MCAL User Manual for Fee

32-bit TriCore™ AURIX™ TC3xx microcontroller

About this document

Scope and purpose

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

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

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

Intended audience

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

Document conventions

Table 1 Conventions

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

Reference documents

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

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

Table of contents

Table of contents

Table of contents

Table of contents

1 Fee driver

1.1 User information

1.1.1 Description

The FEE driver provides Flash EEPROM emulation as per AUTOSAR through standard services and well-defined configuration. Additionally, customer specific features such as virgin Flash handling, quasi-static (QS) data block support, un-configured data block support, erase-suspend resume are also made available. In view of the second generation of AURIX™ hardware, the DFlash0 EEPROM memory region is exclusively used by the FEE driver to provide emulated EEPROM functionality. The DFlash-specific operations such as erase, read and write are implemented in the FLS driver. The DFlash1 is not used by the FEE. DFlash1 is reserved for HSM. The FEE driver is delivered as a Post-Build variant as FEE functionality is not only available for the run time application but also for the boot code. *Note 1: The quasi-static data area has a limit of 500 erase/write cycles. Note 2: FEE module cannot result in an endless loop leading to a watchdog timeout.*

1.1.2 Hardware-software mapping

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

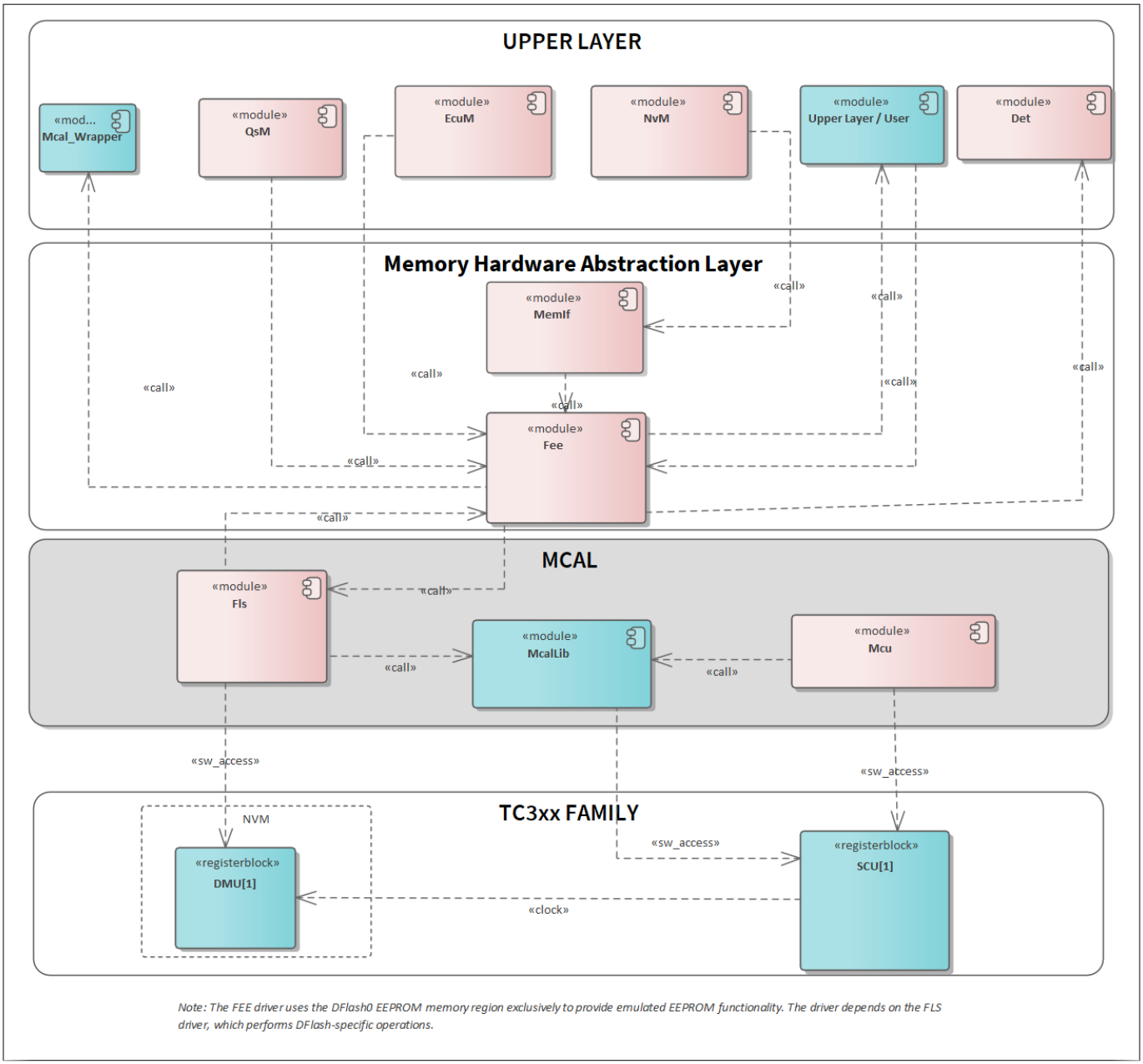


Figure 1 Mapping of hardware-software interfaces

1.1.3 File structure

1.1.3.1 C file structure

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

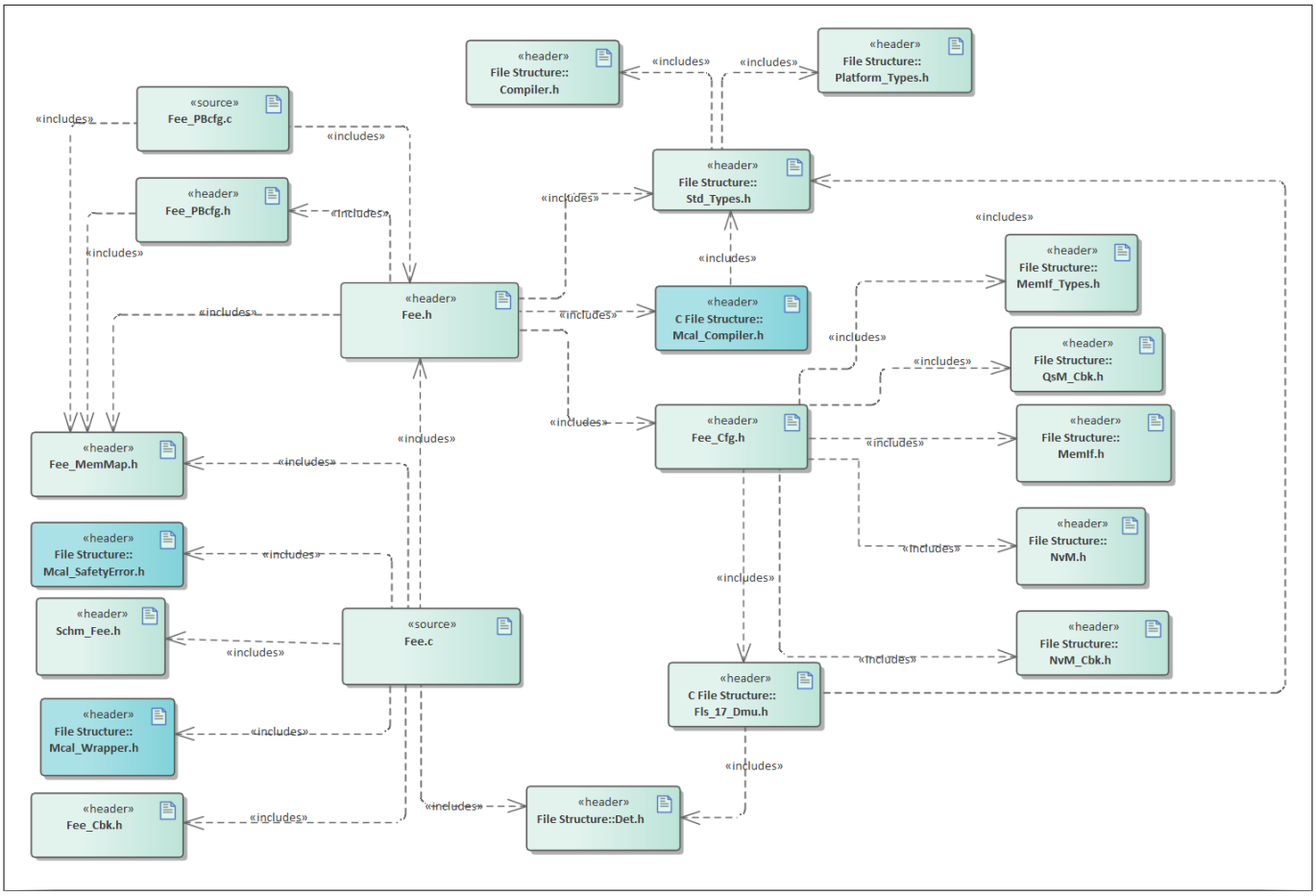


Figure 2 Fee_C_File_Structure-1.png

Table 2 C file structure

File name	Description
`Compiler.h`	Provides abstraction from compiler-specific keywords
`Det.h`	Provides the exported interfaces of Development Error Tracer
`Fee.c`	Contains the functionality of the FEE driver
`Fee.h`	This header file exports the functionality of the FEE driver
`Fee_Cbk.h`	This header file contains the declarations of callback functions provided by the FEE driver
`Fee_Cfg.h`	Contains pre-compile configuration data of the FEE driver
`Fee_MemMap.h`	File containing the memory section definitions used by the FEE driver
`Fee_PBcfg.c`	Contains configuration data of the FEE driver
`Fee_PBcfg.h`	File (Generated) containing declaration of the post-build configuration data structures
`Fls_17_Dmu.h`	This header file exports macros, type definitions, interrupt service routine and function prototypes for the Flash driver
`Mcal_Compiler.h`	Header file providing abstraction for TriCore™-intrinsic instruction.

