DMU are:

- Complement sensing mode for DFlash0
- Suspend and resume operations
- ECC error reporting to safety management unit (SMU)
- Interrupt service requests

Users of the hardware

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

Hardware diagnostic features

The SMU alarms configured for the DMU are not monitored by the FLSLOADER driver.

Hardware events

The FLSLOADER driver uses the following hardware events for error event from the DMU IP:

- 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 hardware when write or erase command executed on protected memory section.
- Operation Error (OPER): This flag is set by hardware when Flash standard interface (FSI) encounters any error.
- Sequence Error (SQER): This flag is set by hardware when improper DMU command sequences are executed.

1.1.2.2 SCU: dependent hardware peripheral

Hardware functional features

The FLSLOADER driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB, fSRI and fFSI clock signals for functioning.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. The FLSLOADER driver accesses the SCU register to disable the Safety Endinit protection temporarily for PFlash write and erase operations. Since the driver is used during the boot, access to the SCU register will be not be concurrent with other drivers in runtime.

Hardware diagnostic features

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

Hardware events

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

1.1.3 File structure

1.1.3.1 C file structure

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

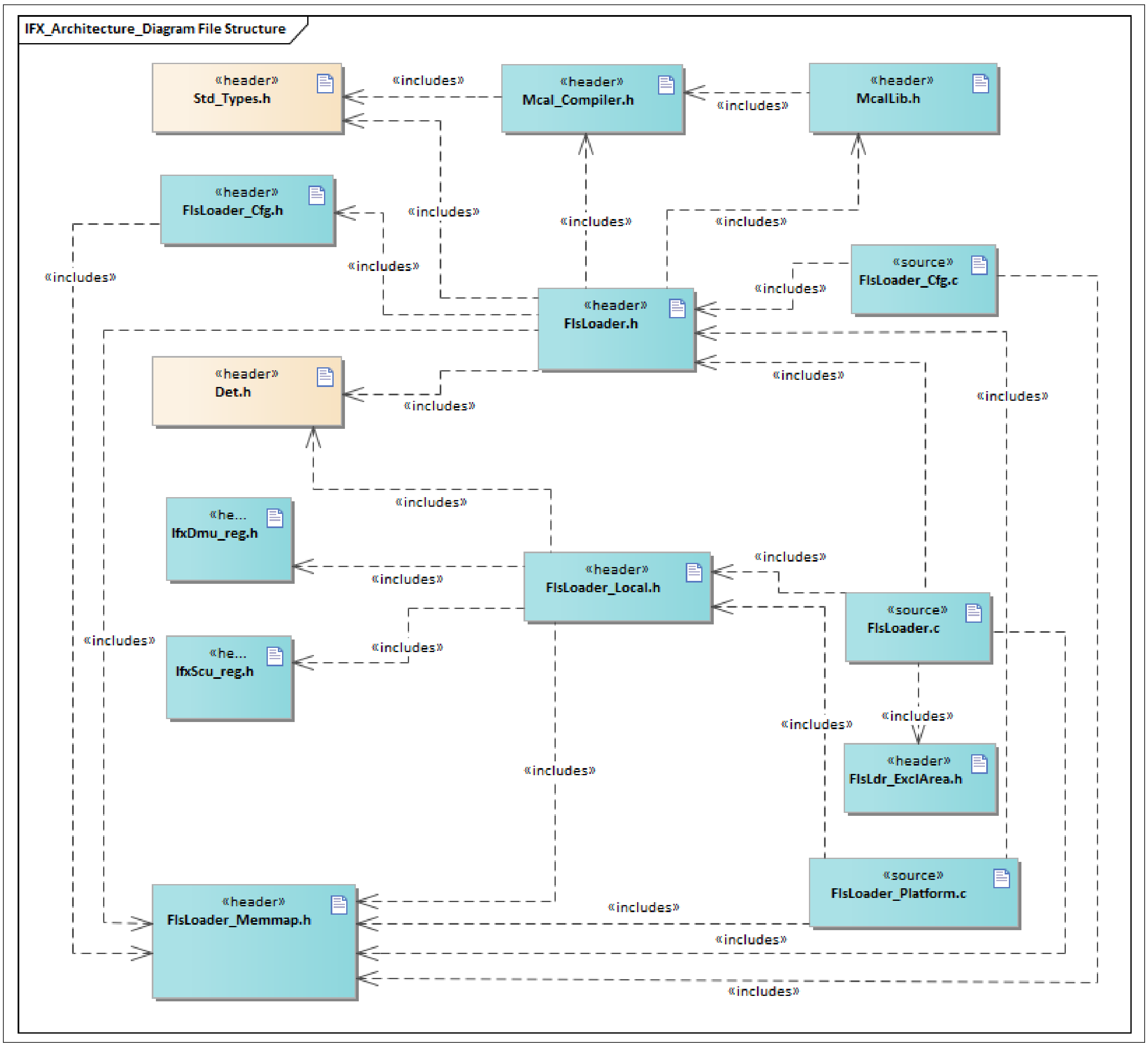


Figure 2 **FlsLoader_File_Structure-1.png**

Table 2 C file structure

File name	Description
`Det.h`	Provides the exported interfaces of Development Error Tracer
`FlsLdr_ExclArea.h`	Header file containing the prototype of the APIs to define the start and the end of an exclusive area of FLSLOADER module.
`FlsLoader.c`	Implementation of FLSLOADER driver functionality
`FlsLoader.h`	FLSLOADER driver header definition file
`FlsLoader_Cfg.c`	Pre-compile configuration data file for the FLSLOADER driver functionality
`FlsLoader_Cfg.h`	FLSLOADER driver configuration generated out of ECUC file
`FlsLoader_Local.h`	FLSLOADER driver local header definition file
`FlsLoader_Memmap.h`	Mapping of code and data (variables, constant variables) used by FLSLOADER driver to specific memory sections
`FlsLoader_Platform.c`	FLSLOADER driver platform-specific source file
`IfxDmu_reg.h`	SFR header file for Dmu
`IfxScu_reg.h`	SFR header file for SCU
`McalLib.h`	Static header file defining prototypes of data structure and APIs exported by the MCALLIB.
`Mcal_Compiler.h`	Header file providing abstraction for TriCore^{TM}-intrinsic instruction.
`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 FLSLOADER driver.

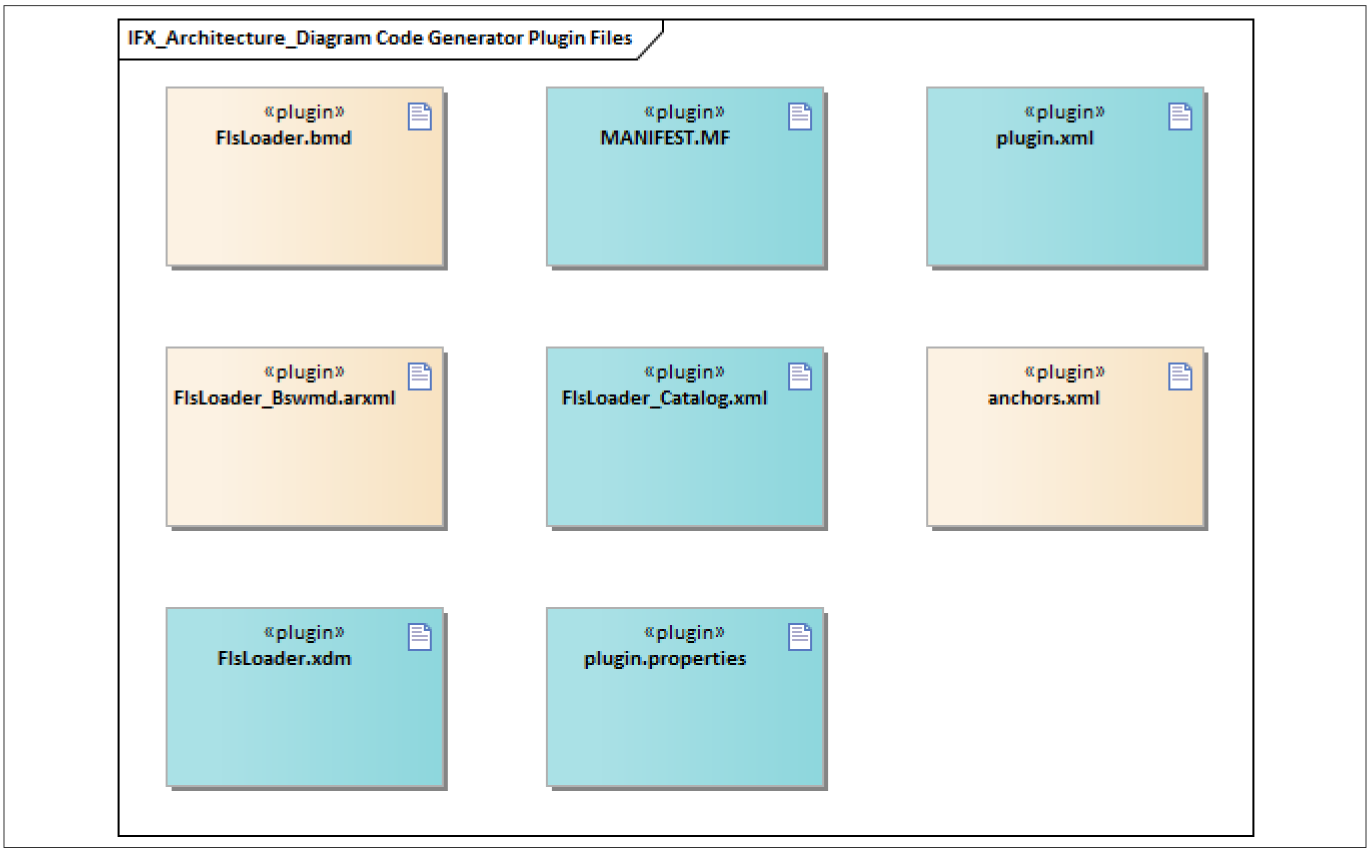

IFX_Architecture_Diagram Code Generator Plugin Files
«plugin» FlsLoader.bmd
«plugin» MANIFEST.MF
«plugin» plugin.xml
«plugin» FlsLoader_Bswmd.arxml
«plugin» FlsLoader_Catalog.xml
«plugin» anchors.xml
«plugin» FlsLoader.xdm
«plugin» plugin.properties

Figure 3 **FlsLoader_Code_Generator_Plugin_Files-1.png**

Table 3 **Code generator plugin files**

File name	Description
`FlsLoader.bmd`	Code template macro file for FLSLOADER driver
`FlsLoader.xdm`	Tresos format XML data model schema file
`FlsLoader_Bswmd.arxml`	AUTOSAR format module description file
`FlsLoader_Catalog.xml`	AUTOSAR format catalog file
`MANIFEST.MF`	Tresos plugin support file containing the metadata for the FLSLOADER driver
`anchors.xml`	AUTOSAR format module description file
`plugin.properties`	Tresos plugin support file for the FLSLOADER driver
`plugin.xml`	Tresos plugin support file for the FLSLOADER driver

1.1.4 Integration hints

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

Flash reprogramming

- Complete driver code or write and erase APIs can be placed in the RAM or in another Flash bank for which Flash operations are not being executed.

For example, if the erase or write operations need to be executed on PFlash bank0 sectors, place the code in either PFlash bank 1 to 5 or the RAM.

- The driver does not provide any APIs or functions for moving the write and erase APIs to the RAM. User of this API has to take care in the application.

Safety watchdog configuration

- Application shall configure the safety watchdog timer in the static and time independent password mode.

1.1.4.1 Integration with AUTOSAR stack

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

- **EcuM**

  The FLSLOADER driver is designed for the boot loader application, which may not contain EcuM.

  Therefore, the application software must ensure that the initialization and de-initialization of the FLSLOADER driver are invoked before and after using its services.

- **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 relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `FlsLoader_MemMap.h` file.

  The `FlsLoader_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.

```
#define MEMMAP_ERROR

 /*** GLOBAL RAM DATA ***/
#if defined FLSLOADER_START_SEC_VAR_CLEARED_QM_LOCAL_8
 /*****User pragmas here for DSPR of CPU core*****/
 #undef FLSLOADER_START_SEC_VAR_CLEARED_QM_LOCAL_8
 #undef MEMMAP_ERROR
#elif defined FLSLOADER_STOP_SEC_VAR_CLEARED_QM_LOCAL_8
 /*****User pragmas here for DSPR of CPU core*****/
 #undef FLSLOADER_STOP_SEC_VAR_CLEARED_QM_LOCAL_8
 #undef MEMMAP_ERROR
#elif defined FLSLOADER_START_SEC_VAR_CLEARED_QM_LOCAL_32
 /*****User pragmas here for DSPR of CPU core*****/
 #undef FLSLOADER_START_SEC_VAR_CLEARED_QM_LOCAL_32
 #undef MEMMAP_ERROR
#elif defined FLSLOADER_STOP_SEC_VAR_CLEARED_QM_LOCAL_32
 /*****User pragmas here for DSPR of CPU core*****/
 #undef FLSLOADER_STOP_SEC_VAR_CLEARED_QM_LOCAL_32
 #undef MEMMAP_ERROR

 /*** CONSTANT DATA ***/
#elif defined FLSLOADER_START_SEC_CONST_QM_LOCAL_8
 /*****User pragmas here for PFlash*****/
 #undef FLSLOADER_START_SEC_CONST_QM_LOCAL_8
 #undef MEMMAP_ERROR
#elif defined FLSLOADER_STOP_SEC_CONST_QM_LOCAL_8
 /*****User pragmas here for PFlash*****/
 #undef FLSLOADER_STOP_SEC_CONST_QM_LOCAL_8
 #undef MEMMAP_ERROR
#elif defined FLSLOADER_START_SEC_CONST_QM_LOCAL_32
 /*****User pragmas here for PFlash*****/
 #undef FLSLOADER_START_SEC_CONST_QM_LOCAL_32
 #undef MEMMAP_ERROR
#elif defined FLSLOADER_STOP_SEC_CONST_QM_LOCAL_32
 /*****User pragmas here for PFlash*****/
 #undef FLSLOADER_STOP_SEC_CONST_QM_LOCAL_32
 #undef MEMMAP_ERROR

 /*** CODE ***/
#elif defined FLSLOADER_START_SEC_CODE_QM_LOCAL
 /*****User pragmas here for PFlash*****/
 #undef FLSLOADER_START_SEC_CODE_QM_LOCAL
 #undef MEMMAP_ERROR
#elif defined FLSLOADER_STOP_SEC_CODE_QM_LOCAL
 /*****User pragmas here for PFlash*****/
 #undef FLSLOADER_STOP_SEC_CODE_QM_LOCAL
 #undef MEMMAP_ERROR
#elif defined FLSLOADER_START_SEC_WRITEERASE_CODE_QM_LOCAL
 /*****User pragmas here for PFlash*****/
 #undef FLSLOADER_START_SEC_WRITEERASE_CODE_QM_LOCAL
```

```
   #undef MEMMAP_ERROR
  #elif defined FLSLOADER_STOP_SEC_WRITEERASE_CODE_QM_LOCAL
   /*****User pragmas here for PFlash*****/
   #undef FLSLOADER_STOP_SEC_WRITEERASE_CODE_QM_LOCAL
   #undef MEMMAP_ERROR
  #endif

  #if defined MEMMAP_ERROR
  #error "Flsloader_MemMap.h, wrong pragma command"

  #endif
```

- **DET**

  The DET module is a part of the AUTOSAR stack that handles all the development and runtime errors reported by the BSW modules. The FLSLOADER driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the FLSLOADER driver must process all the errors reported to the DET module through the API `Det_ReportError()`. The files `Det.h` and `Det.c` are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.
- **DEM**

  The DEM module is not required for the integration of the FLSLOADER driver.
- **SchM**

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

  The FLSLOADER driver does not report safety errors.
- **Callout**

  The FLSLOADER driver provides optional callout function to the application while looping for hardware busy status during write and erase operations.