(table continues...)

Table 2 (continued) C file structure

File name	Description
`Mcal_SafetyError.h`	Header file containing the prototype of the API for reporting safety-related errors
`Mcal_Wrapper.h`	Provides the exported interfaces for Production Error and Runtime Development Errors. Implemented by default to include functions of Dem.h and Det.h files. This file can be modified by the user but function prototype is not user modifiable.
`MemIf.h`	Header file containing exported interfaces and type definitions of MemIf module. *Note: This file is available only for AUTOSAR version 4.4.0.*
`MemIf_Types.h`	Header file containing the type declaration of MemIf. *Note: This file is available only for AUTOSAR version 4.2.2.*
`NvM.h`	Header file containing call back definitions of Nvm module. *Note: This file is available only for AUTOSAR version 4.4.0*
`NvM_Cbk.h`	Call back header file for NvM. *Note: This file is available only for AUTOSAR version 4.2.2*
`Platform_Types.h`	Platform-specific type declaration file as defined by AUTOSAR
`QsM_Cbk.h`	Interface file that provides the callback function prototypes to be used by FEE driver.
`Schm_Fee.h`	Header file containing prototype of the scheduled function of the Fee driver.
`Std_Types.h`	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

1.1.3.2 Code generator plugin files

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

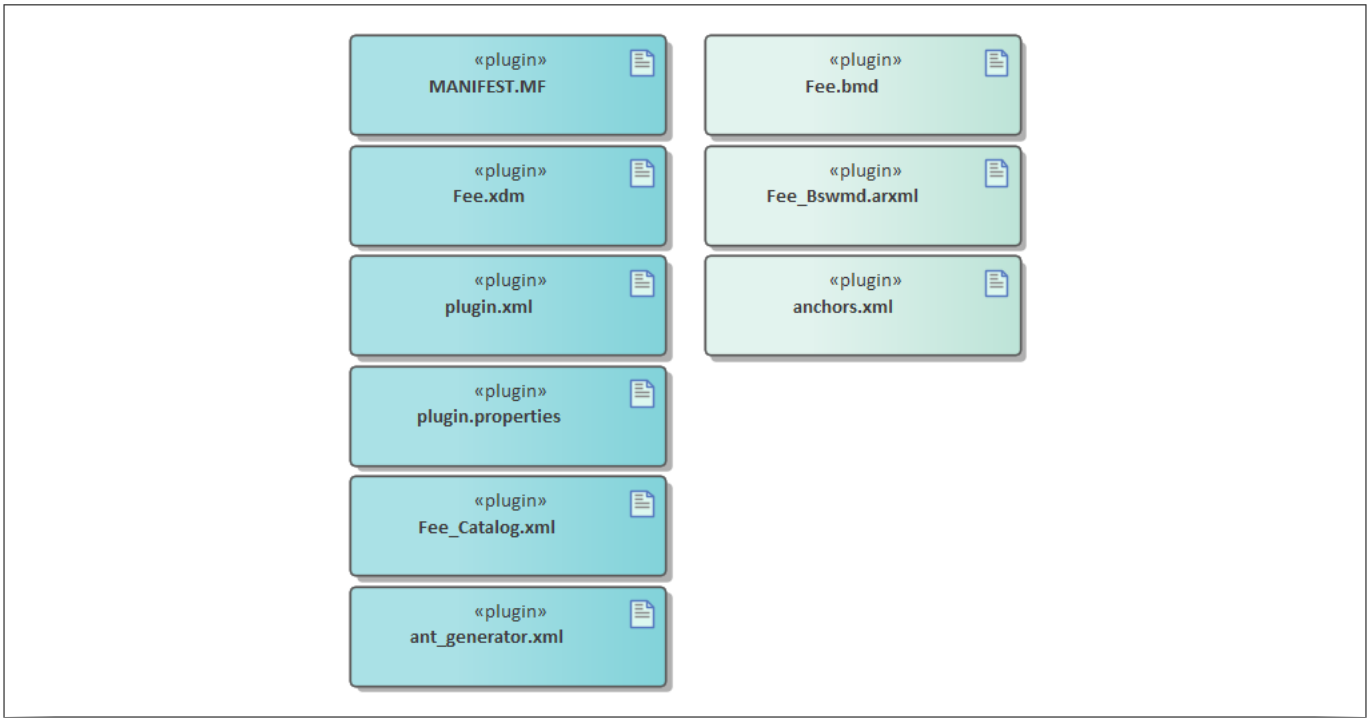


Figure 3 Fee_Code_Generator_Plugin_Files-1.png

Table 3 Code generator plugin files

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

1.1.4 Integration hints

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

1.1.4.1 Integration with AUTOSAR stack

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

- **EcuM:**

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

- **FLS:**

  The FEE driver depends on the FLS driver for operation. Therefore, the Infineon FLS driver is required to be configured to operate with the Infineon FEE driver.

- **Memory mapping:**

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

  The file `Fee_MemMap.h` 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 depicted below.

```
#if defined FEE_START_SEC_VAR_INIT_ASIL_B_GLOBAL_UNSPECIFIED
 /* User pragmas here */
 #undef FEE_START_SEC_VAR_INIT_ASIL_B_GLOBAL_UNSPECIFIED
 #undef MEMMAP_ERROR

#elif defined FEE_STOP_SEC_VAR_INIT_ASIL_B_GLOBAL_UNSPECIFIED
 /* User pragmas here */
 #undef FEE_STOP_SEC_VAR_INIT_ASIL_B_GLOBAL_UNSPECIFIED
 #undef MEMMAP_ERROR

#elif defined FEE_START_SEC_CODE_ASIL_B_GLOBAL
 /* User pragmas here */
 #undef FEE_START_SEC_CODE_ASIL_B_GLOBAL
 #undef MEMMAP_ERROR

#elif defined FEE_STOP_SEC_CODE_ASIL_B_GLOBAL
 /* User pragmas here */
 #undef FEE_STOP_SEC_CODE_ASIL_B_GLOBAL
 #undef MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Fee_MemMap.h, wrong pragma command"
#endif
```

- **DET:**

  The DET module is a part of the AUTOSAR stack that handles all the development error reported by the Basic Software modules. The FEE driver reports all the development errors to the DET module through the `Det_ReportError()` API for AUTOSAR release version 4.4.0. The user of the FEE driver must process all the errors reported to the DET module through the `Det_ReportError()` API for AUTOSAR release version 4.4.0.

  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.

- **Mcal_Wrapper:**

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

  The user of the FEE driver shall process all the production errors (fail/pass) and Runtime errors reported to the Mcal_Wrapper module. The interface used for reporting production error In AUTOSAR version 4.2.2 is `Mcal_Wrapper_Dem_ReportErrorStatus()` and for AUTOSAR version 4.4.0 is `Mcal_Wrapper_Dem_SetEventStatus()`, for reporting Runtime error `Mcal_Wrapper_Det_ReportRuntimeError()` API is used. The `Mcal_Wrapper.c` and `Mcal_Wrapper.h` files are provided in the MCAL package as a stub code and can be replaced with a user specific production and Runtime error handling module/s during the integration phase.

- **SchM:**

  SchM is not required for the integration of FEE driver.

- **Safety error:**

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

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

  Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and call-backs:**

  The FEE driver implements callback functions invoked from the FLS driver. These functions are to be configured at the time of configuring the FLS driver.

  The FEE driver reports the completion of jobs and errors through configurable notification functions. These functions can be configured by the user while configuring the FEE module in Tresos.

- **Operating system:**

  The FEE driver does not use any operating system service.

1.1.4.2 Multicore and Resource Manager

FEE driver does not support execution on multiple cores in parallel.

1.1.4.3 MCU support

FEE driver does not use any services provided by the MCU driver.

1.1.4.4 Port support

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

1.1.4.5 DMA support

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

1.1.4.6 Interrupt connections

The FEE driver does not use any interrupt source.

1.1.4.7 Example usage

Configuration of the driver

The FEE driver could be configured in the following three modes:

FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA, FEE_DOUBLE_SECTOR_DATA_ONLY or FEE_QUASI_STATIC_DATA_ONLY.

For configuration, the user needs to set FeeBlockTypeConfigured appropriately. FeeBlockTypeConfigured could be found under general tab in the EB tresos configuration for FEE. For example, if the user intends to use both double-sector as well as Quasi-Static (QS) data, then FeeBlockTypeConfigured should be set as FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA.

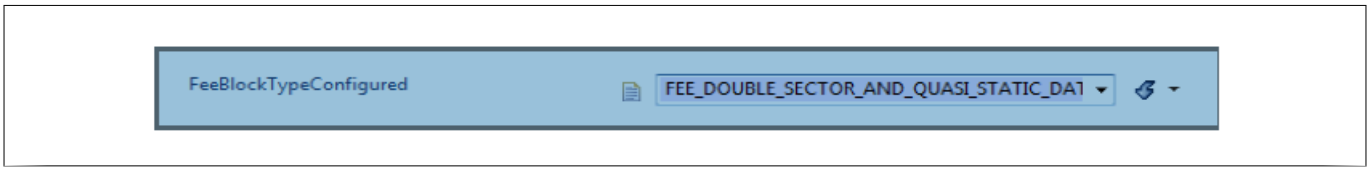


Figure 4 Configuration of FeeBlockTypeConfigured

Note:

- *For only normal data (double sector), FeeBlockTypeConfigured should be set as FEE_DOUBLE_SECTOR_DATA_ONLY.*
- *For only QS configuration FeeBlockTypeConfigured should be set as FEE_QUASI_STATIC_DATA_ONLY.*

Depending upon the configuration chosen by the user, the corresponding data blocks should be configured in FeeBlockConfiguration. For example, if FeeBlockTypeConfigured is set as FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA, then in the FeeBlockConfiguration section both the normal double sector and QS blocks need to be configured.

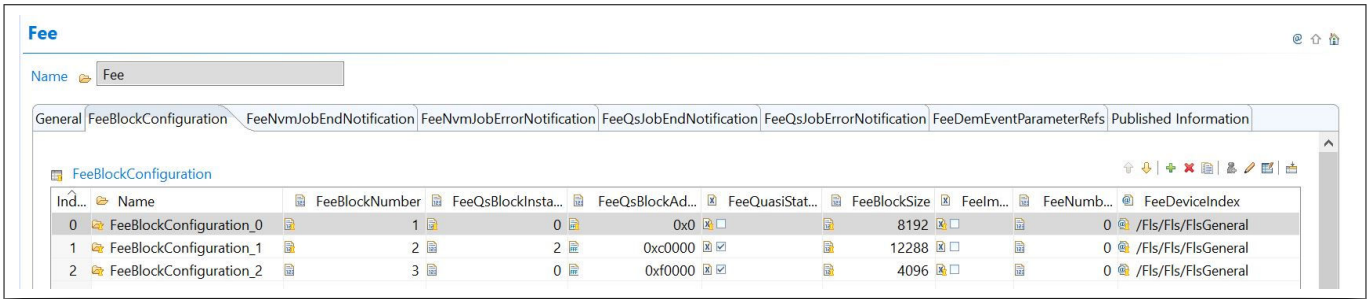


Figure 5 FeeBlockConfiguration

Note that FeeQsBlockAddress is applicable only for QS blocks. In the above example, it is 0xc0000 because this address is configured in FLS. This address should not overlap with the normal double-sector size addresses. Further, while configuring the QS block address (FeeQsBlockAddress) and QS block sizes (FeeBlockSize), the user must take care to fill the values for each QS data block properly in a way that they do not breach the QS sector size configured in FLS. The QS addresses should not overlap each other as well and should be contiguous.

- If FeeBlockTypeConfigured is configured as FEE_DOUBLE_SECTOR_DATA_ONLY then only the normal double-sector data blocks should be configured.
- If FeeBlockTypeConfigured is configured as FEE_QUASI_STATIC_DATA_ONLY is selected then only the QS data blocks should be configured.

The configuration, corresponding to that done in FEE, must be done in FLS as well. This could be done as follows:

1. Go to FLS configuration and configure the sectors for normal double sector and QS.
2. Mention the sector size for each.

3. Give the start address for both the sectors appropriately. Note that the QS sector address should be after the sectors of the normal double-sector address.

For example, in FEE, each sector of the normal double sector is of 0x60000 or 393216 bytes (384 KB) in size, giving a total of 768 KB or 786432 bytes in size for both the sectors (double sector), therefore the value of QS address is 0xC0000 (786432), which is after the addresses used for the double sector. The total size of the QS sector is therefore, 0x40000, which is the remaining size after the sector size for both the sectors of the normal double sector is allocated.

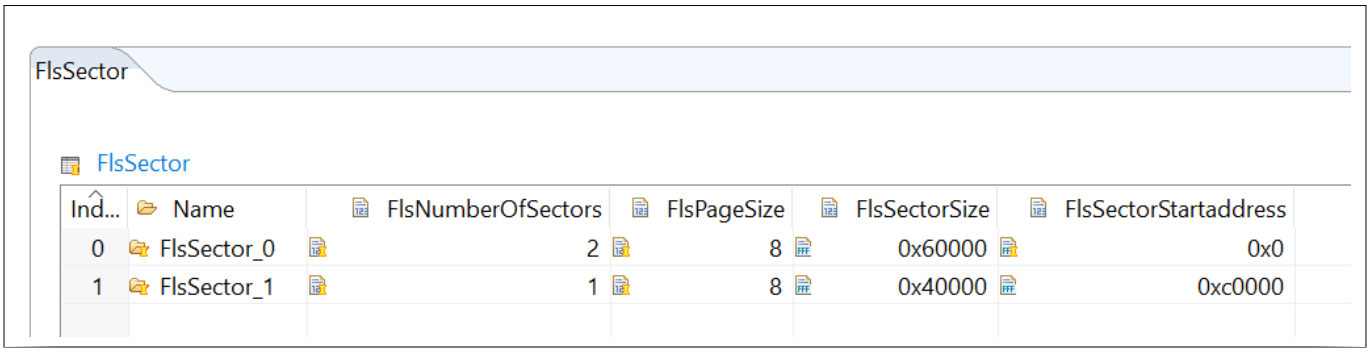
The calculation for the QS sector start address could be as follows:

- QS Sector start address = Sector size of one sector for double sector algorithm X 2
- As per the above-mentioned example:
  - Sector size of one sector for double sector algorithm = 0x60000
  - QS sector start address = 2 x 0x60000 = 0xC0000

The calculation for the QS sector size, in case the normal double sectors are also being used, could be summarized as follows:

- QS sector size = Total Flash size - (Sector size of one sector for double sector algorithm X 2)
- As per Figure:
  - Total Flash size = 0x100000
  - Sector size of one sector for double-sector algorithm = 0x60000
  - QS sector = 0x100000 – (2 X 0x60000) = 0x40000

Note that it is responsibility of the user to configure the sectors judiciously in the manner as described above.



Ind...	Name	FlsNumberOfSectors	FlsPageSize	FlsSectorSize	FlsSectorStartaddress
0	FlsSector_0	2	8	0x60000	0x0
1	FlsSector_1	1	8	0x40000	0xc0000

Figure 6 FLS sectors configuration

Note:

- *FlsSectorStartaddress for QS should not overlap with NVM blocks.*
- *If FeeBlockTypeConfigured is configured as FEE_DOUBLE_SECTOR_DATA_ONLY then only one FlsSector with a value of 2 for FlsNumberOfSectors should be configured.*
- *If FeeBlockTypeConfigured is configured as FEE_QUASI_STATIC_DATA_ONLY then only one FlsSector with a value of 1 for FlsNumberOfSectors. So, in the above case, FlsSector_0 with a value of 1 should be configured.*
- *If FeeBlockTypeConfigured is configured as FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA then the order of sectors in FLS configuration should the FlsSector_0 should be configured with 2 sectors and then FlsSector_1 should be configured with a value of 1 for the number of sectors.*

Initialization of FEE driver

As part of application initialization task, initialize the FLS and FEE drivers by calling the following APIs.

```
#include "Fee.h"
extern const Fls_17_Dmu_ConfigType Fls_17_Dmu_Config;
extern const Fee_ConfigType Fee_Config;
Fls_17_Dmu_Init(&Fls_17_Dmu_Config); /* taken from Fls_17_Dmu_PBcfg.c */
Fee_Init(&Fee_Config); /* taken from Fee_PBcfg.c */
```

This completes the initialization sequence.

FEE operation

For runtime FEE services, Fee_MainFunction is the scheduling function provided by the FEE driver. This function along with the scheduling function of FLS - Fls_17_Dmu_MainFunction () should be called periodically so that it can process the jobs. This API is a service for performing the processing of the Fee_Read (), Fee_Write (), etc. So, the main periodic task of the application should include the following.