  To enable the callout function, user shall define callout function name using the `FlsLoaderCallOutFunction` configuration and a timeout interval using the `FlsLoaderCallOutTime` configuration.

  If enabled, the callout function is invoked at configured timeout interval during FLSLOADER write and erase operations. Application shall define the callout function in application software.

  The following sequence diagram shows callout functionality during erase operation. Similarly callout shall be invoked by the driver during the write operations.

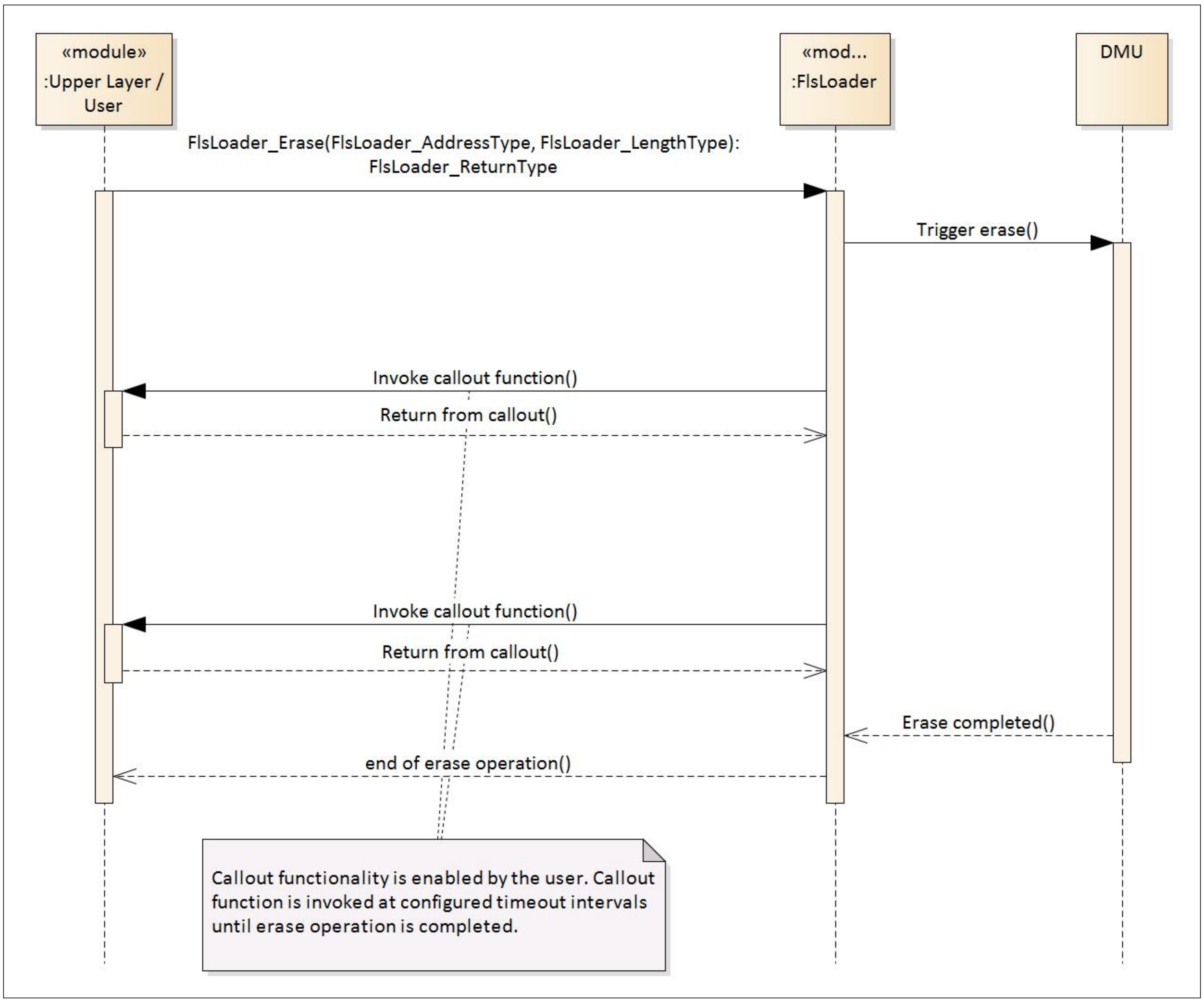

«module»
:Upper Layer / User
«mod...
:FlsLoader
DMU
FlsLoader_Erase(FlsLoader_AddressType, FlsLoader_LengthType): FlsLoader_ReturnType
Trigger erase()
Invoke callout function()
Return from callout()
Invoke callout function()
Return from callout()
Erase completed()
end of erase operation()
Callout functionality is enabled by the user. Callout function is invoked at configured timeout intervals until erase operation is completed.

Figure 4 **Erase operation with callout enabled**

- **Operating system (OS)**
  The OS is not required for the integration of the FLSLOADER driver.
- **Notifications and callbacks**
  The FLSLOADER driver does not provide call-backs and notifications.

1.1.4.2 Multicore and Resource Manager

The FLSLOADER driver does not support execution on multiple cores in simultaneously.

1.1.4.3 MCU support

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

1.1.4.4 Port support

The FLSLOADER driver does not use any services provided by the Port driver.

1.1.4.5 DMA support

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

1.1.4.6 Interrupt connections

The FLSLOADER driver does not use any interrupt source.

1.1.4.7 Example usage

This section describes how the FLSLOADER driver can be configured and how to use different APIs provided by it. All APIs should be provided with valid input parameters. To detect the invalid function parameters, DET (Development Error Tracer) should be enabled. Behavior of APIs undefined if DET is disabled and wrong parameters passed.

Driver APIs are designed to be non-reentrant, which means if there is an ongoing Flash operation and then an interrupt occurs or driver invokes its callout function if enabled, the driver APIs should not be called again within this callout function or interrupt context.

For more details on program or data Flash wait cycle configuration refer to the hardware user manual section “Configuring Flash Read Access Cycles”.

Configuration

Configuration of the FLSLOADER driver involves the following steps:

- Configuration to generate system clock of the required frequency. This configuration is done using the MCU driver.
- General guidelines for configuration of the FLSLOADER driver.

  - Lock check is an optional functionality for erase and write APIs and can be switched off.

  - For write and erase of Flash a minimum set of configuration without DET, Deinit, Lock and Unlock can be configured.

Refer to section 1.3 for all EB tresos configuration interfaces of the FLSLOADER driver.

Initialization of the driver

Initialize the MCU driver so that system clock is up and running. Initialize the FLSLOADER driver by calling the FlsLoader_Init API with the NULL pointer.

The following code listing shows FLSLOADER driver initialization.

```
/* MCU driver initialization */
Mcu_Init(ConfigPtr);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock();

/* Initialize the FLSLOADER driver */
retval = FlsLoader_Init(NULL_PTR);

/* Check if driver initialized successfully */
if (retval == FLSLOADER_E_OK)
{
 /* FlsLoader_Init returned FLSLOADER_E_OK. Driver initialized successfully.*/
}
```

Erase and write PFlash and DFlash0

PFlash and DFlash0 are erased sector wise and are written page wise.

For erasing the Flash, erase API should be called with start address of a sector in Flash and number of sectors to be erased.

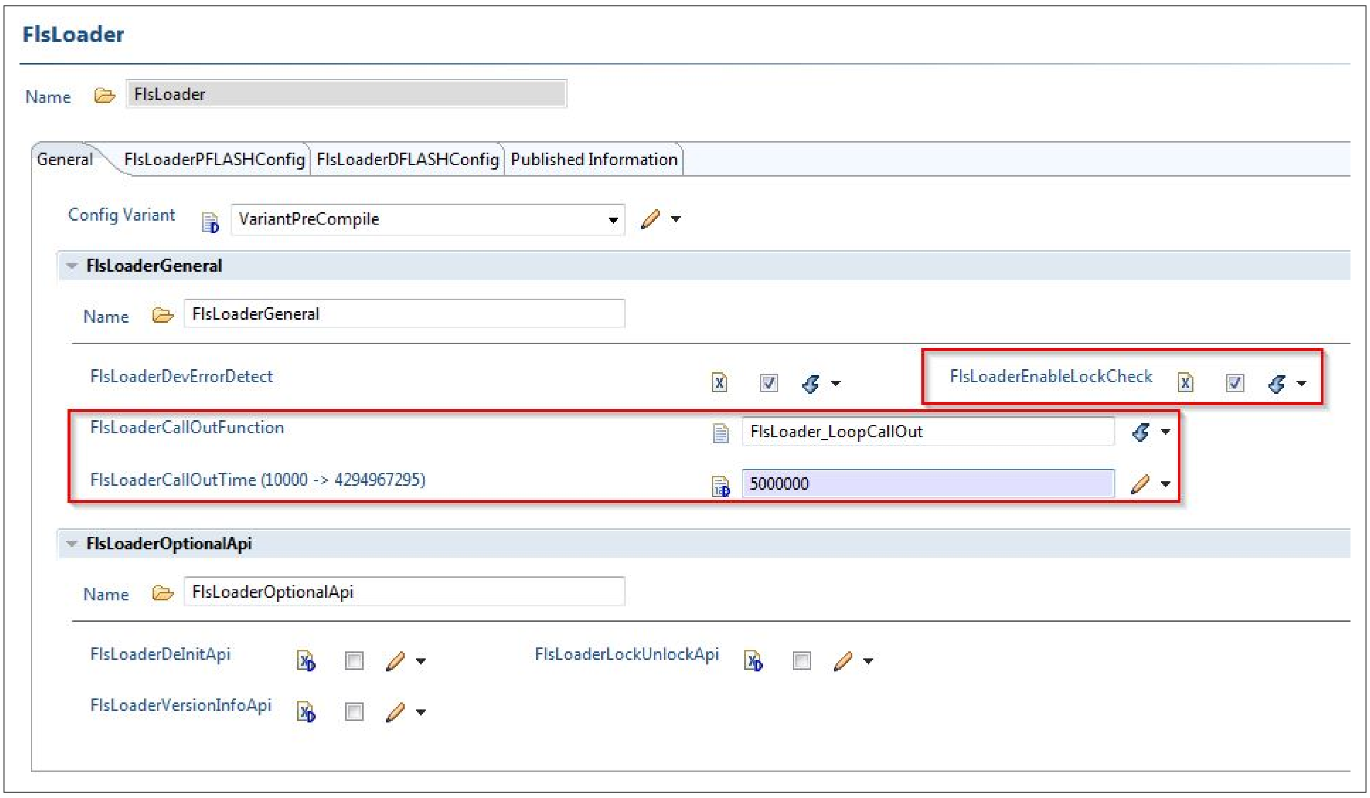
For writing the Flash, write API should be called with start address of a page in Flash, number of bytes to be written along with source data address. Number of bytes to be written should be multiple of the page size.

Refer to the hardware user manual sections **Program Flash Banks** and **Data Flash Bank DFLASH0** regarding more details on structure of PFlash and DFlash0 banks.

The driver provides an optional lock-check functionality for erase and write APIs, which when enabled checks if a sector or bank falling under current erase or write request is protected and it exits from the API safely if sector or bank found to be protected. User can enable this functionality by enabling the configuration `FlsLoaderEnableLockCheck`.

The driver provides another optional callout functionality for erase and write APIs, which when enabled invokes callout function to application at regular time intervals. Configuration and details of callout functionality is explained under **Callout** subsection of **Integration with AUTOSAR stack** section.

If a user requirement is to enable lock-check and callout functionalities with callout function name and callout time configured as `FlsLoader_LoopCallOut` and `5ms` respectively, following configuration should be made in configuration tool is shown as follows.


FlsLoader

Name: FlsLoader

General | FlsLoaderPFLASHConfig | FlsLoaderDFLASHConfig | Published Information

Config Variant: VariantPreCompile

FlsLoaderGeneral

Name: FlsLoaderGeneral

FlsLoaderDevErrorDetect ✓
FlsLoaderEnableLockCheck ✓
FlsLoaderCallOutFunction: FlsLoader_LoopCallOut
FlsLoaderCallOutTime (10000 -> 4294967295): 5000000

FlsLoaderOptionalApi

Name: FlsLoaderOptionalApi

FlsLoaderDeInitApi ☐
FlsLoaderLockUnlockApi ☐
FlsLoaderVersionInfoApi ☐

Figure 5 **Configuration: Enable lock-check, Callout function FlsLoader_LoopCallOut, Callout time 52000000 ns**

Sequence for the FLSLOADER erase operation with callout enabled is shown as follows.