Fls_17_Dmu_MainFunction ();

Fee_MainFunction ();

After performing any FEE operation like Fee_Read (), Fee_Write (), the following main functions should be called as follows:

```
while (Fee_GetStatus() != MEMIF_IDLE)
{
Fls_17_Dmu_MainFunction ();
Fee_MainFunction ();
}
```

Configuration of QS blocks

When user chooses to configure Quasi-Static data the following points should be considered:

- FeeQsBlockInstances of the FEE block should be 0 if it is declared as one of the multiple QS instances of an another FEE block.
- FEE block with multiple QS instances should have same FeeBlockSize for all the block instances.
- FeeQsBlockAddress should not overlap with the previously configured FEE QS block instance.
- FeeQsBlockAddress should be contiguous for all the QS block instances configured as a part of FEE block having multiple QS instances.
- FeeBlockNumber should be contiguous for all the block instances(for the QS block having multiple instances).
- Number of QS block instances configured as a part of FEE block having multiple QS instances, should be same as FeeQsBlockInstances set for the FEE block configured.

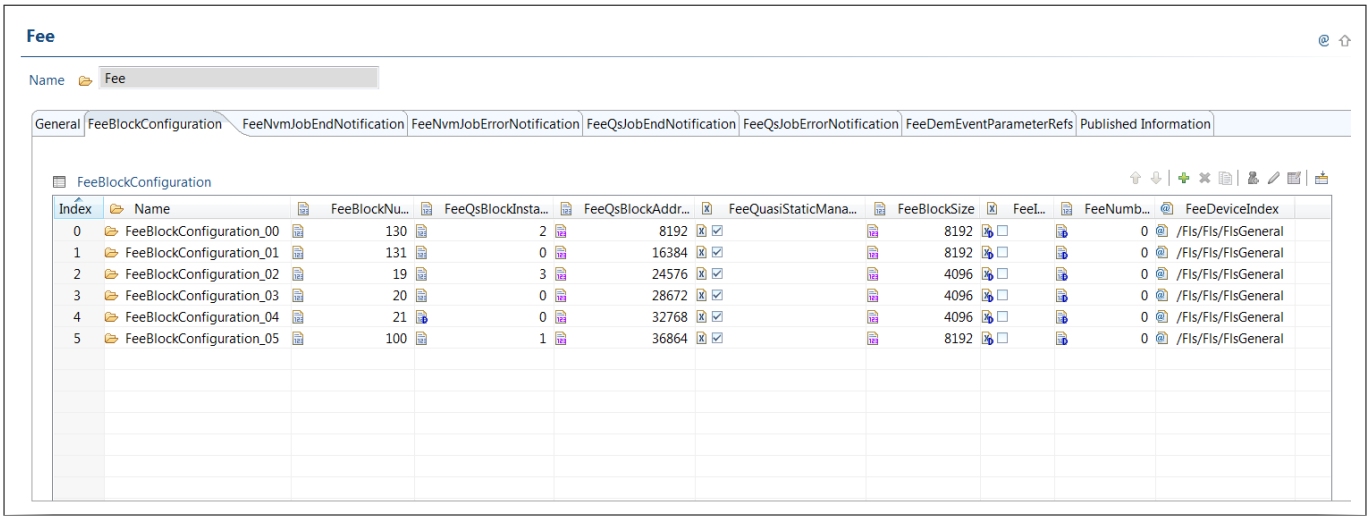


Figure 7 Sample QS block configuration

Key points to consider

FEE and FLS dependency

The user has to take care to link the Infineon FEE with the FLS module that is configured (FLS configuration parameter: FlsIfxFeeUse) specifically to support Infineon implementation of the FEE. This is required because the FLS module implements additional non-Autosar APIs for use by Infineon FEE and these are available only when FLS is configured to support the Infineon FEE.

Writing blocks close to GC threshold

When writing a new data block, the size of the previous data blocks present in a WL is added to the size of the new incoming block and the result is compared with the threshold. If the threshold is likely to be breached, then the new incoming block will be attempted to be written on the next consecutive word-line. If it is determined that even in this scenario the threshold will be breached then the GC is triggered. The consequence of this decision is that a few pages of the Flash memory close to the threshold is unutilized and the GC may seem to be triggered earlier than expected. The occurrence of this behaviour is dependent on the size of the blocks already present in the Flash close to the threshold and the size of the block that is requested to be written.

FEE_E_GC_TRIG Production Error

During GC, if the total size of the blocks to be copied is greater than the available space in the sector (threshold is breached), then FEE_E_GC_TRIG production error will be triggered and an illegal state notification will be raised. The user must make sure that the block sizes and threshold are configured judiciously.

Data pointer for Fee_Read and Fee_Write API

Data pointer passed in read and write API need to be memory aligned(word aligned).

Quasi Static data blocks

Quasi Static data blocks are big data blocks (multiple of 4K) that are infrequently updated over the life time of ECU. The NVM cannot be easily adapted to handle quasi static data. So, all standard NVM blocks is handled via NVRAM Manager. However, quasi static data is to be handled by quasi static manager.

Quasi Static manager is implemented by the user to manage the quasi-static data. Quasi Static data blocks are read and written using FEE's read and write APIs. There are other APIs provided by FEE, for example, Fee_17_EraseQuasiStaticData, that are meant exclusively for QS data. Please refer the API chapter for further information on APIs applicable for Quasi Static data.

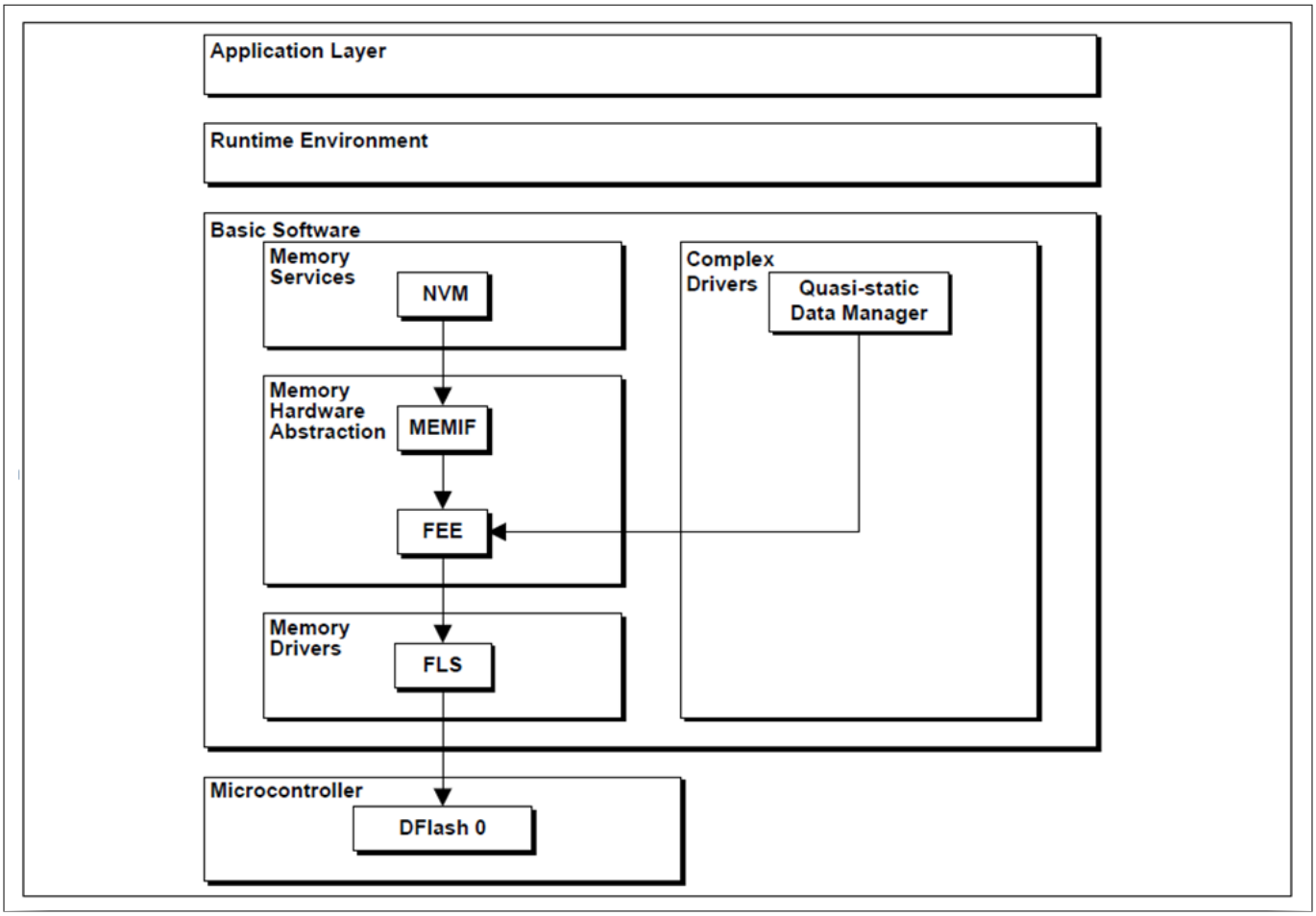


Figure 8 Handling DFlash0 by NVM and Quasi-static Manager

Quasi Static Block configuration: For Quasi Static block configuration please refers to section configuration interface.

DFlash Configuration: User can configure DFlash in one of three different configurations using configuration parameter “FeeBlockTypeConfigured”.

FeeBlockTypeConfigured = FEE_DOUBLE_SECTOR_DATA_ONLY

Only NVM data is present

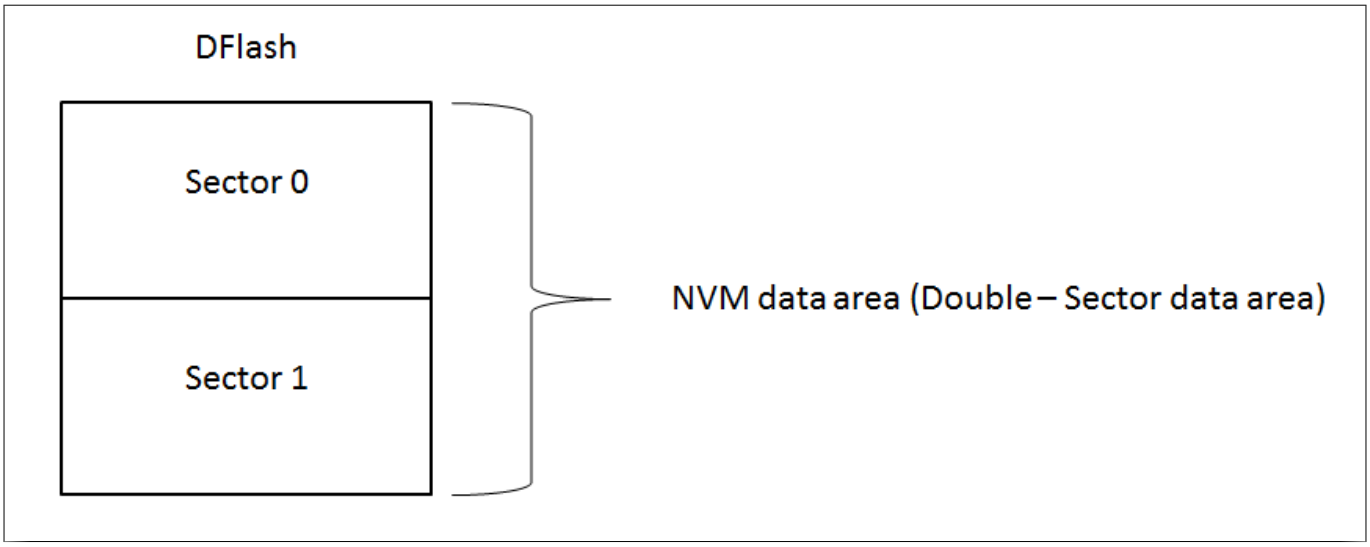


Figure 9 Only NVM data

FeeBlockTypeConfigured = FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA

NVM and Quasi static data is present

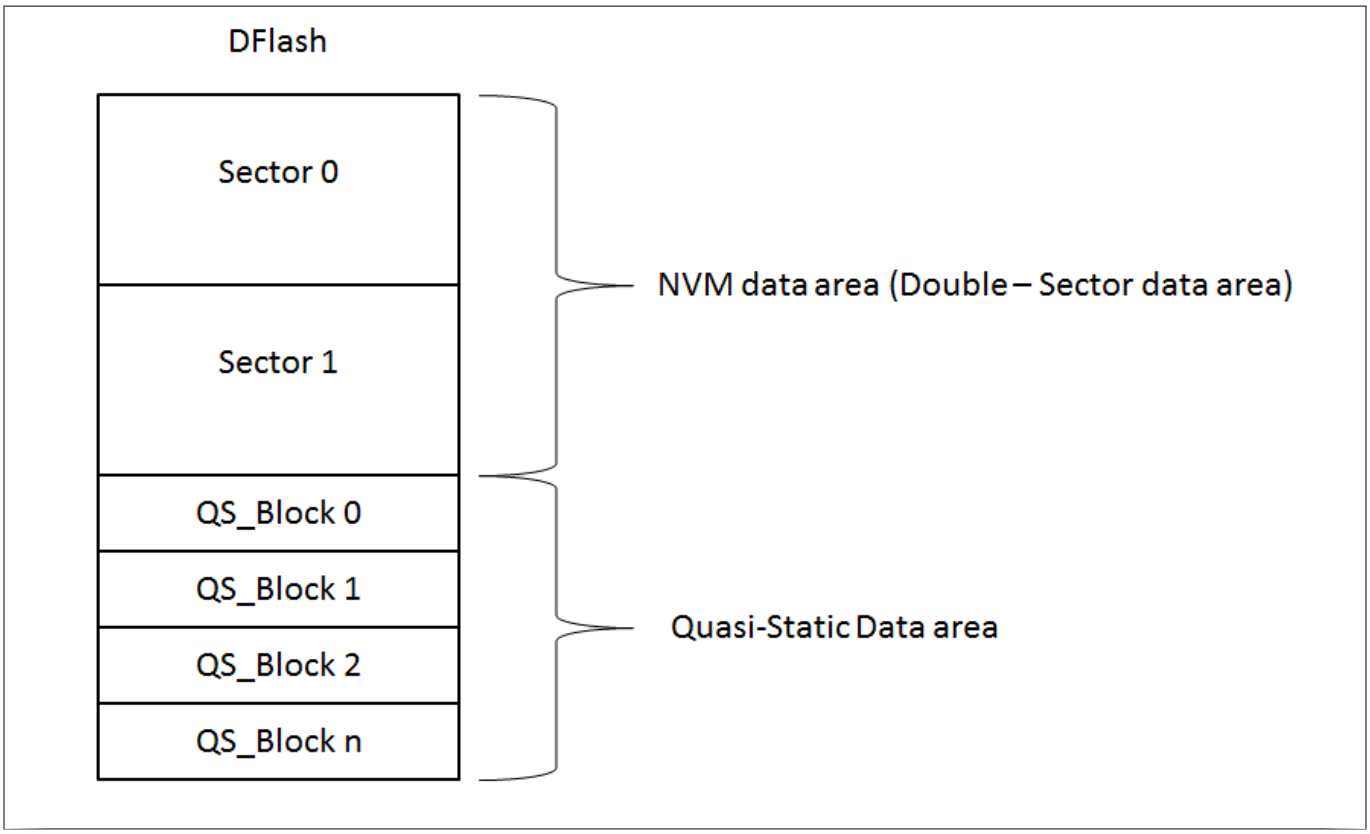


Figure 10 NVM and Quasi static data

FeeBlockTypeConfigured = FEE_QUASI_STATIC_DATA_ONLY

Only Quasi static data is present

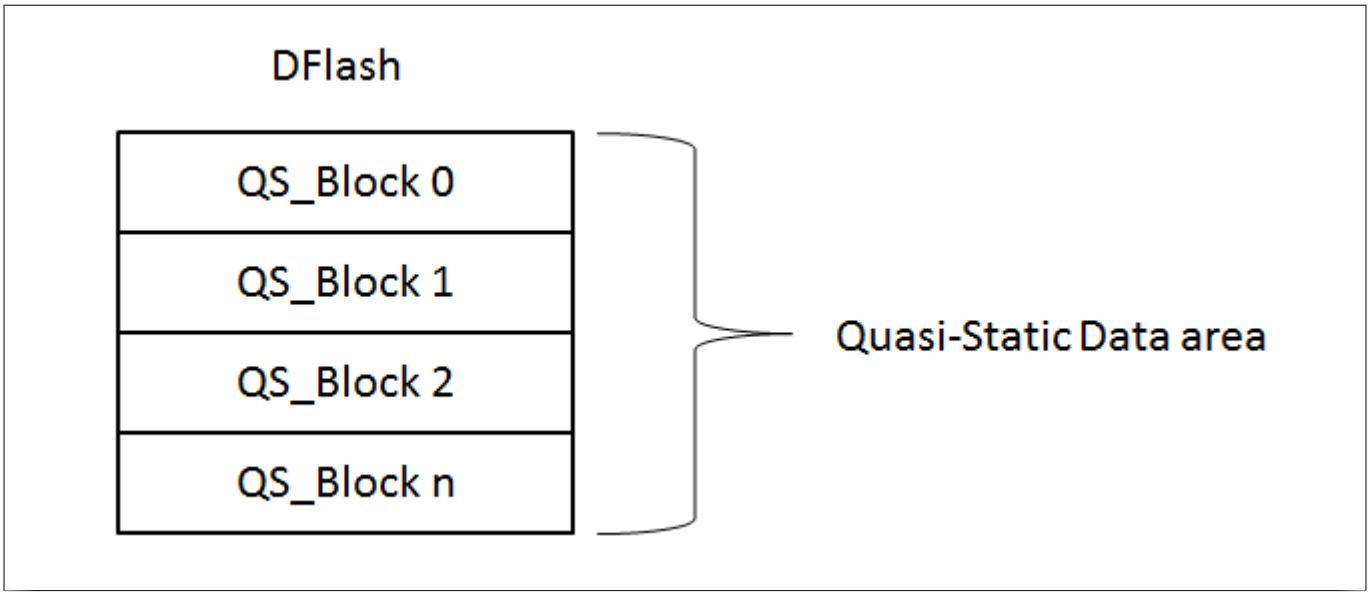


Figure 11 Quasi static data

Evaluation of disturbs in Quasi-Static area in DFLASH

For details please refer the TC3xx_SW_MCAL_FEE_Cycle_Calculator.xlsx provided in release package.