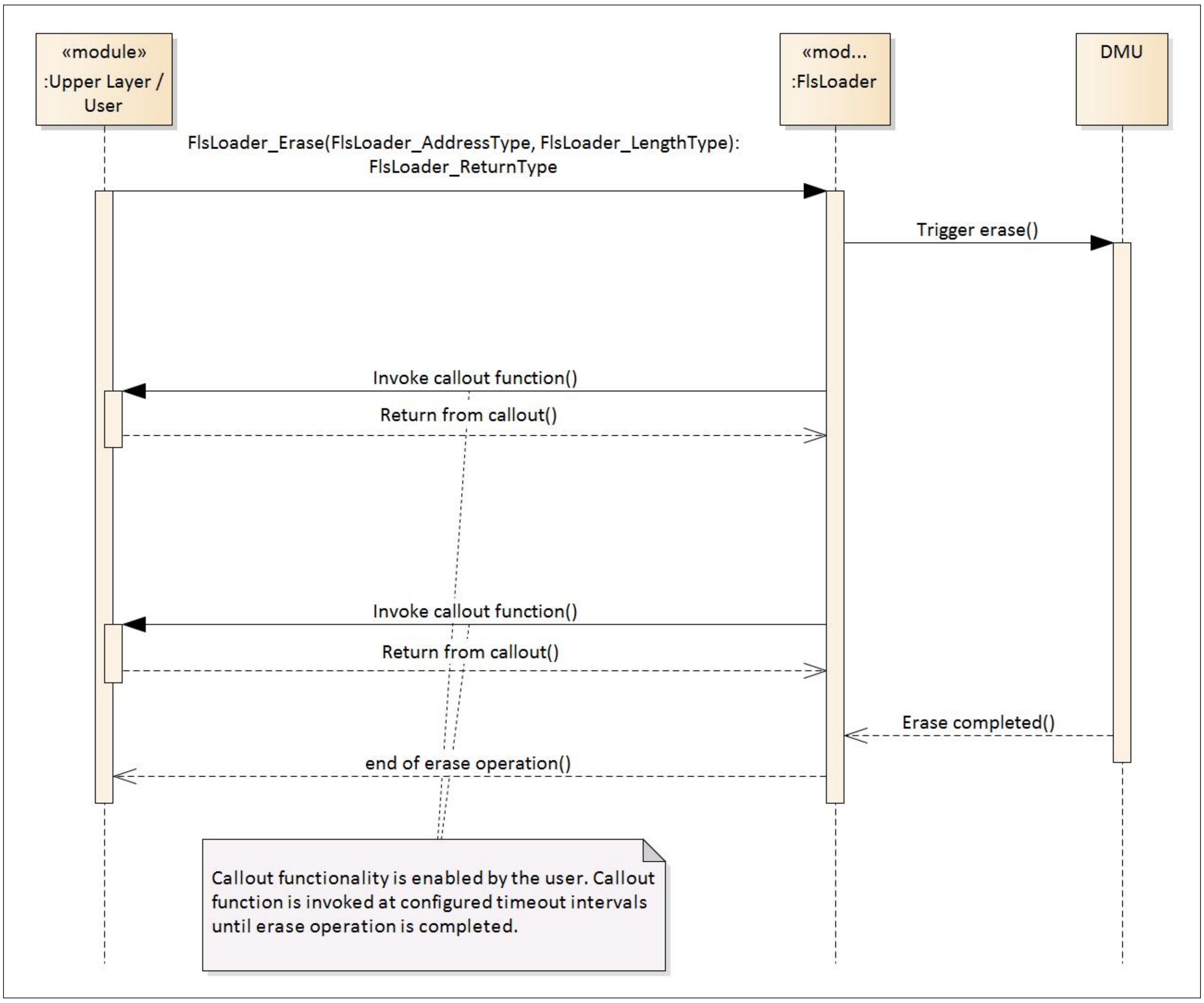

«module»
:Upper Layer / User
«mod...
:FlsLoader
DMU
FlsLoader_Erase(FlsLoader_AddressType, FlsLoader_LengthType): FlsLoader_ReturnType
Trigger erase()
Invoke callout function()
Return from callout()
Invoke callout function()
Return from callout()
Erase completed()
end of erase operation()
Callout functionality is enabled by the user. Callout function is invoked at configured timeout intervals until erase operation is completed.

Figure 6 **Sequence for FLSLOADER erase operation with callout enabled**

Sequence for the FLSLOADER write operation with callout enabled is shown as follows.

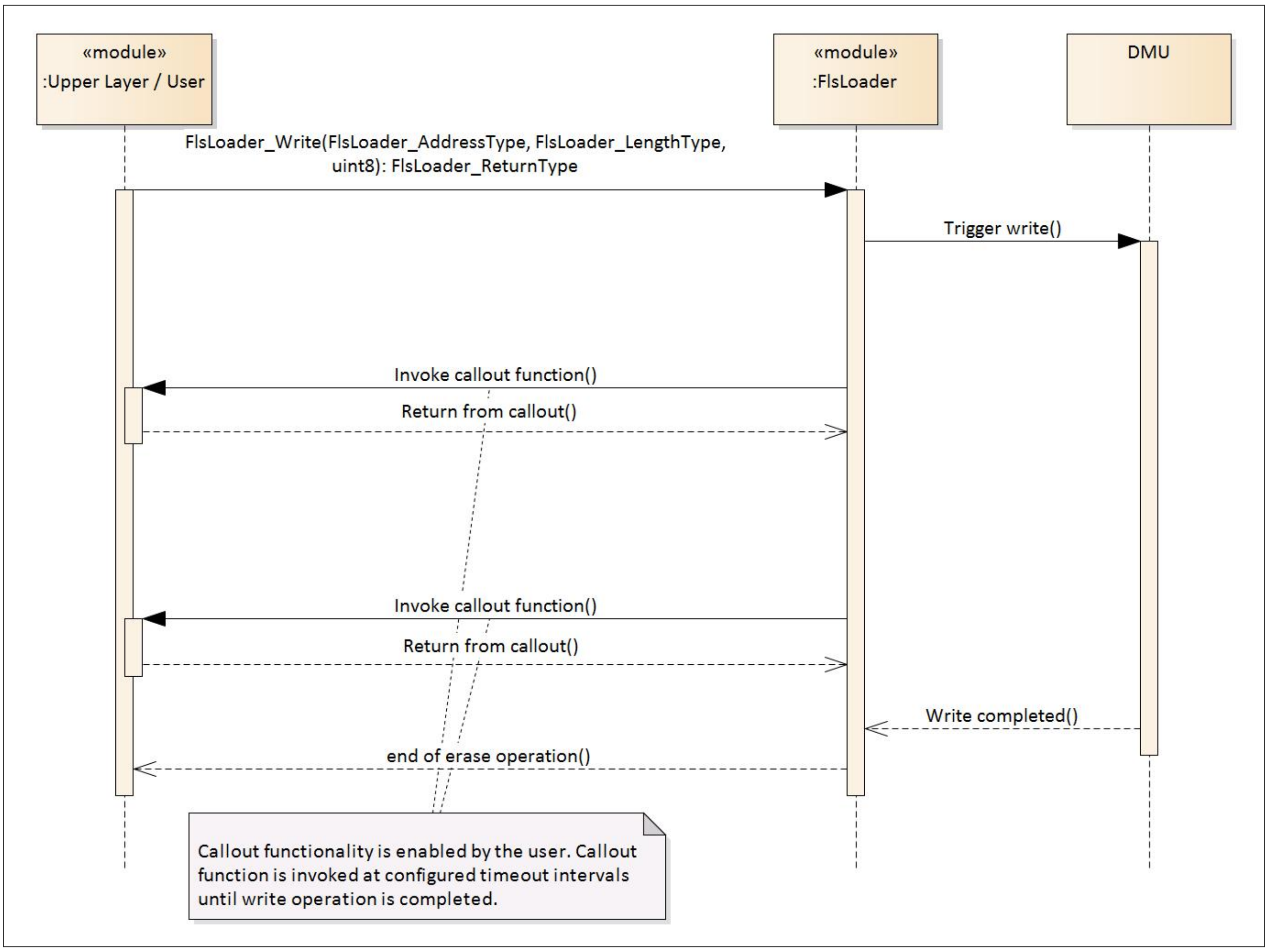

«module»
:Upper Layer / User
«module»
:FlsLoader
DMU
FlsLoader_Write(FlsLoader_AddressType, FlsLoader_LengthType, uint8): FlsLoader_ReturnType
Trigger write()
Invoke callout function()
Return from callout()
Invoke callout function()
Return from callout()
Write completed()
end of erase operation()
Callout functionality is enabled by the user. Callout function is invoked at configured timeout intervals until write operation is completed.

Figure 7 **Sequence for FLSLOADER write operation with callout enabled**

Sample code to erase the first two sectors of the PFlash bank 2 and write 1024 bytes of data to sector 0 is as follows:

```
/* Erase operation start.. */
/* Erase 2 sectors starting from 0xA0600000U */

/* Erase Start address */
FlsLoader_AddressType FlsLoader_Address = 0xA0600000U;

/*Erase Length*/
FlsLoader_LengthType EraseLength = 2U;

retval = FlsLoader_Erase((uint32)FlsLoader_Address, EraseLength);

/* Check if sectors erased successfully */
if(retval == FLSLOADER_E_OK)
{
 /* FlsLoader_Erase API returned with FLSLOADER_E_OK. Sectors erased successfully */

 /* Programming operation start.. */
 /* Writing 1024byte data to 0xA0600000U */

 /*Write Start Address*/
 FlsLoader_Address = 0xA0600000U;

 /*Write Length*/
 FlsLoader_LengthType WriteLength = 1024U;

 uint8 Buffer[1024];

 /* Buffer[1024] is source data buffer, with data to be written */
 retval = FlsLoader_Write((uint32)FlsLoader_Address, WriteLength, &Buffer[0]);

 /* Check if data written successfully */
 if(retval == FLSLOADER_E_OK)
 {
 /* FlsLoader_Write API returned with FLSLOADER_E_OK. Data written successfully */
 }
}
```

Erase and write an user configuration block (UCB)

User can update an UCB using the FLSLOADER erase and write APIs.

For erasing an UCB, the start address should be a valid UCB sector address and length should be 1 sector.

For writing an UCB, the start address should be a valid UCB sector address and length should be size of a UCB sector

(512 bytes) and data should contain valid confirmation code on specified offset (refer to **UCB Address Map** in the hardware user manual).

Before updating an UCB user shall ensure that the UCB is not protected or is in ERRORED state. Refer to section **UCB Confirmation** in the hardware user manual to know about UCB confirmation states.

While updating a dual UCB user should follow sequence mentioned in **Dual Password UCB ORIG and COPY Re-programming** under section **UCB Confirmation** in the hardware user manual.

Sample code to erase and write UCB_DFLASH_ORIG and UCB_DFLASH_COPY is as follows:

```
/* Erase UCB_DFLASH_COPY */

/* Erase Start address */
FlsLoader_AddressType FlsLoader_Address = 0xAF403200U;

/*Erase Length*/
FlsLoader_LengthType EraseLength = 1U;

retval = FlsLoader_Erase((uint32)FlsLoader_Address, EraseLength);

/* Check if UCB_DFLASH_COPY erased successfully */
if(retval == FLSLOADER_E_OK)
{
 /* FlsLoader_Erase API returned with FLSLOADER_E_OK. UCB_DFLASH_COPY erased successfully */

 /* Writing UCB_DFLASH_COPY */

 /*Write Start Address*/
 FlsLoader_Address = 0xAF403200U;

 /*Write Length*/
 FlsLoader_LengthType WriteLength = 512U;

uint8 Buffer[512];

 /* Buffer[512] is source data buffer, with UCB data*/
 retval = FlsLoader_Write((uint32)FlsLoader_Address, WriteLength, &Buffer[0]);

 /* Check if UCB_DFLASH_COPY written successfully */
 if(retval == FLSLOADER_E_OK)
 {
 /* FlsLoader_Write API returned with FLSLOADER_E_OK. Data written successfully to
UCB_DFLASH_COPY*/
 }
}

/* Erase UCB_DFLASH_ORIG */

/* Erase Start address */
FlsLoader_Address = 0xAF402200U;

/*Erase Length*/
EraseLength = 1U;

retval = FlsLoader_Erase((uint32)FlsLoader_Address, EraseLength);

/* Check if UCB_DFLASH_ORIG erased successfully */
if(retval == FLSLOADER_E_OK)
{
 /* FlsLoader_Erase API returned with FLSLOADER_E_OK. UCB_DFLASH_ORIG erased successfully */
```

```
    /* Writing UCB_DFLASH_ORIG */

    /*Write Start Address*/
    FlsLoader_Address = 0xAF402200U;

    /*Write Length*/
    WriteLength = 512U;

    /* Buffer[512] is source data buffer with UCB data*/
    retval = FlsLoader_Write((uint32)FlsLoader_Address, WriteLength, &Buffer[0]);

    /* Check if UCB_DFLASH_ORIG written successfully */
    if(retval == FLSLOADER_E_OK)
    {
    /* FlsLoader_Write API returned with FLSLOADER_E_OK. Data written successfully to
   UCB_DFLASH_ORIG*/
    }
   }
```

FLSLOADER lock operation

The lock API installs the protections configured for the Flash banks during pre-compile time.

DFlash0 protections are configurable at bank level. Supported protections for DFlash0 are read and write protections.

PFlash protections are configurable at sector level. Supported protections for PFlash are write protection, write once protection (WOP), one time programmable (OTP) protection.

A controller reset is required after execution of lock API for the installed protections to become effective in the hardware.

If callout functionality is enabled by the user, the driver invokes the callout function to application at configured timeout intervals while executing the lock API.

Following sequence diagram depicts FLSLOADER lock operation with callout enabled.