Fee_SetMode behavior for AUTOSAR version 4.4.0

- The `Fee_SetMode()` API for AUTOSAR 440 behaves as synchronous API when the API is called during MEMIF_IDLE driver state. Since underlying FLS driver implements the set mode change in synchronous manner and the mode switch is not time consuming operation.
- The `Fee_SetMode()` API behaves as asynchronous API when the driver is in MEMIF_BUSY_INTERNAL state. In this case the API call registers the request and execute the mode switch once the internal job is completed.
- During asynchronous behavior, if another `Fee_SetMode()` request is made before the execution of the previous request, then the second request will be executed. The registered information will be overwritten for the first call.

  Please refer example below:-

```
MemIf_StatusType DriverStatus;
DriverStatus = Fee_GetStatus(); /* DriverStatus = MEMIF_IDLE */

/* Fee_SetMode example for synchronous call when driver state is MEMIF_IDLE */
Fee_setmode(MEMIF_MODE_FAST); /* Mode switch operation completed */
…
DriverStatus = Fee_GetStatus(); /* DriverStatus = MEMIF_BUSY_INTERNAL */

/* Fee_SetMode example for asynchronous behavior when driver state is MEMIF_BUSY_INTERNAL */
Fee_setmode(MEMIF_MODE_SLOW); /* The API call registers the request but not completed the
operation */

/* Second call */
Fee_setmode(MEMIF_MODE_FAST); /* If set mode second call is before execution of first call then
current request will be over written */
```

Number of DFlash pages processed during initialization (cache build) in one Fee_MainFunction() cycle

Fee do the scanning of DFlash during initialization to identify the data block status, it is a time consuming process and causes a high peak execution time of the Fee_MainFunction(). To keep the peak execution time within tolerable limits, the scanning is performed over several cycles of the Fee_MainFunction(), with each cycle scanning only a limited number of blocks. The number of blocks scanned in one Fee_MainFunction() cycle can be configured through FeeBlocksScannedPerCycle configuration parameter.

Concurrent access to DFlash 0

FEE and FLS driver are designed assuming exclusive access to the DFlash 0. If DFLASH0 is shared by FEE/FLS and user implemented application then user need to take care the handshaking between FEE and user application to avoid concurrent accesses to DFLASH 0.

1.1.5 Key architectural considerations

The key architectural considerations are as follows:

- The FEE driver is platform independent and does not implement any interrupt service routines.
- For the double-sector algorithm, the area of the DFlash0 is divided into two sectors.
- At a given time, the double-sector algorithm will fill one of the sectors with data blocks and data will be updated by simply appending/writing the new data block to the free space, available at the end of the previous written block. When the sector being used becomes filled to a certain threshold, the process of garbage collection is triggered. The garbage collection process copies the most recent copies of the data blocks to the other sector and then erases the previously filled sector. This will be repeated when the second sector gets filled to a threshold.
- As data is frequently updated, word-line/bit-line shorts may occur in the double-sector DFlash range during erase and write.
- After any erase operation in the double-sector algorithm, word-line failures will be detected and handled. The design is scalable to handle a maximum of two word-line failures.
- Quasi-Static data handling: The standard EEPROM emulation algorithm is not efficient for relatively big data blocks with a low update rate (500 erase/write cycles over the entire QS data area). Therefore, in addition to the standard EEPROM emulation algorithm, a special algorithm for infrequently updated data blocks (Quasi-Static data) in a special DFlash area has been provided.

1.1.5.1 Double-sector algorithm is used

The algorithm used for EEPROM emulation with DFlash is Double-Sector algorithm.

1.1.5.2 Quasi-static data algorithm is used

In addition to the standard double-sector EEPROM emulation algorithm, a special algorithm for infrequently updated data blocks (called Quasi-Static data) is made available.

1.1.5.3 Post build variant

In FEE module the configuration parameters namely FeeQsBlockInstances, FeeQsBlockAddress, FeeBlockSize, FeeImmediateData and FeeNumberOfWriteCycles are implemented as post build parameters.

1.1.5.4 Callback notification when erase verify error occurs

The Fee_17_JobEraseErrorNotification function shall perform error handling operations and subsequently call the user configured QS job error notification routine of the upper layer module if configured.

1.1.5.5 Notifications to upper layer

Job end, job error, illegal state notifications are given through different interfaces for NVM and QS separately.

1.1.5.6 Handling ECC errors during GC

Configured and un-configured blocks containing ECC errors will be dropped during garbage collection. No notification is provided when such instances occur.

Note: When an ECC error occurs, not much can be done to retrieve the data block. To generate an illegal state notification might be too harsh as ECC errors might go away after the next erase cycle. Providing a notification without informing the user about the block ID that was dropped does not seem helpful. FEE notification interfaces do not allow passing of parameters. When the user makes a request to read a configured block that was dropped during GC due to an ECC error, the job will end with a job result MEMIF_BLOCK_INCONSISTENT.

1.1.5.7 Ongoing erase cannot be cancelled

Ongoing erase, be it in NVM (during GC) or QS cannot be cancelled.

1.1.5.8 DET check for Fee_JobErrorNotification and Fee_JobEndNotification

For the Fee_JobErrorNotification and Fee_JobEndNotification APIs, the FEE_E_UNINIT DET is detected and reported when these APIs are called before initializing the FEE module.

1.1.5.9 Init check

To help protect against corruption of FEE global data structures, in addition to checking if the provided configuration pointer is not NULL, a CRC based protection mechanism is implemented to protect the global data structure. The CRC is determined at the end of Fee_Init API and again as part of Fee_17_InitCheck API. The two values are compared to determine if the global data was corrupted after Fee_Init was called. For this to work correctly, it is assumed that the Fee_17_InitCheck will be called after Fee_Init API and before any other Fee API or the Fee_Mainfunction API is called. Fee_17_InitCheck API should be called before Fee_MainFunction API because it modifies the global data structure when it runs asynchronous operations needed for preparing the FEE to service user requests.

1.1.5.10 Handling of invalid sectors

The FeeEraseAllEnable configurable parameter is provided, for the user, to allow erasing sectors when the sectors are found to be invalid.

1.1.5.11 Behavior of Fee_17_EraseQuasiStatic data for QS blocks with multiple instances

The behavior of the Fee_17_EraseQuasiStaticData API is illustrated as follows.

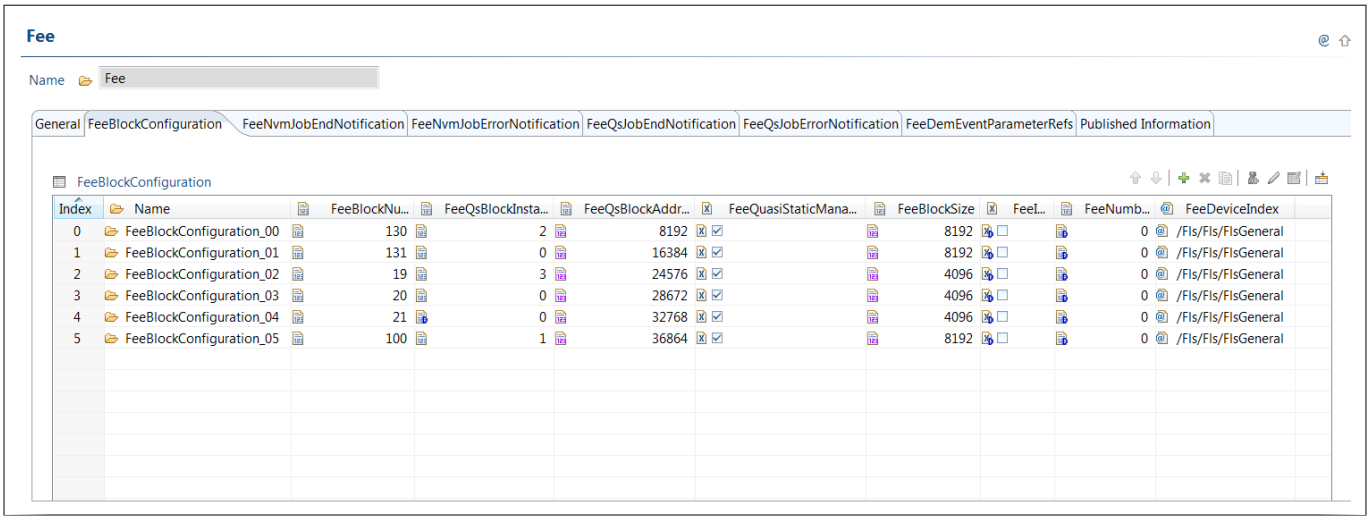


Figure 12 Sample QS block configuration

For the above-mentioned configurations, the Fee_17_EraseQuasiStaticData API would behave as follows:

Fee_17_EraseQuasiStaticData(19, 3): Blocks 19, 20, 21 will be erased

Fee_17_EraseQuasiStaticData(19, 1): Block 19 will be erased

Fee_17_EraseQuasiStaticData(19, 2): Blocks 19, 20 will be erased

Fee_17_EraseQuasiStaticData(21, 1): Block 21 will be erased

Fee_17_EraseQuasiStaticData(20, 2): Request rejected with INVALID_BLOCK_INSTANCES error will be raised

Fee_17_EraseQuasiStaticData(21, 0): Request rejected with INVALID_BLOCK_INSTANCES error will be raised

1.1.5.12 Behavior of illegal state notifications

If FeeVirginFlashIllegalState is set to TRUE, FEE invokes the illegal state notification upon detection of virgin Flash. Three cases arise: either the double sector area is in the virgin state or the QS area is in the virgin state or both areas are in the virgin state. If the area of the Flash used by the double-sector algorithm is found to be in the virgin state, the illegal state notification function configured in FeeNvmIllegalStateNotification is invoked and the illegal state notification function configured in FeeQsIllegalStateNotification will not be invoked irrespective of whether QS is in virgin or not. If the area used by the double-sector algorithm is not in the virgin state but the area used for Quasi-Static data is in the virgin state, then only the notification function configured in FeeQsIllegalStateNotification will be invoked. The user application should process the notification functions considering this behavior.

1.1.5.13 User mode support

FEE driver is a platform (device/hardware) independent module. It does not access any SFR directly.

[cover parentID FEE={1A65EADD-AFD0-4845-B2D2-8257E086DD67}]

1.2 Assumptions of Use (AoU)

The AoU for the FEE driver are as follows.

- **Association of QS job end notifications**

If the erase-suspend feature is enabled (FeeUseEraseSuspend=TRUE) and if a read or write request is made to a QS block when there is an ongoing QS erase operation, then it is assumed that the user shall associate the first job end notification with the completed QS read or write request and the second job end notification shall be associated with the completed erase operation.

[cover parentID FEE={A98061B0-8442-437f-938B-ADA60082DD87}]

- **Calling init-check API**

It is assumed that the Fee_17_InitCheck() API shall be called after calling Fee_Init and before any other FEE API or the Fee_Mainfunction is called.

[cover parentID FEE={38B90F4D-4D06-4acc-9076-F19EE0CE40F1}]

- **Canceling write and invalidate requests**

It is assumed that the ongoing user-initiated write and invalidate block request is not cancelled using the Fee_Cancel() API when the configuration parameter 'FeeOngoingWriteCancelSupport' is disabled. It is advisable that the module status be ascertained by making a call to the Fee_GetStatus() API and a new request be made only after the module status reaches MEMIF_IDLE.

[cover parentID FEE={6FC8C9A5-45A4-4d50-B7F7-07111FAC7677}]

- **Erase all when stateless**

When FeeEraseAllEnable = TRUE, and during initialization, it is found the Flash does not contain state pages, the area of the Flash used for the normal data (NVM) will be erased without any indication to the user. It is assumed that the user is aware of this behavior.

[cover parentID FEE={351399FF-6B79-4ba2-922C-B8B87E9D76BA}]

- **Erasing Immediate blocks**

Due to the way the double sector algorithm works, an immediate block cannot be erased. Therefore, AUTOSAR-defined Fee_EraseImmediateBlock API shall be implemented as a dummy function that always returns E_NOT_OK. It is assumed that the user will handle the implications of this behavior.

[cover parentID FEE={D0E18D9D-6DC2-4420-AFE2-F71830D81D22}]

- **FEE and pre-emption**

It is assumed that a call to an FEE service shall not be made so as to pre-empt another ongoing FEE service. This means that the FEE calls cannot be made from separate executing tasks in a manner that one call can pre-empt another. A call to a FEE service made from an interrupt context is also prohibited if an ongoing FEE service is interrupted.

The FEE services are non-reentrant and when an FEE service is pre-empted by another FEE service, internal data structures and state machines of the FEE become inconsistent.

[cover parentID FEE={8D3A9F16-281E-4073-B93E-0A0DB5BB5AA0}]

- **FEE single core execution**

It is assumed that the FEE software is called in the run-time application phase as well as in the boot phase from one core only.

[cover parentID FEE={BA2D7D53-75A9-4919-87BA-7F28A21A1493}]

- **Handling illegal state notification**

When an illegal state notification is generated by the FEE it means that the FEE module can no longer be used until the system is reset. Therefore, it is assumed that the user shall not make any further requests to the FEE after an illegal state notification has been received.

[cover parentID FEE={8BF148A6-BD68-4566-BFAB-B642C1355690}]

- **High priority QS write requests**

Before requesting a high priority write to a QS block, the Fee_17_CancelAll() API should be called to cancel all ongoing operations. If the GC is ongoing and the erase-suspend feature (FeeUseEraseSuspend) is configured as FALSE, then the high-priority write request will be serviced after the GC is completed.

[cover parentID FEE={21C52084-8CDF-4266-AE87-F3687BD0778F}]

- **QS block configuration**

It is assumed that the user shall adhere to the following rules while configuring the QS data blocks:

- Block addresses shall not overlap with the address of any other block
- Block size for the all instances of the block with multiple instances shall be same
- Block number shall be contiguous for the blocks having multiple instances
- Block instance number shall be 1 for blocks that do not have multiple instances
- Block instance number shall be zero if it is an instance of a block with multiple instances

[cover parentID FEE={6AB71A5D-A820-47ec-A9A6-3D2A67031E11}]

- **QS writes on pre-erased blocks**

When a QS block write is requested, the FLS performs a program verify error check by verifying the state of the PVER bit. If the PVER reports an error, the FEE causes a job error notification to be generated. One situation where this could occur is when the user application attempts to write to a QS block that was not erased. The FEE does not manage the erase of QS blocks. The user application is assumed to perform this.

It is assumed that the write to a QS block will be requested on a pre-erased QS block. The state of the QS block may be determined by calling the Fee_17_GetQuasiStaticBlockInfo() API. If it is determined that the block is invalid, the user may choose to erase the block.

[cover parentID FEE={26D6A684-6C95-4087-A7BF-97B04B0B8830}]

- **Re-enabling GC**

In order to re-enable the GC, it is assumed that the user shall first call Fee_Cancel to cancel any pending write job and then call Fee_17_EnableGCStart.

[cover parentID FEE={E1DE636E-5623-4874-87CB-52B8274BF4A0}]

- **Re-trying a QS request**

If a read (Fee_Read) or write (Fee_Write) request made to a quasi-static data block is returned with E_NOT_OK, it is assumed that the user application shall retry the request in a subsequent raster (after one cycles of the FEE main function).

If the erase-suspend feature is enabled and when the erase is ongoing in the hardware and when the FEE requests to suspend the erase to perform a QS write, a corner case arises when the erase operation has just finished in the hardware, but the FLS has not yet processed the event. In this case, the attempt to suspend the erase will fail, causing the QS block write operation to be rejected with E_NOT_OK. This situation is momentary and it is assumed that the user application will re-attempt the QS write request in a subsequent raster.

[cover parentID FEE={90C082AB-1FE4-420a-80CF-A9A6E3473515}]

- **Requesting cancel from interrupt context**

It is assumed that Fee_Cancel and Fee_17_CancelAll are not called from an interrupt context. As FEE services are asynchronous and non-re-entrant, calling Fee_Cancel or Fee_17_CancelAll from an interrupt context causes problems if the main task is already executing an FEE service. Fee_Cancel or Fee_17_CancelAll should also not be called in a manner that can cause it to pre-empt another FEE service.

[cover parentID FEE={D92B66E8-4B6F-4167-94D0-957C43D11A34}]

- **Retrying canceled jobs**

It is assumed that canceled jobs will be re-submitted by the application at some later point in time, after the QS block write is completed.

[cover parentID FEE={3F79DA18-B6E0-43e1-942C-9753B2CBAC97}]

- **Sensing mode configuration**

It is assumed that the Flash will be operated in the single ended sensing mode.

[cover parentID FEE={32FE1DAF-A76D-4146-80EE-0E5EED506DB0}]

- **User data protection**

User data is not protected by a CRC. It is assumed that the CRC forms a part of user data block and the user application will take the responsibility for checking the correctness of its data.

[cover parentID FEE={207D5CD1-185A-424e-A9B9-6A7EF6F7214D}]

- **Virgin state of Flash**

When in the virgin state, it is assumed that the user shall ensure that the entire area of DFlash0 allocated for the Double-Sector algorithm and Quasi-Static data is erased and contains no data.