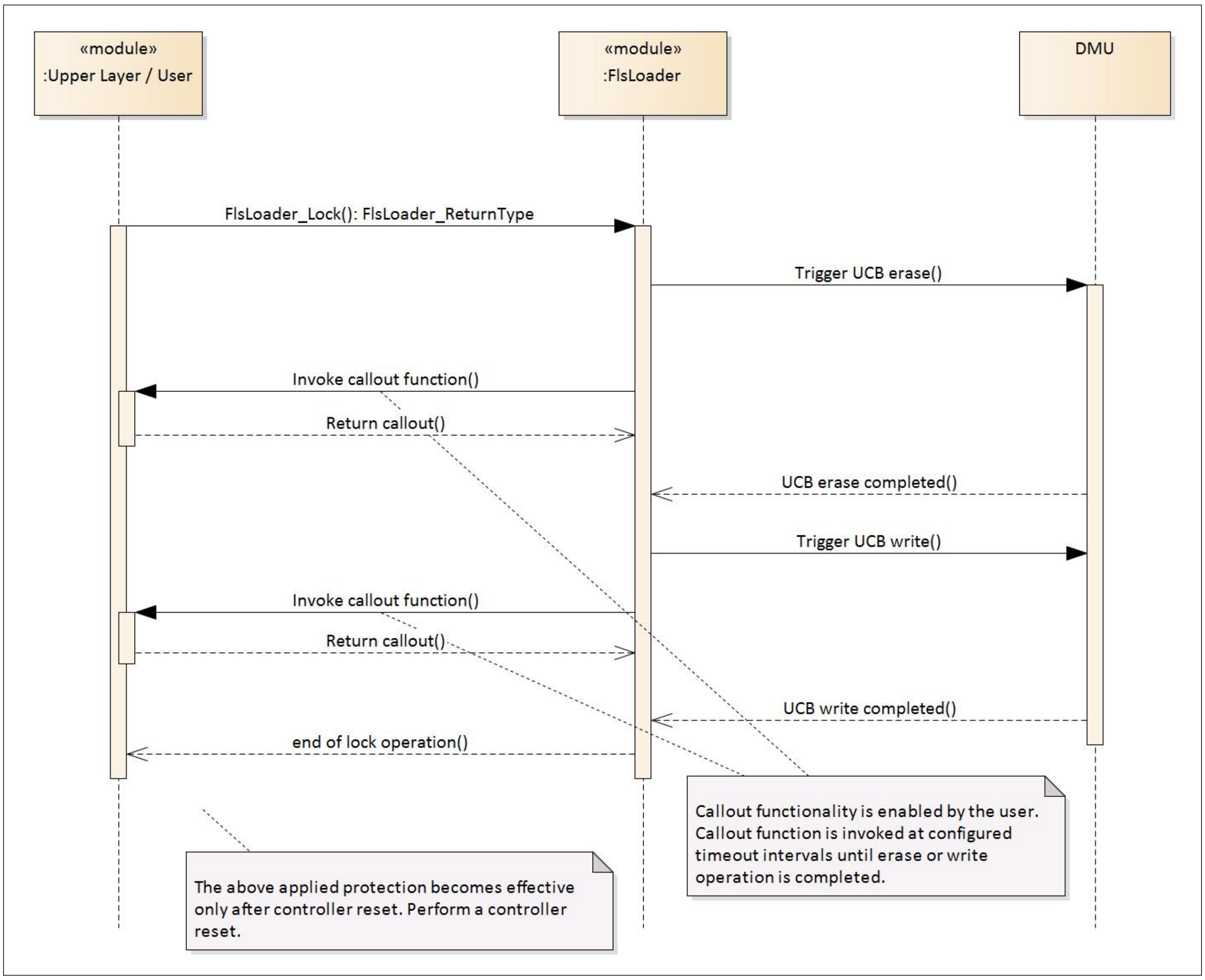
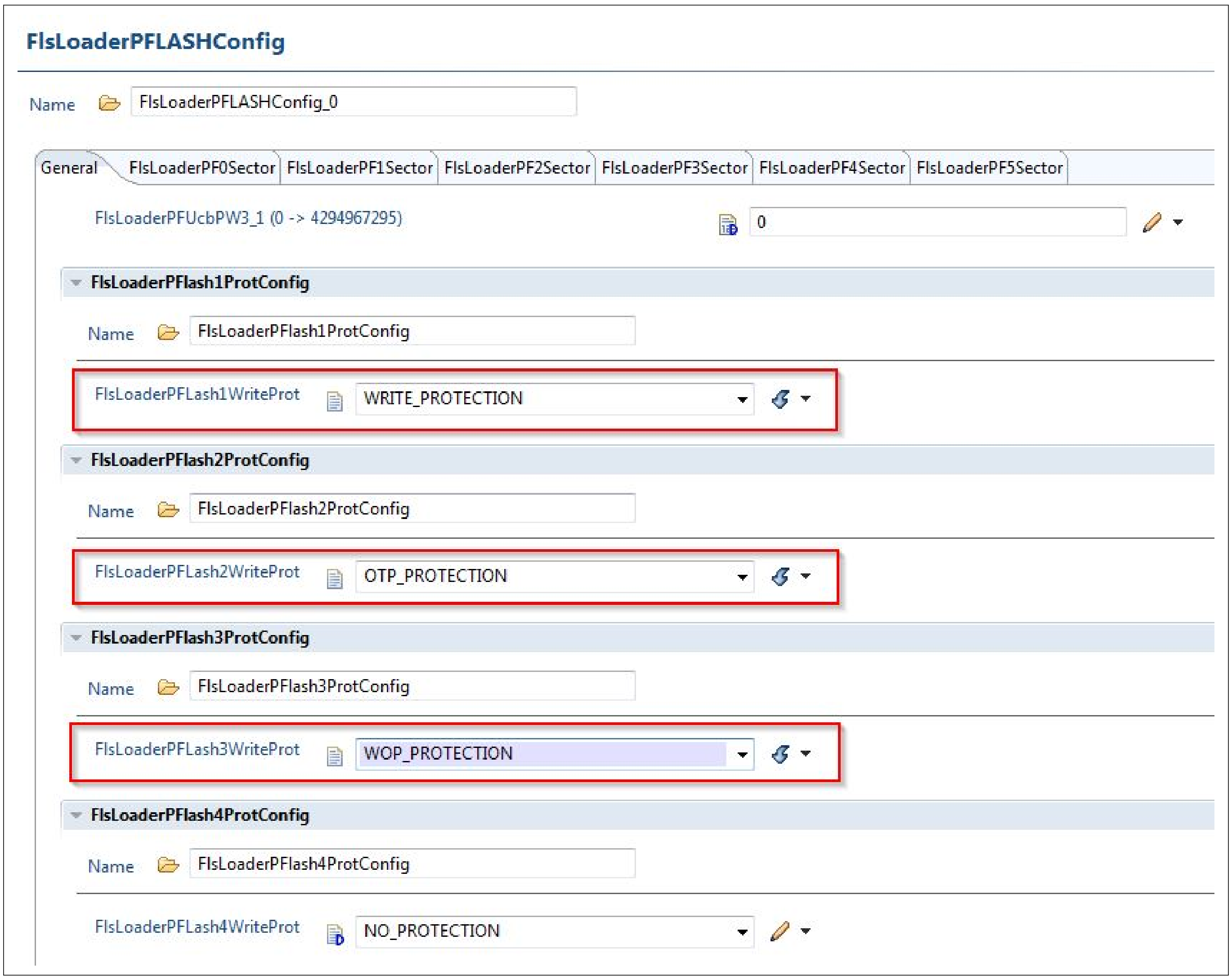

«module»
:Upper Layer / User
«module»
:FlsLoader
DMU
FlsLoader_Lock(): FlsLoader_ReturnType
Trigger UCB erase()
Invoke callout function()
Return callout()
UCB erase completed()
Trigger UCB write()
Invoke callout function()
Return callout()
UCB write completed()
end of lock operation()
Callout functionality is enabled by the user. Callout function is invoked at configured timeout intervals until erase or write operation is completed.
The above applied protection becomes effective only after controller reset. Perform a controller reset.

Figure 8 **Sequence for FLSLOADER lock operation with callout enabled**

If requirement is to enable following protections for Flash banks:
- Write protection for sector 0 of PFlash bank 1
- OTP protection for sector 0 of PFlash bank 2
- WOP protection for sector 0 of PFlash bank 3
- Write protection for DFlash bank 0

Then following configuration should be set in the configuration tool:


FlsLoaderPFLASHConfig

Name: FlsLoaderPFLASHConfig_0

FlsLoaderPFUcbPW3_1 (0 -> 4294967295): 0

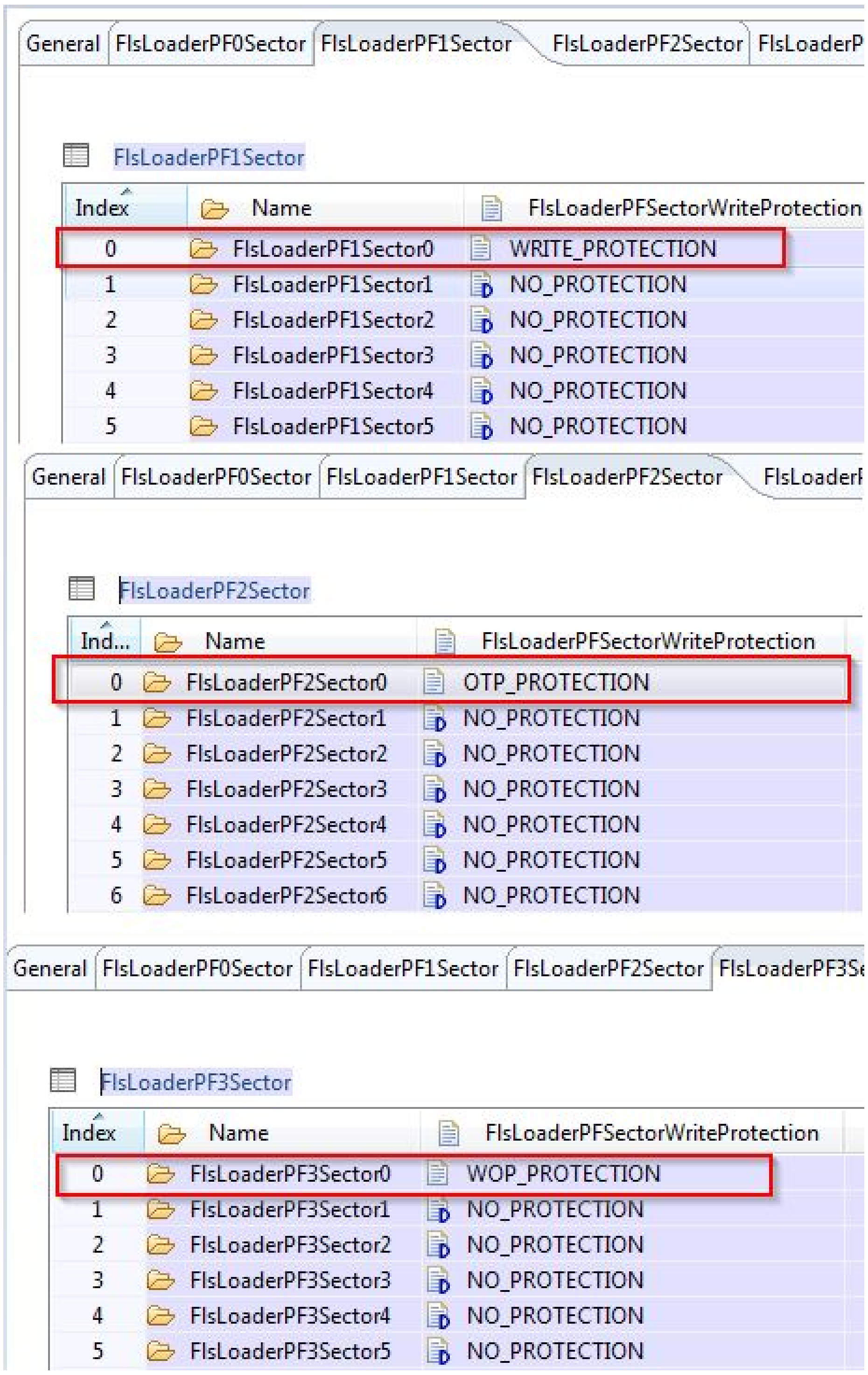
FlsLoaderPFlash1ProtConfig
Name: FlsLoaderPFlash1ProtConfig
FlsLoaderPFLash1WriteProt: WRITE_PROTECTION

FlsLoaderPFlash2ProtConfig
Name: FlsLoaderPFlash2ProtConfig
FlsLoaderPFLash2WriteProt: OTP_PROTECTION

FlsLoaderPFlash3ProtConfig
Name: FlsLoaderPFlash3ProtConfig
FlsLoaderPFLash3WriteProt: WOP_PROTECTION

FlsLoaderPFlash4ProtConfig
Name: FlsLoaderPFlash4ProtConfig
FlsLoaderPFLash4WriteProt: NO_PROTECTION

Figure 9 **Step1: Select PFlash 1, PFlash 2, PFlash 3 bank protections as write, OTP and WOP protections**



FlsLoaderPF1Sector

Index	Name	FlsLoaderPFSectorWriteProtection
0	FlsLoaderPF1Sector0	WRITE_PROTECTION
1	FlsLoaderPF1Sector1	NO_PROTECTION
2	FlsLoaderPF1Sector2	NO_PROTECTION
3	FlsLoaderPF1Sector3	NO_PROTECTION
4	FlsLoaderPF1Sector4	NO_PROTECTION
5	FlsLoaderPF1Sector5	NO_PROTECTION

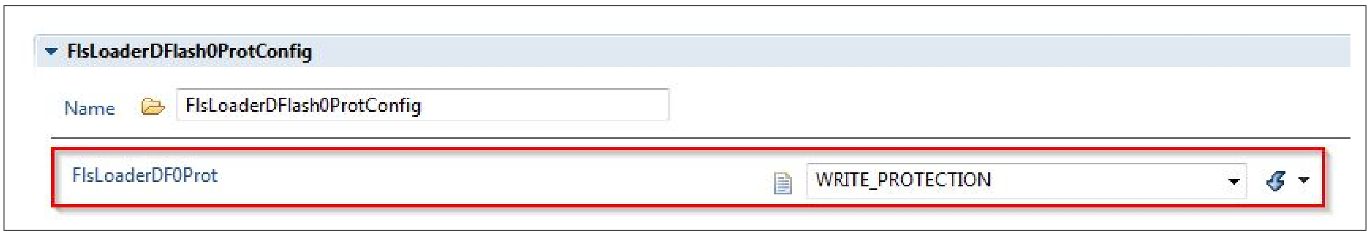
FlsLoaderPF2Sector

Ind...	Name	FlsLoaderPFSectorWriteProtection
0	FlsLoaderPF2Sector0	OTP_PROTECTION
1	FlsLoaderPF2Sector1	NO_PROTECTION
2	FlsLoaderPF2Sector2	NO_PROTECTION
3	FlsLoaderPF2Sector3	NO_PROTECTION
4	FlsLoaderPF2Sector4	NO_PROTECTION
5	FlsLoaderPF2Sector5	NO_PROTECTION
6	FlsLoaderPF2Sector6	NO_PROTECTION

FlsLoaderPF3Sector

Index	Name	FlsLoaderPFSectorWriteProtection
0	FlsLoaderPF3Sector0	WOP_PROTECTION
1	FlsLoaderPF3Sector1	NO_PROTECTION
2	FlsLoaderPF3Sector2	NO_PROTECTION
3	FlsLoaderPF3Sector3	NO_PROTECTION
4	FlsLoaderPF3Sector4	NO_PROTECTION
5	FlsLoaderPF3Sector5	NO_PROTECTION

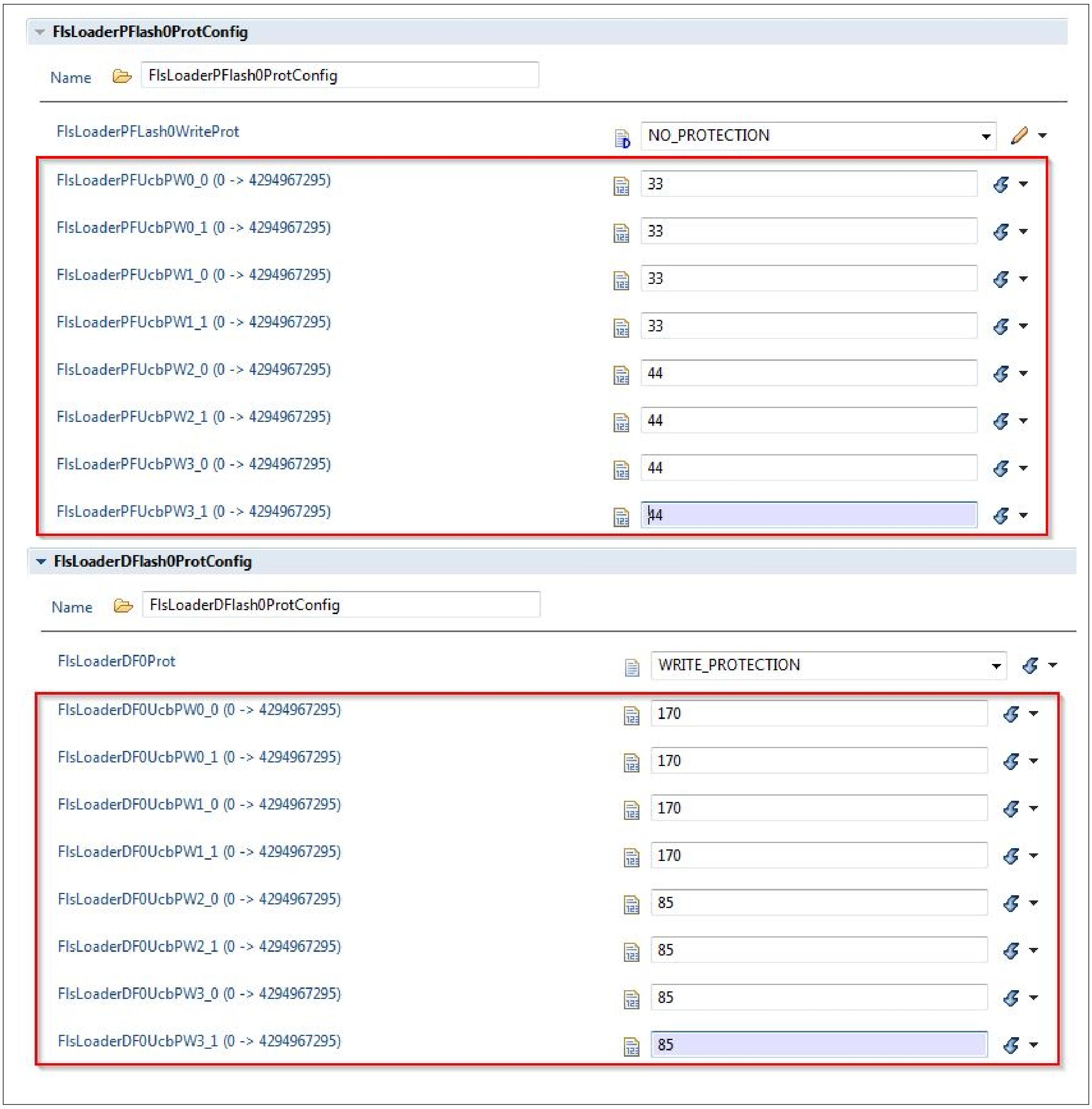
Figure 10 **Step2: Select sector 0 of PFlash 1, PFlash 2, PFlash 3 banks as write, OTP and WOP protections**



FlsLoaderDFlash0ProtConfig

Name	FlsLoaderDFlash0ProtConfig
FlsLoaderDF0Prot	WRITE_PROTECTION

Figure 11 **Step3: Select DFlash0 bank protection as write protection**



FlsLoaderPFlash0ProtConfig

Name	FlsLoaderPFlash0ProtConfig
FlsLoaderPFLash0WriteProt	NO_PROTECTION
FlsLoaderPFUcbPW0_0 (0 -> 4294967295)	33
FlsLoaderPFUcbPW0_1 (0 -> 4294967295)	33
FlsLoaderPFUcbPW1_0 (0 -> 4294967295)	33
FlsLoaderPFUcbPW1_1 (0 -> 4294967295)	33
FlsLoaderPFUcbPW2_0 (0 -> 4294967295)	44
FlsLoaderPFUcbPW2_1 (0 -> 4294967295)	44
FlsLoaderPFUcbPW3_0 (0 -> 4294967295)	44
FlsLoaderPFUcbPW3_1 (0 -> 4294967295)	44

FlsLoaderDFlash0ProtConfig

Name	FlsLoaderDFlash0ProtConfig
FlsLoaderDF0Prot	WRITE_PROTECTION
FlsLoaderDF0UcbPW0_0 (0 -> 4294967295)	170
FlsLoaderDF0UcbPW0_1 (0 -> 4294967295)	170
FlsLoaderDF0UcbPW1_0 (0 -> 4294967295)	170
FlsLoaderDF0UcbPW1_1 (0 -> 4294967295)	170
FlsLoaderDF0UcbPW2_0 (0 -> 4294967295)	85
FlsLoaderDF0UcbPW2_1 (0 -> 4294967295)	85
FlsLoaderDF0UcbPW3_0 (0 -> 4294967295)	85
FlsLoaderDF0UcbPW3_1 (0 -> 4294967295)	85

Figure 12 **Step4: Configure passwords PFlash and DFlash0 access**

Sample code to depict lock operation is as follows:

```
/* FlsLoader_Lock execution starts.. */
retval = FlsLoader_Lock();

/*Lock executed successfully*/
if(retval == FLSLOADER_E_OK)
{
 /* FlsLoader_Lock Execution Passed */
 /* Trigger controller reset for HW to install protections */
}

/* Following part should be executed after micro-controller reset */

/* Erasing the Sector-0 of PFlash Bank-1 where write Protection is enabled */

/*Address of Sector 0 of PFlash bank 1*/
FlsLoader_AddressType FlsLoader_Address = 0xA0300000U;

/*Number of sectors to be erased is 1*/
FlsLoader_LengthType Length = 1U;

retval = FlsLoader_Erase(FlsLoader_Address, Length);

/* Check if erase operation returned with FLSLOADER_E_LOCKED */
if (retval == FLSLOADER_E_LOCKED)
{
 /* FlsLoader_Erase returned FLSLOADER_E_LOCKED. Sector is locked. */
}
```

FLSLOADER unlock operation

The unlock API is used to temporarily (until next controller reset) disable the protections installed by lock API.

Only read and write protections installed by the lock API are disabled. OTP and WOP protections can't be disabled.

Sequence for FLSLOADER unlock operation is depicted as below:

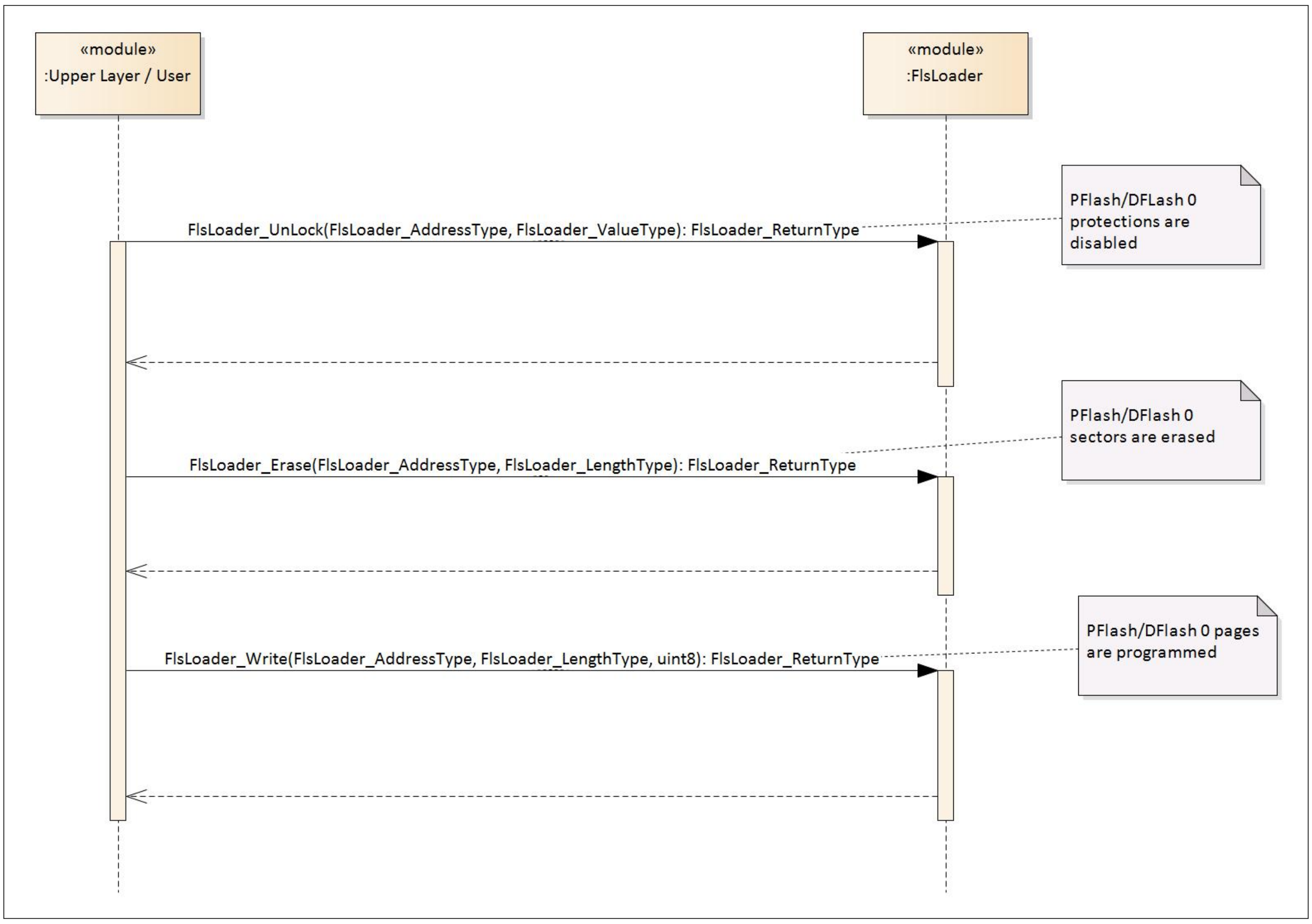

«module»
:Upper Layer / User
«module»
:FlsLoader
PFlash/DFLash 0 protections are disabled
FlsLoader_UnLock(FlsLoader_AddressType, FlsLoader_ValueType): FlsLoader_ReturnType
PFlash/DFlash 0 sectors are erased
FlsLoader_Erase(FlsLoader_AddressType, FlsLoader_LengthType): FlsLoader_ReturnType
PFlash/DFlash 0 pages are programmed
FlsLoader_Write(FlsLoader_AddressType, FlsLoader_LengthType, uint8): FlsLoader_ReturnType

Figure 13 **Sequence for FLSLOADER unlock operation**

Sample code to depict FLSLOADER unlock operation is as follows:

```
/* FlsLoader_Unlock execution starts.. */

/* Unlocking the Write Protections installed for PFlash sectors.*/

/* PFLASH_ORIG ucb address*/
FlsLoader_AddressType FlsLoader_Address = 0xAF402000U;

FlsLoader_ValueType PfPassword[8];

/* PfPassword[8] is 8-word password for PFlash */
retval = FlsLoader_Unlock(FlsLoader_Address, &PfPassword[0]);

/* Check if PFlash Write protection Disabled */
if(retval == FLSLOADER_E_OK)
{

 /* Write Protections installed for PFlash sectors are disabled*/
 /* Erasing the Sector-0 of PFlash Bank-1 where write Protection is Disabled.*/

 /*Address of Sector 0 of PFlash bank 1*/
 FlsLoader_AddressType FlsLoader_Address = 0xA0300000U;

 /*Number of sectors to be erased is 1*/
 FlsLoader_LengthType Length = 1U;

 retval = FlsLoader_Erase(FlsLoader_Address, Length);

 /* Check if sector erased successfully */
 if (retval == FLSLOADER_E_OK)
 {
 /* FlsLoader_Erase returned FLSLOADER_E_OK. Sector erased successfully.*/
 }
}
```

Load FLSLOADER write and erase routines to RAM

- In driver code, the FLSLOADER write and erase routines alone are encapsulated under memory map section FLSLOADER_START_SEC_WRITEERASE_CODE_QM_LOCAL and FLSLOADER_STOP_SEC_WRITEERASE_CODE_QM_LOCAL. The constant data required for the write and erase routines are encapsulated under memory map section FLSLOADER_START_SEC_CONST_QM_LOCAL_32 and FLSLOADER_STOP_SEC_CONST_QM_LOCAL_32.
- The dependent MCALLIB APIs Mcal_DelayTickResolution() and Mcal_DelayGetTick() for write and erase routines are encapsulated under memory map section MCALLIB_START_SEC_CODE_ASIL_B_GLOBAL and MCALLIB_STOP_SEC_CODE_ASIL_B_GLOBAL
- In order to load complete write and erase routines to RAM, above mentioned sections need to be moved to RAM. This can be done using linker settings. Copying of RAM can be done either by user application or can make use of the init functions provided by the compiler.
- The DET should be disabled if FLSLOADER write and erase routines are executed from the RAM.
- If callout functionality is enabled, user shall ensure that configured callout function is placed in the RAM.

E_NOT_OK return value for FlsLoader_Erase and FlsLoader_Write API

FlsLoader_Erase and FlsLoader_Write API operation may return E_NOT_OK under either of the following conditions.

- Hardware operation takes more time than the calculated timeout value for the operation
- Hardware reports an operational error while an operation is being performed

In such a situation, if the hardware continues to be in the busy state then further calls to any of the following APIs FlsLoader_DeInit, FlsLoader_Write, FlsLoader_Erase, FlsLoader_Lock, FlsLoader_UnLock will cause FLSLOADER_E_BUSY to be returned. As hardware continues to be in busy state a hardware reset is required.

Exclusive area for PFlash erase and write operation

- Safety Endint protection is disabled for the duration of PFlash erase and write command cycle execution. These operations are performed within an exclusive area enforced with the help of following exclusive area interfaces :-

  FlsLdr_ExclArea_PfProg_Enter()

  FlsLdr_ExclArea_PfProg_Exit()

  FlsLdr_ExclArea_PfErase_Enter()

  FlsLdr_ExclArea_PfErase_Exit()

  Provided by files FlsLdr_ExclArea.h and FlsLdr_ExclArea.c.
- The user needs to implement these functions so as to prevent the interruption of operations within the exclusive area. This will allow the code in exclusive area to execute quickly and not reach a safety watchdog timeout condition on the disabling of Safety Endinit protection. For above mentioned functions, please refer following example code.

```
void FlsLdr_ExclArea_PfProg_Enter(void)
{
 SuspendAllInterrupts();
}

void FlsLdr_ExclArea_PfProg_Exit(void)
{
 ResumeAllInterrupts();
}

void FlsLdr_ExclArea_PfErase_Enter(void)
{
 SuspendAllInterrupts();
}

void FlsLdr_ExclArea_PfErase_Exit(void)
{
 ResumeAllInterrupts();
}
```

1.1.5 Key architectural considerations

1.1.5.1 User mode is not supported by the driver

The FLSLOADER driver does not support user mode configuration for any of its APIs. The driver is meant to be used in the boot loader application for Flash programming. Boot loader is not a part of the MCAL runtime component. Therefore, all APIs of the driver shall be executed in the supervisory mode.

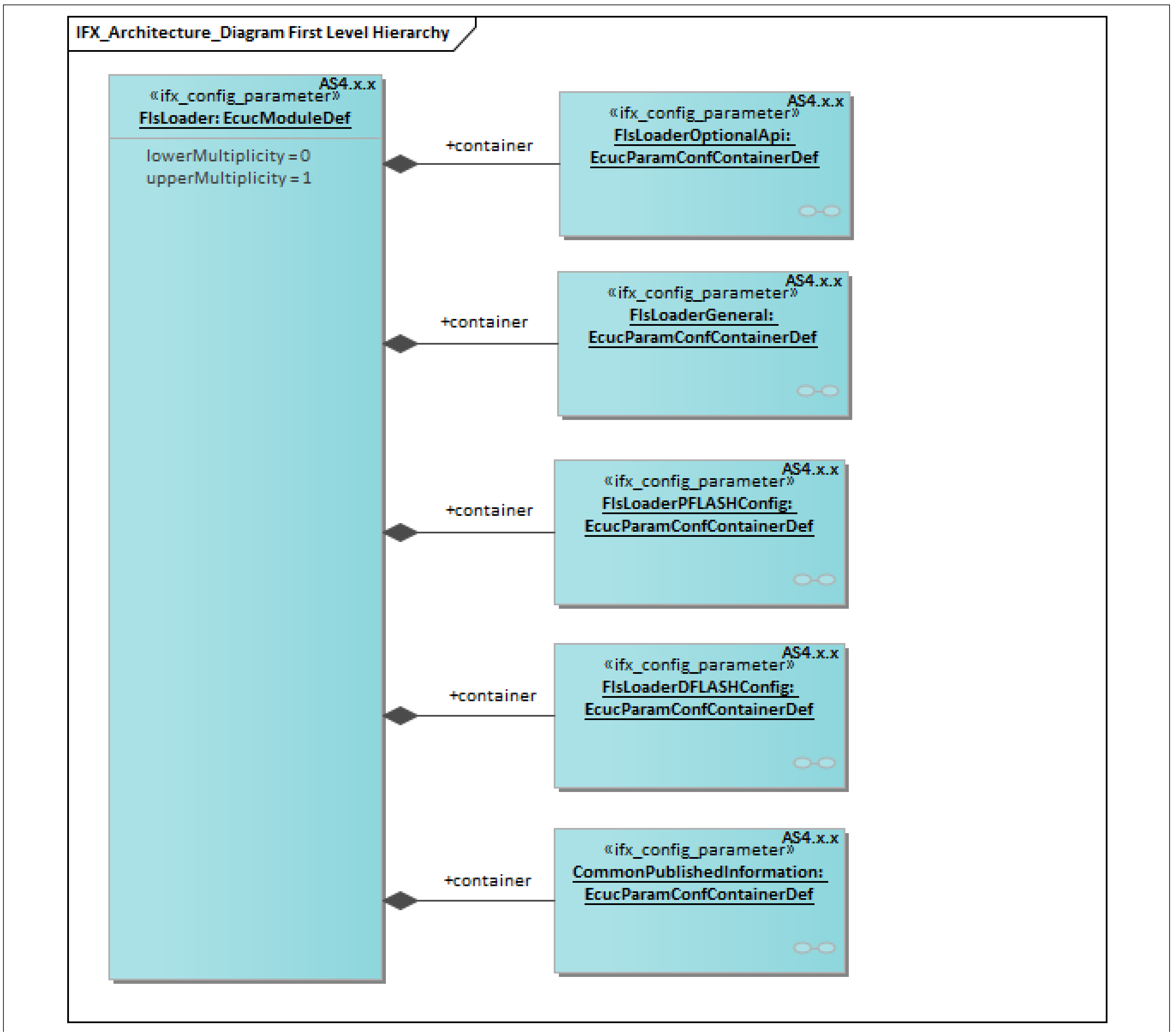
[cover parentID FLSLOADER={DC8DF695-F796-4e0e-9514-DEF2CE8C2AA4}]

1.2 Assumptions of Use (AoU)

There are no AoU for the FlsLoader driver.

1.3 Reference information

1.3.1 Configuration interfaces


IFX_Architecture_Diagram First Level Hierarchy
AS4.x.x
«ifx_config_parameter»
FlsLoader: EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
AS4.x.x
«ifx_config_parameter»
FlsLoaderOptionalApi: EcucParamConfContainerDef
+container
AS4.x.x
«ifx_config_parameter»
FlsLoaderGeneral: EcucParamConfContainerDef
+container
AS4.x.x
«ifx_config_parameter»
FlsLoaderPFLASHConfig: EcucParamConfContainerDef
+container
AS4.x.x
«ifx_config_parameter»
FlsLoaderDFLASHConfig: EcucParamConfContainerDef
+container
AS4.x.x
«ifx_config_parameter»
CommonPublishedInformation: EcucParamConfContainerDef

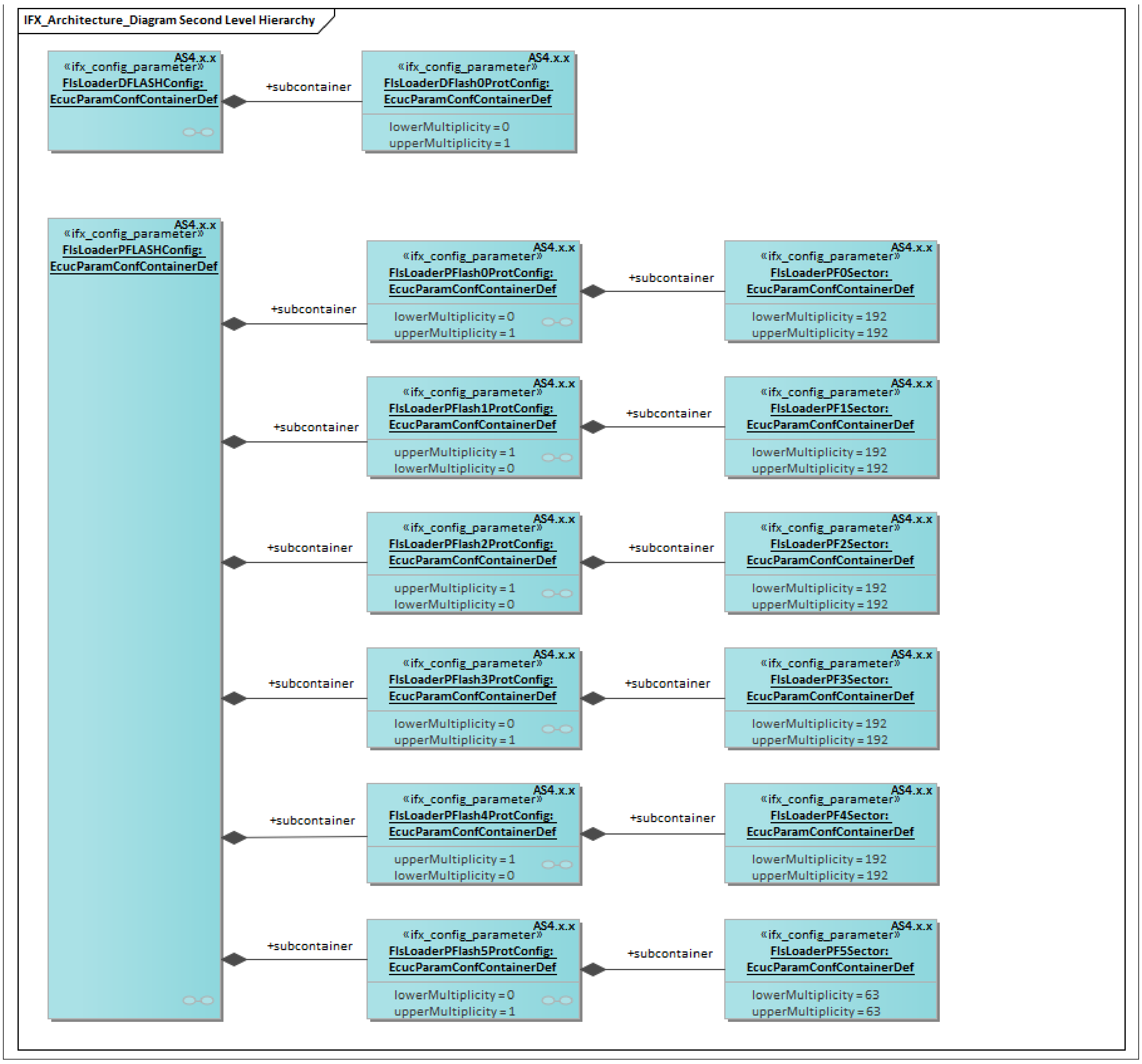

IFX_Architecture_Diagram Second Level Hierarchy
AS4.x.x «ifx_config_parameter» FlsLoaderDFLASHConfig: EcucParamConfContainerDef
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderDFlash0ProtConfig: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
AS4.x.x «ifx_config_parameter» FlsLoaderPFLASHConfig: EcucParamConfContainerDef
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderPFlash0ProtConfig: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderPF0Sector: EcucParamConfContainerDef
lowerMultiplicity = 192
upperMultiplicity = 192
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderPFlash1ProtConfig: EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 0
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderPF1Sector: EcucParamConfContainerDef
lowerMultiplicity = 192
upperMultiplicity = 192
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderPFlash2ProtConfig: EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 0
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderPF2Sector: EcucParamConfContainerDef
lowerMultiplicity = 192
upperMultiplicity = 192
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderPFlash3ProtConfig: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderPF3Sector: EcucParamConfContainerDef
lowerMultiplicity = 192
upperMultiplicity = 192
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderPFlash4ProtConfig: EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 0
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderPF4Sector: EcucParamConfContainerDef
lowerMultiplicity = 192
upperMultiplicity = 192
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderPFlash5ProtConfig: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subcontainer
AS4.x.x «ifx_config_parameter» FlsLoaderPF5Sector: EcucParamConfContainerDef
lowerMultiplicity = 63
upperMultiplicity = 63

Figure 14 **Container hierarchy along with their configuration parameters**

1.3.1.1 Container: CommonPublishedInformation

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.1.1 ArMajorVersion

Table 4 **Specification for ArMajorVersion**

Name	`ArMajorVersion`		
Description	This parameter provides the major version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef

Table 4 Specification for ArMajorVersion (continued)

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	This parameter provides the minor version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the 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`		
Description	This parameter provides the patch version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the selected AUTOSAR version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 6 Specification for ArPatchVersion (continued)

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	This parameter provides the Module ID.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	255		
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	The configuration parameter defines the TC3xx derivative used for the implementation.		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	As per selected TC3xx derivative		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL

Table 8 Specification for Release (continued)

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	This parameter provides the major version of the software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
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	This parameter provides the minor version of the software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
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.		

1.3.1.1.8 SwPatchVersion

Table 11 Specification for SwPatchVersion

Name	`SwPatchVersion`		
Description	This parameter provides the patch version of the software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
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 VendorId

Table 12 Specification for VendorId

Name	`VendorId`		
Description	This parameter provides the Vendor ID.		
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: FlsLoader

Configuration of the FLSLOADER driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.3 Container: FlsLoaderDFlash0ProtConfig

Container for configuring DFlash bank 0 protection. Container is available only if DFlash bank 0 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.3.1 FlsLoaderDF0Prot

Table 13 Specification for FlsLoaderDF0Prot

Name	`FlsLoaderDF0Prot`		
Description	Configures DFlash bank 0 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection READ_PROTECTION: Read protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
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.		

1.3.1.3.2 FlsLoaderDF0UcbPW0_0

Table 14 Specification for FlsLoaderDF0UcbPW0_0

Name	`FlsLoaderDF0UcbPW0_0`		
Description	PW0: First least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 14 Specification for FlsLoaderDF0UcbPW0_0 (continued)

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderDF0Prot		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.3 FlsLoaderDF0UcbPW0_1

Table 15 Specification for FlsLoaderDF0UcbPW0_1

Name	`FlsLoaderDF0UcbPW0_1`		
Description	PW1: Second least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderDF0Prot		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.4 FlsLoaderDF0UcbPW1_0

Table 16 Specification for FlsLoaderDF0UcbPW1_0

Name	`FlsLoaderDF0UcbPW1_0`		
Description	PW2: Third least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 16 Specification for FlsLoaderDF0UcbPW1_0 (continued)

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderDF0Prot		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.5 FlsLoaderDF0UcbPW1_1

Table 17 Specification for FlsLoaderDF0UcbPW1_1

Name	`FlsLoaderDF0UcbPW1_1`		
Description	PW3: Fourth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderDF0Prot		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.6 FlsLoaderDF0UcbPW2_0

Table 18 Specification for FlsLoaderDF0UcbPW2_0

Name	`FlsLoaderDF0UcbPW2_0`		
Description	PW4: Fifth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 18 Specification for FlsLoaderDF0UcbPW2_0 (continued)

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderDF0Prot		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.7 FlsLoaderDF0UcbPW2_1

Table 19 Specification for FlsLoaderDF0UcbPW2_1

Name	`FlsLoaderDF0UcbPW2_1`		
Description	PW5: Sixth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderDF0Prot		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.8 FlsLoaderDF0UcbPW3_0

Table 20 Specification for FlsLoaderDF0UcbPW3_0

Name	`FlsLoaderDF0UcbPW3_0`		
Description	PW6: Seventh least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 20 **Specification for FlsLoaderDF0UcbPW3_0 (continued)**

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderDF0Prot		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.9 FlsLoaderDF0UcbPW3_1

Table 21 **Specification for FlsLoaderDF0UcbPW3_1**

Name	`FlsLoaderDF0UcbPW3_1`		
Description	PW7: Eighth least significant 32-bit word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderDF0Prot		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4 Container: FlsLoaderDFLASHConfig

This container contains the configuration parameters and sub-containers for configuration of DFlash.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.5 Container: FlsLoaderGeneral

This container contains the general configuration parameters of the FLSLOADER driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.5.1 FlsLoaderCallOutFunction

Table 22 Specification for FlsLoaderCallOutFunction

Name	`FlsLoaderCallOutFunction`		
Description	FLSLOADER provides a call out function to the user while performing Flash write and erase operations using its APIs. It is invoked at every user defined time rate (FlsLoaderCallOutTime). Call out function should be defined by the user using this parameter. User can configure either the name or the valid address of the call out function in Flash. Call out function is enabled only if valid value as mentioned above is configured for the parameter. By default this parameter contains NULL_PTR and call out function is disabled to reduce the executable code size. When call out is enabled user should configure call out time using parameter FlsLoaderCallOutTime.		
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.		

1.3.1.5.2 FlsLoaderCallOutTime

Table 23 Specification for FlsLoaderCallOutTime

Name	`FlsLoaderCallOutTime`		
Description	Specifies the maximum time in nanoseconds to wait before invoking call out function to application while looping for status during write and erase operations. The default value of this parameter is set to 5000000 as an example value within the range. This parameter is valid only if FlsLoaderCallOutFuntion is configured with a value other than NULL_PTR.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	10000 - 4294967295		
Default value	5000000		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 23 Specification for FlsLoaderCallOutTime (continued)

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

1.3.1.5.3 FlsLoaderDevErrorDetect

Table 24 Specification for FlsLoaderDevErrorDetect

Name	`FlsLoaderDevErrorDetect`		
Description	Switch for enabling the development error tracer (DET). True: DET enabled False: DET 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	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.4 FlsLoaderEnableLockCheck

Table 25 Specification for FlsLoaderEnableLockCheck

Name	`FlsLoaderEnableLockCheck`		
Description	Switch for enabling the Lock-check functionality. True: Enable the Lock-check routine in write /erase APIs False: Disable the Lock-check routine in write/erase APIs This optional feature is disabled by default to reduce the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef

Table 25 Specification for FlsLoaderEnableLockCheck (continued)

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 Container: FlsLoaderOptionalApi

This container contains the configuration parameters for optional APIs of the FLSLOADER driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.6.1 FlsLoaderDeInitApi

Table 26 Specification for FlsLoaderDeInitApi

Name	`FlsLoaderDeInitApi`		
Description	Switch for enabling the FlsLoader_DeInit API. True: FlsLoader_DeInit API enabled False: FlsLoader_DeInit API disabled This optional API is disabled by default to reduce 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	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.2 FlsLoaderLockUnlockApi

Table 27 Specification for FlsLoaderLockUnlockApi

Name	`FlsLoaderLockUnlockApi`		
Description	Switch for enabling the FlsLoader_Lock and FlsLoader_Unlock APIs. True: FlsLoader_Lock and FlsLoader_Unlock APIs enabled False: FlsLoader_Lock and FlsLoader_Unlock APIs disabled These optional APIs are disabled by default to reduce 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	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.3 FlsLoaderVersionInfoApi

Table 28 Specification for FlsLoaderVersionInfoApi

Name	`FlsLoaderVersionInfoApi`		
Description	Switch for enabling the FlsLoaderVersionInfo API. True: FlsLoaderVersionInfo API enabled False: FlsLoaderVersionInfo API disabled This optional API is disabled by default to reduce 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**	-

Table 28 Specification for FlsLoaderVersionInfoApi (continued)

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

1.3.1.7 Container: FlsLoaderPF0Sector

Container for configuration of PFlash bank 0 sectors

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.7.1 FlsLoaderPFSectorWriteProtection

Table 29 Specification for FlsLoaderPFSectorWriteProtection

Name	`FlsLoaderPFSectorWriteProtection`		
Description	Configuration of PFlash bank 0 sector protection. Any active protection for the sector shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash0WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8 Container: FlsLoaderPF1Sector

Container for configuration of PFlash bank 1 sectors.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.8.1 FlsLoaderPFSectorWriteProtection

Table 30 Specification for FlsLoaderPFSectorWriteProtection

Name	`FlsLoaderPFSectorWriteProtection`		
Description	Configuration of PFlash bank 1 sector protection. Any active protection for the sector shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash1WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9 Container: FlsLoaderPF2Sector

Container for configuration of PFlash bank 2 sectors.

Post-Build Variant Multiplicity: -
Multiplicity Configuration Class: -

1.3.1.9.1 FlsLoaderPFSectorWriteProtection

Table 31 Specification for FlsLoaderPFSectorWriteProtection

Name	`FlsLoaderPFSectorWriteProtection`		
Description	Configuration of PFlash bank 2 sector protection. Any active protection for the sector shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected		

Table 31 **Specification for FlsLoaderPFSectorWriteProtection (continued)**

	WRITE_PROTECTION: Write protection		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash2WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10 Container: FlsLoaderPF3Sector

Container for configuration of PFlash bank 3 sectors.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.10.1 FlsLoaderPFSectorWriteProtection

Table 32 **Specification for FlsLoaderPFSectorWriteProtection**

Name	`FlsLoaderPFSectorWriteProtection`		
Description	Configuration of PFlash bank 3 sector protection. Any active protection for the sector shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash3WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.11 Container: FlsLoaderPF4Sector

Container for configuration of PFlash bank 4 sectors.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.11.1 FlsLoaderPFSectorWriteProtection

Table 33 Specification for FlsLoaderPFSectorWriteProtection

Name	`FlsLoaderPFSectorWriteProtection`		
Description	Configuration of PFlash bank 4 sector protection. Any active protection for the sector shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash4WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12 Container: FlsLoaderPF5Sector

Container for configuration of PFlash bank 5 sectors.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.12.1 FlsLoaderPFSectorWriteProtection

Table 34 Specification for FlsLoaderPFSectorWriteProtection

Name	`FlsLoaderPFSectorWriteProtection`
Description	Configuration of PFlash bank 5 sector protection. Any active protection for the sector shall be selected by the user as per application need. Therefore default value is provided as no protection.

Table 34 Specification for FlsLoaderPFSectorWriteProtection (continued)

Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash5WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13 Container: FlsLoaderPFlash0ProtConfig

Container for configuring PFlash bank 0 protection. Container is available only if PFlash bank 0 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.13.1 FlsLoaderPFLash0WriteProt

Table 35 Specification for FlsLoaderPFLash0WriteProt

Name	`FlsLoaderPFLash0WriteProt`		
Description	Configuration of PFlash bank 0 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 35 **Specification for FlsLoaderPFLash0WriteProt (continued)**

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.		

1.3.1.13.2 FlsLoaderPFUcbPW0_0

Table 36 **Specification for FlsLoaderPFUcbPW0_0**

Name	`FlsLoaderPFUcbPW0_0`		
Description	PW0: First least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash0WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.3 FlsLoaderPFUcbPW0_1

Table 37 **Specification for FlsLoaderPFUcbPW0_1**

Name	`FlsLoaderPFUcbPW0_1`		
Description	PW1: Second least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 37 Specification for FlsLoaderPFUcbPW0_1 (continued)

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash0WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.4 FlsLoaderPFUcbPW1_0

Table 38 Specification for FlsLoaderPFUcbPW1_0

Name	`FlsLoaderPFUcbPW1_0`		
Description	PW2: Third least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash0WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.5 FlsLoaderPFUcbPW1_1

Table 39 Specification for FlsLoaderPFUcbPW1_1

Name	`FlsLoaderPFUcbPW1_1`		
Description	PW3: Fourth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 39 **Specification for FlsLoaderPFUcbPW1_1 (continued)**

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash0WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.6 FlsLoaderPFUcbPW2_0

Table 40 **Specification for FlsLoaderPFUcbPW2_0**

Name	`FlsLoaderPFUcbPW2_0`		
Description	PW4: Fifth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash0WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.7 FlsLoaderPFUcbPW2_1

Table 41 **Specification for FlsLoaderPFUcbPW2_1**

Name	`FlsLoaderPFUcbPW2_1`		
Description	PW5: Sixth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 41 Specification for FlsLoaderPFUcbPW2_1 (continued)

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash0WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.8 FlsLoaderPFUcbPW3_0

Table 42 Specification for FlsLoaderPFUcbPW3_0

Name	`FlsLoaderPFUcbPW3_0`		
Description	PW6: Seventh least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash0WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.9 FlsLoaderPFUcbPW3_1

Table 43 Specification for FlsLoaderPFUcbPW3_1

Name	`FlsLoaderPFUcbPW3_1`		
Description	PW7: Eighth least significant word of 256-bit password. Default value is 0 as the initial password is set to 0.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 4294967295		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 43 Specification for FlsLoaderPFUcbPW3_1 (continued)

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FlsLoaderPFLash0WriteProt		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.14 Container: FlsLoaderPFlash1ProtConfig

Container for configuring PFlash bank 1 protection. Container is available only if PFlash bank 1 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.14.1 FlsLoaderPFLash1WriteProt

Table 44 Specification for FlsLoaderPFLash1WriteProt

Name	`FlsLoaderPFLash1WriteProt`		
Description	Configuration of PFlash bank 1 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
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.		

1.3.1.15 Container: FlsLoaderPFlash2ProtConfig

Container for configuring PFlash bank 2 protection. Container is available only if PFlash bank 2 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.15.1 FlsLoaderPFLash2WriteProt

Table 45 Specification for FlsLoaderPFLash2WriteProt

Name	`FlsLoaderPFLash2WriteProt`		
Description	Configuration of PFlash bank 2 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
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.		

1.3.1.16 Container: FlsLoaderPFlash3ProtConfig

Container for configuring PFlash bank 3 protection. Container is available only if PFlash bank 3 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.16.1 FlsLoaderPFLash3WriteProt

Table 46 Specification for FlsLoaderPFLash3WriteProt

Name	`FlsLoaderPFLash3WriteProt`		
Description	Configuration of PFlash bank 3 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

Table 46 Specification for FlsLoaderPFLash3WriteProt (continued)

Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
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.		

1.3.1.17 Container: FlsLoaderPFlash4ProtConfig

Container for configuring PFlash bank 4 protection. Container is available only if PFlash bank 4 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.17.1 FlsLoaderPFLash4WriteProt

Table 47 Specification for FlsLoaderPFLash4WriteProt

Name	`FlsLoaderPFLash4WriteProt`		
Description	Configuration of PFlash bank 4 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-

Table 47 Specification for FlsLoaderPFLash4WriteProt (continued)

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

1.3.1.18 Container: FlsLoaderPFlash5ProtConfig

Container for configuring PFlash bank 5 protection. Container is available only if PFlash bank 5 is available in the TC3xx device selected by configuration else the container is not available.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.18.1 FlsLoaderPFLash5WriteProt

Table 48 Specification for FlsLoaderPFLash5WriteProt

Name	`FlsLoaderPFLash5WriteProt`		
Description	Configuration of PFlash bank 5 protection. Any active protection for the bank shall be selected by the user as per application need. Therefore default value is provided as no protection.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	NO_PROTECTION: No protection OTP_PROTECTION: OTP protected WOP_PROTECTION: WOP protected WRITE_PROTECTION: Write protected		
Default value	NO_PROTECTION		
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.		

1.3.1.19 Container: FlsLoaderPFLASHConfig

This container contains the configuration parameters and sub-containers for configuration of PFlash.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.2 Functions - Type definitions

This section lists all the data type of the FLSLOADER driver.

1.3.2.1 FlsLoader_AddressType

Table 49 Specification for FlsLoader_AddressType

Syntax	`FlsLoader_AddressType`	
Type	uint32	
File	`FlsLoader.h`	
Range	0 to 4294967295	Target address in Flash
Description	This type specifies the logical destination address of Flash in PFlash or DFlash.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.2 FlsLoader_CallOutFunc

Table 50 Specification for FlsLoader_CallOutFunc

Syntax	`FlsLoader_CallOutFunc`
Type	Pointer to a function of type void Function_Name (void)
File	`FlsLoader.h`
Description	Call out function to application which is called at every user defined time rate.
Source	IFX
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.2.3 FlsLoader_ConfigType

Table 51 Specification for FlsLoader_ConfigType

Syntax	`FlsLoader_ConfigType`	
Type	void	
File	`FlsLoader.h`	
Range	None	
Description	This defines the void configuration type as the module supports single configuration variant.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.4 FlsLoader_LengthType

Table 52 Specification for FlsLoader_LengthType

Syntax	`FlsLoader_LengthType`	
Type	uint32	
File	`FlsLoader.h`	
Range	0 to 4294967295	Length information for write and erase operations
Description	This type specifies length information for write and erase operations as following: Write: Number of bytes to be written. Erase: Number of logical sectors to be erased.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.5 FlsLoader_ReturnType

Table 53 Specification for FlsLoader_ReturnType

Syntax	`FlsLoader_ReturnType`	
Type	uint32	
File	`FlsLoader.h`	
Range	0 - FLSLOADER_E_OK	Successful execution
	1 - FLSLOADER_E_NOT_OK	Development error occurred
	2 - FLSLOADER_E_LOCKED	Sectors are read/write protected
	3 - FLSLOADER_E_ROMVERSION	All sectors are protected under OTP
	5 - FLSLOADER_E_BUSY	Device is busy
Description	This specifies the various Return types that can be specified by the APIs. This type is used for the errors detected by the APIs.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.6 FlsLoader_ValueType

Table 54 Specification for FlsLoader_ValueType

Syntax	`FlsLoader_ValueType`	
Type	uint32	
File	`FlsLoader.h`	
Range	0 to 4294967295	Password (8 words)
Description	The type specifies values for Flash (PFlash or DFlash) protection password (8 words).	

Table 54 Specification for FlsLoader_ValueType (continued)

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 FLSLOADER driver.

1.3.3.1 FlsLoader_DeInit

Table 55 Specification for `FlsLoader_DeInit` API

Syntax	`FlsLoader_ReturnType  FlsLoader_DeInit`
	`(`
	`  void`
	`)`	

Service ID	0x30	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	FlsLoader_ReturnType	FLSLOADER_E_OK: Successful execution FLSLOADER_E_BUSY: Flash is busy with erase/write operation FLSLOADER_E_NOT_OK: Development error occurred
Description	This function de-initializes the Flash module. This Function sets the registers to their default state and executes the reset to read command.	
Source	IFX	
Error handling	FLSLOADER_E_BUSY, FLSLOADER_E_UNINIT	
Configuration dependencies	FlsLoaderDeInitApi	
User hints	-	
SFR accessed	CPU_BIV(w), CPU_BTV(w), CPU_CORE_ID(r), CPU_DCON0(w), CPU_ISP(w), CPU_PCON0(w), CPU_PMA0(w), CPU_PMA1(w), CPU_SEGEN(w), DMU_HF_ECCC(rw), DMU_HF_STATUS(r), SCU_WDTCPU_CON0(rw), SCU_WDTCPU_SR(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 55 Specification for `FlsLoader_DeInit` API (continued)

Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.2 FlsLoader_Erase

Table 56 Specification for `FlsLoader_Erase` API

Syntax	`FlsLoader_ReturnType FlsLoader_Erase`
	`(`
	`    const FlsLoader_AddressType TargetAddress,`
	`    const FlsLoader_LengthType Length`
	`)`	

Service ID	0x32	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	TargetAddress Length	Target address in Flash memory. It should be aligned to the following sector sizes of the selected Flash for erase. PFlash: 16 Kbyte DFlash: 4 Kbyte Number of Flash (PFlash or DFlash) sectors to be erased. *Note: Number of sectors should lie within single Flash bank. Erase operation across the Flash banks is not supported.*
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	FlsLoader_ReturnType	FLSLOADER_E_OK: Successful completion FLSLOADER_E_BUSY: Flash is busy with erase/write operation FLSLOADER_E_NOT_OK: DET error, Sequence error, Erase verify error, Program verify error (for PFlash), Protection error or Operation error occurred FLSLOADER_E_LOCKED: Sector is protected (If FlsLoaderEnableLockCheck is enabled)
Description	This function erases the logical sectors of the internal Flash. The completion of this operation is denoted by clearing of busy status flag or error.	
Source	IFX	
Error handling	FLSLOADER_E_PARAM_ADDRESS, FLSLOADER_E_PARAM_LENGTH, FLSLOADER_E_UNINIT, FLSLOADER_E_BUSY	
Configuration dependencies	-	
User hints	-	

Table 56 Specification for `FlsLoader_Erase` API (continued)

SFR accessed	DMU_HF_ERRSR(r), DMU_HF_OPERATION(r), DMU_HF_PROCONDF(r), DMU_HF_PROTECT(r), DMU_HF_STATUS(r), DMU_HP_PROCON_OTP0(r), DMU_HP_PROCON_OTP1(r), DMU_HP_PROCON_OTP2(r), DMU_HP_PROCON_OTP3(r), DMU_HP_PROCON_OTP4(r), DMU_HP_PROCON_OTP5(r), DMU_HP_PROCON_P0(r), DMU_HP_PROCON_P1(r), DMU_HP_PROCON_P2(r), DMU_HP_PROCON_P3(r), DMU_HP_PROCON_P4(r), DMU_HP_PROCON_P5(r), DMU_HP_PROCON_WOP0(r), DMU_HP_PROCON_WOP1(r), DMU_HP_PROCON_WOP2(r), DMU_HP_PROCON_WOP3(r), DMU_HP_PROCON_WOP4(r), DMU_HP_PROCON_WOP5(r), SCU_WDTS_CON0(rw), 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.

1.3.3.3 FlsLoader_GetVersionInfo

Table 57 Specification for `FlsLoader_GetVersionInfo` API

Syntax	`void FlsLoader_GetVersionInfo ( Std_VersionInfoType * const VersionInfoPtr )`	
Service ID	0x35	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	VersionInfoPtr	Pointer to where the version information has to be stored
Parameters (in - out)	-	-
Return	void	-
Description	This functions provides the version information of the FLSLOADER driver.	
Source	IFX	
Error handling	FLSLOADER_E_PARAM_POINTER	
Configuration dependencies	FlsLoaderVersionInfoApi	
User hints	None	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.4 FlsLoader_Init

Table 58 Specification for `FlsLoader_Init` API

Syntax	`FlsLoader_ReturnType FlsLoader_Init ( const FlsLoader_ConfigType * const Address )`	
Service ID	0x2F	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	Address	NULL pointer because the driver supports single configuration variant
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	FlsLoader_ReturnType	FLSLOADER_E_OK: Successful execution FLSLOADER_E_ROMVERSION: All sectors are protected with OTP or WOP protection FLSLOADER_E_NOT_OK: Development error occurred
Description	This function initializes the Flash module and checks if all the Flash sectors are configured as ROM (OTP or WOP protected).	
Source	IFX	
Error handling	FLSLOADER_E_PARAM_IGNORED	
Configuration dependencies	-	
User hints	None	
SFR accessed	CPU_BIV(w), CPU_BTV(w), CPU_CORE_ID(r), CPU_DCON0(w), CPU_ISP(w), CPU_PCON0(w), CPU_PMA0(w), CPU_PMA1(w), CPU_SEGEN(w), DMU_HF_ECCC(rw), DMU_HP_PROCON_OTP0(r), DMU_HP_PROCON_OTP1(r), DMU_HP_PROCON_OTP2(r), DMU_HP_PROCON_OTP3(r), DMU_HP_PROCON_OTP4(r), DMU_HP_PROCON_OTP5(r), DMU_HP_PROCON_WOP0(r), DMU_HP_PROCON_WOP1(r), DMU_HP_PROCON_WOP2(r), DMU_HP_PROCON_WOP3(r), DMU_HP_PROCON_WOP4(r), DMU_HP_PROCON_WOP5(r), SCU_WDTCPU_CON0(rw), SCU_WDTCPU_SR(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.	

1.3.3.5 FlsLoader_Lock

Table 59 Specification for `FlsLoader_Lock` API

Syntax	`FlsLoader_ReturnType  FlsLoader_Lock`
	`(`
	`  void`
	`)`	

Service ID	0x33	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	FlsLoader_ReturnType	FLSLOADER_E_OK: Protections are installed successfully FLSLOADER_E_BUSY: Flash is busy with erase/write operation FLSLOADER_E_NOT_OK: DET error or failure (Sequence error, Protection error, Operation error, Program verify error or Erase verify error) reported while installing protections for at least one among DFlash0 read/write or PFlash write or PFlash OTP/WOP protections FLSLOADER_E_LOCKED: Protections for DFlash0 read/write, PFlash write and PFlash WOP/OTP are already installed
Description	This function locks (protect) the internal PFlash and DFlash0 of micro-controller with pre-configured protections. Following protection configurations are supported by the driver: -DFlash: Read protection, write protection. -DFlash protections are configurable at bank level. -PFlash: Write protection, write once protection (WOP), one time programmable (OTP) protection. -PFlash protections are configurable at sector level. However a bank and its corresponding sectors to be protected shall have same protection configured.	
Source	IFX	
Error handling	FLSLOADER_E_UNINIT, FLSLOADER_E_BUSY	
Configuration dependencies	FlsLoaderLockUnlockApi	
User hints	None	
SFR accessed	DMU_HF_CONFIRM1(r), DMU_HF_CONFIRM2(r), DMU_HF_ERRSR(r), DMU_HF_OPERATION(r), DMU_HF_PROTECT(r), DMU_HF_STATUS(r), STM_TIM0(r)	

Table 59 Specification for `FlsLoader_Lock` API (continued)

	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 FlsLoader_UnLock

Table 60 Specification for `FlsLoader_UnLock` API

Syntax	`FlsLoader_ReturnType  FlsLoader_UnLock`
	`(`
	`    const FlsLoader_AddressType TargetAddress,`
	`    const FlsLoader_ValueType * const Password`
	`)`	

Service ID	0x34	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	TargetAddress Password	UCB address of corresponding Flash to be unlocked. 0xAF402000 - PFlash UCB 0xAF402200 - DFlash UCB Pointer to the 4 double word (256 bit) UCB password of corresponding Flash to be unlocked.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	FlsLoader_ReturnType	FLSLOADER_E_OK: Successful completion FLSLOADER_E_BUSY: Flash is busy with erase/write operation FLSLOADER_E_NOT_OK: DET error, Operation error, Sequence error or Protection error occurred
Description	This function is used to unlock the internal PFlash and DFlash0 of the micro-controller from active protection. It temporarily (until next controller reset) disables the current active read or write protection. A wrong password results in protection error. - DFlash0 can be unlocked from read and write protections. - PFlash can be unlocked from write protection. WOP and OTP cannot be unlocked.	
Source	IFX	
Error handling	FLSLOADER_E_UNINIT, FLSLOADER_E_PARAM_ADDRESS, FLSLOADER_E_BUSY	
Configuration dependencies	FlsLoaderLockUnlockApi	

Table 60 Specification for `FlsLoader_UnLock` API (continued)

User hints	None
SFR accessed	DMU_HF_ERRSR(r), DMU_HF_PROTECT(r), DMU_HF_STATUS(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 FlsLoader_Write

Table 61 Specification for `FlsLoader_Write` API

Syntax	`FlsLoader_ReturnType  FlsLoader_Write`
	`(`
	`    const FlsLoader_AddressType TargetAddress,`
	`    const FlsLoader_LengthType Length,`
	`    const uint8 * const SourceAddressPtr`
	`)`	

Service ID	0x31	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	TargetAddress Length SourceAddressPtr	Target address in Flash memory. It should be aligned to the following page sizes of the selected Flash for write. PFlash: 32 bytes DFlash: 8 bytes Number of bytes to be written. It should be multiple of the following page sizes of the selected Flash for write. PFlash: 32 bytes DFlash: 8 bytes Pointer to source data buffer
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	FlsLoader_ReturnType	FLSLOADER_E_OK: Successful execution FLSLOADER_E_BUSY: Flash is busy with erase/write operation FLSLOADER_E_NOT_OK: DET error, Sequence error, Program verify error, Protection error or Operation error occurred FLSLOADER_E_LOCKED: Sector is protected (If FlsLoaderEnableLockCheck is enabled)

Table 61 **Specification for `FlsLoader_Write` API (continued)**

Description	This function is used to program a page of internal Flash. Sectors of PFlash and DFlash can be programmed.
Source	IFX
Error handling	FLSLOADER_E_PARAM_ADDRESS, FLSLOADER_E_PARAM_LENGTH, FLSLOADER_E_UNINIT, FLSLOADER_E_BUSY
Configuration dependencies	-
User hints	None
SFR accessed	DMU_HF_ERRSR(r), DMU_HF_OPERATION(r), DMU_HF_PROCONDF(r), DMU_HF_PROTECT(r), DMU_HF_STATUS(r), DMU_HP_PROCON_OTP0(r), DMU_HP_PROCON_OTP1(r), DMU_HP_PROCON_OTP2(r), DMU_HP_PROCON_OTP3(r), DMU_HP_PROCON_OTP4(r), DMU_HP_PROCON_OTP5(r), DMU_HP_PROCON_P0(r), DMU_HP_PROCON_P1(r), DMU_HP_PROCON_P2(r), DMU_HP_PROCON_P3(r), DMU_HP_PROCON_P4(r), DMU_HP_PROCON_P5(r), DMU_HP_PROCON_WOP0(r), DMU_HP_PROCON_WOP1(r), DMU_HP_PROCON_WOP2(r), DMU_HP_PROCON_WOP3(r), DMU_HP_PROCON_WOP4(r), DMU_HP_PROCON_WOP5(r), SCU_WDTS_CON0(rw), 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 FLSLOADER driver by itself does not implement any notifications. However, if the call out feature is enabled by user, it provides a call out to application while looping for status during write and erase operations.

1.3.5 Scheduled functions

The FLSLOADER driver does not provide any scheduled functions.

1.3.6 Interrupt service routines

The FLSLOADER driver does not provide any interrupt handlers.

1.3.7 Callout

The driver does not support any callout functions.

1.3.8 Errors Handling

This section describes the various errors reported by FLSLOADER driver.

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
FLSLOADER_E_BUSY: API service called while driver still busy.	IFX	0x5	DET	0x5	DET
FLSLOADER_E_PARAM_ADDRESS: API service called with wrong address.	IFX	0x3	DET	0x3	DET
FLSLOADER_E_PARAM_IGNORED: API service called with not a NULL pointer.	IFX	0x0	DET	0x0	DET
FLSLOADER_E_PARAM_LENGTH: API service called with wrong length.	IFX	0x2	DET	0x2	DET
FLSLOADER_E_PARAM_POINTER: API service called with NULL pointer.	IFX	0x6	DET	0x6	DET
FLSLOADER_E_UNINIT: APIs are invoked without initialization of the driver.	IFX	0x4	DET	0x4	DET

1.3.9 Deviations and limitations

The section describes the deviations and limitations of the FLSLOADER driver.

1.3.9.1 Deviations

This section describes the deviation of the FLSLOADER driver.

1.3.9.1.1 Software specification deviations

Currently there are no deviations for the FLSLOADER driver.

1.3.9.1.2 AMDC Violations

This FLSLOADER driver does not have any AMDC violations.

1.3.9.1.3 VSMD Violations

The FLSLOADER is complex driver, VSMD violation is not applicable.

1.3.9.2 Limitations

The section describes the limitations from software specification.

Table 62 Known limitations

Reference	Limitation
PFlash erase	The maximum PFlash sectors that can be erased by erase API is 192. If length is more than 192 sectors, the application should call erase API multiple times as per the erase length.
Write to WOP protected sector	The driver does not support write to the PFlash sectors which are protected with WOP.
Timeout values for write and erase operations	All timeout values used by the FLSLOADER are calculated assuming the FSI operates at 100MHz.
Increased timeout durations	When the DFlash0/PFlash is accessed by the FlsLoader on the TriCore side and DFlash1/PFlash is accessed simultaneously on the HSM side, 5ms additional time is taken for write operations and the duration of erase operations increases by about 15%. Timeout calculations are performed assuming the DFlash0/PFlash and DFlash1/PFlash are accessed simultaneously from the TriCore side and the HSM side. Therefore, if there is no simultaneous access, then the timeouts will be delayed.

Revision history

Table 63 Revision History

Date	Version	Description
2021-03-09	4.0	Document is released.
2021-03-05	3.1	Tag formatting corrected in section 1.1.5.1, No functional change.
2020-12-10	3.0	Document is released.
2020-12-10	2.1	Parameter names corrected in UM document to FlsLoaderPFUcbPW0_0, FlsLoaderPFUcbPW0_1, FlsLoaderPFUcbPW1_0, FlsLoaderPFUcbPW1_1, FlsLoaderPFUcbPW2_0, FlsLoaderPFUcbPW2_1, FlsLoaderPFUcbPW3_0, FlsLoaderPFUcbPW3_1, FlsLoaderPFLash0WriteProt, FlsLoaderPFLash1WriteProt, FlsLoaderPFLash2WriteProt, FlsLoaderPFLash3WriteProt, FlsLoaderPFLash4WriteProt, FlsLoaderPFLash5WriteProt. No change in implementation.
2020-12-07	2.0	Document is released.
2020-11-26	1.1	No functional change, updated to align with template.
2020-08-14	1.0	Document is released.
2020-07-29	0.1	- Initial version FLSLOADER chapter moved from MC-ISAR_TC3xx_UM_CD to this document. - Added file FlsLdr_ExclArea.h for FLSLOADER exclusive area, updated the file structure diagram. - Added the example usage for FLSLOADER exclusive area during PFLSH write and erase operations. - added limitation for increased timeout values during write and erase operation considering parallel access to DFlash0/PFlash and DFlash1/PFlash by TriCore and HSM respectively.

Edition 2021-03-09

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