[cover parentID FEE={2DE71581-F37F-4fd7-8271-06562F2A6647}]

1.3 Reference information

1.3.1 Configuration interfaces

Supported configuration variant: Post-Build

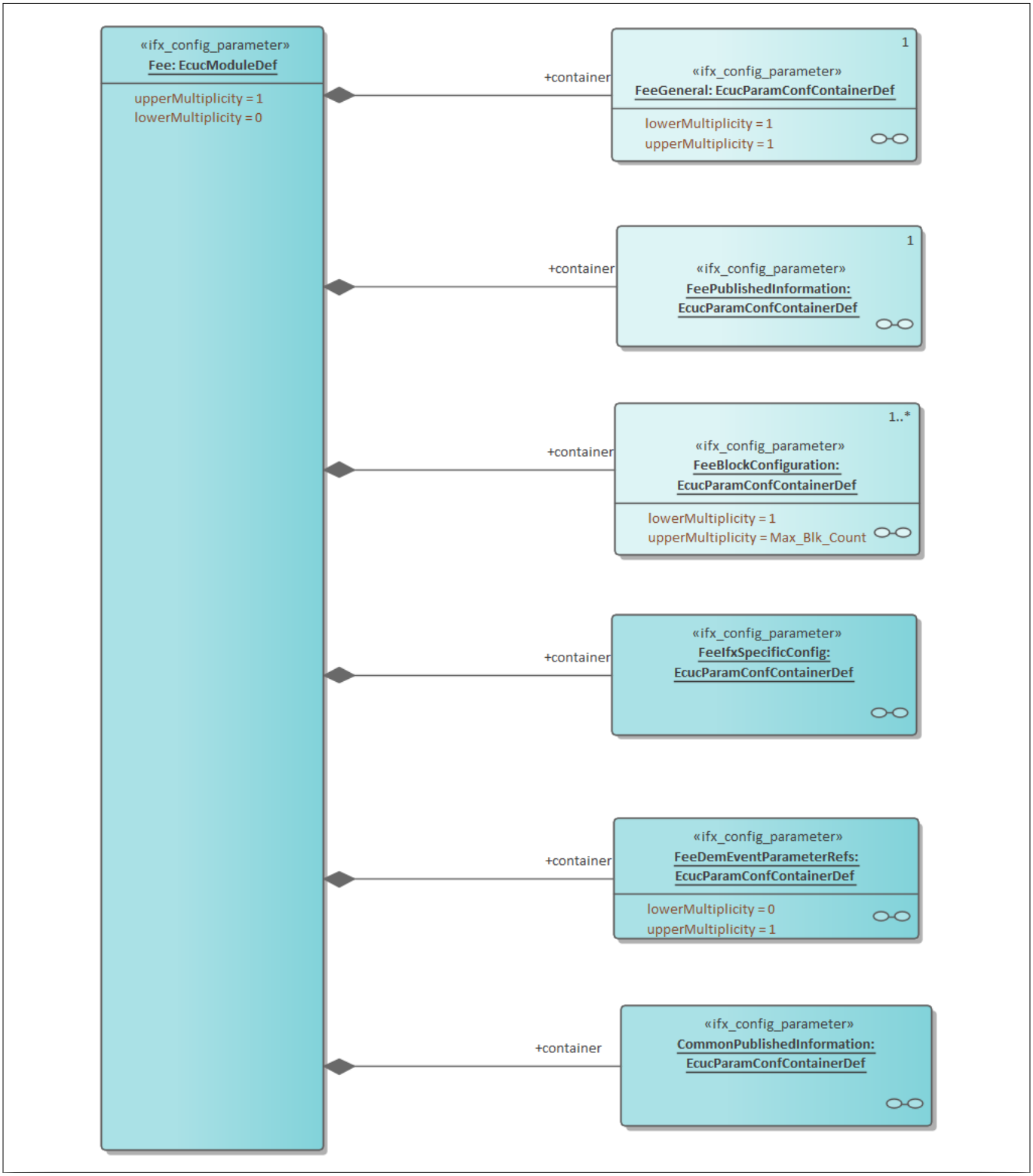


Figure 13 Container hierarchy along with their configuration parameters

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

1.3.1.1.2 ArMinorVersion

Table 5 Specification for ArMinorVersion

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

1.3.1.1.3 ArPatchVersion

Table 6 Specification for ArPatchVersion

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

1.3.1.1.4 ModuleId

Table 7 Specification for ModuleId

Name	`ModuleId`		
Description	Provides the FEE driver module ID as described by AUTOSAR : Wp1.1.2 Basic Software Module List.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	21		
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`

(table continues...)

Table 8 (continued) Specification for Release

Description	Specifies the derivative for which the configuration project is created.		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	As per the hardware derivative		
Post-build variant value	None	**Post-build variant multiplicity**	-
Value configuration class	None	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	None
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.6 SwMajorVersion

Table 9 Specification for SwMajorVersion

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

1.3.1.1.7 SwMinorVersion

Table 10 Specification for SwMinorVersion

Name	`SwMinorVersion`		
Description	Minor version number of the vendor specific implementation of the driver. The numbering is vendor specific.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef

(table continues...)

Table 10 (continued) Specification for SwMinorVersion

Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.1.8 SwPatchVersion

Table 11 Specification for SwPatchVersion

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

1.3.1.1.9 VendorId

Table 12 Specification for VendorId

Name	`VendorId`		
Description	Vendor Id of the supplier.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		

(table continues...)

Table 12 (continued) Specification for VendorId

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: Fee

This container holds the configurations of the FEE (Flash EEPROM Emulation) module.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

1.3.1.3 Container: FeeBlockConfiguration

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.3.1 FeeBlockNumber

Table 13 Specification for FeeBlockNumber

Name	`FeeBlockNumber`		
Description	Specifies the block number of the logical block. The value of this parameter should be unique across configurations. *Note: To find the physical address of the block, the FEE algorithm uses the cache table, that is, the physical address of the block is not calculated based on the block number using any static relation as suggested by AUTOSAR. Also for QS data the address is not directly calculated from the block number. Therefore, the block number can be any unique arbitrary number different from 0x0001 and 0xFFFE.* Since the minimum possible value for the block number is 1, therefore the default value is set as 1.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - 65534		
Default value	1		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-

(table continues...)

Table 13 (continued) Specification for FeeBlockNumber

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

1.3.1.3.2 FeeBlockSize

Table 14 Specification for FeeBlockSize

Name	`FeeBlockSize`		
Description	Specifies the size of the logical block. *Note:* *- For each of the NVM logical block, the FEE algorithm appends 8 bytes of header information at the beginning of the block and 1 page (8 bytes) of marker information at the end of the block. In between the header and the marker, each page (8 bytes) of the DFlash0 contains 1 byte of FEE internal information and 7 bytes of data of the logical block.* *Therefore, the total size (in bytes) occupied by the logical block in the DFlash0 is:* *If ((FeeBlockSize % 7) == 0), total size= ((FeeBlockSize / 7) * 8) + 16* *If ((FeeBlockSize % 7) != 0), total size= (((FeeBlockSize / 7) + 1) * 8) + 16* *Arithmetic of type integer is used in the calculation.* *- For QS blocks, the size should be an integer multiple of 4Kb and this also includes the block overhead.* *- The Block size range depends on the configured sector size for NVM and QS.* Since the minimum possible value for the block size is 1 therefore, the default value is set as 1.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - 65535		
Default value	1		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.3 FeeDeviceIndex

Table 15 Specification for FeeDeviceIndex

Name	`FeeDeviceIndex`		
Description	Specifies the device index (handle). This information is needed by the NVRAM manager to direct a request to the memory abstraction layer (MemIf) to address a certain logical block. Since the name of the dependent container is user configurable, the default value is kept as NULL.		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: FlsGeneral		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.4 FeeImmediateData

Table 16 Specification for FeeImmediateData

Name	`FeeImmediateData`		
Description	Specifies the type (priority) of the logical block. It is applicable when the FEE module is configured as Double-Sector only or Double-Sector and Quasi-Static. The parameter is set to FALSE for normal data and TRUE for immediate data.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.5 FeeNumberOfWriteCycles

Table 17 Specification for FeeNumberOfWriteCycles

Name	`FeeNumberOfWriteCycles`		
Description	Defines the block write cycle count of a particular logical block. It denotes the maximum number of times a particular logical block can be written. However, the value 0 denotes that this block is not bound by any limit and can be written as many times as desired by the user. It is applicable when the FEE module is configured as Double-Sector only or Double-Sector and Quasi-Static. The minimum value is chosen as the default value.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.6 FeeQsBlockAddress

Table 18 Specification for FeeQsBlockAddress

Name	`FeeQsBlockAddress`		
Description	Specifies the address of The QS block in the Flash. In case of NVM blocks, this parameter is not used. The address should be 4K aligned. It is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both. minimum value is chosen as the default value.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - Fls Range		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-

(table continues...)

Table 18 (continued) Specification for FeeQsBlockAddress

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

1.3.1.3.7 FeeQsBlockInstances

Table 19 Specification for FeeQsBlockInstances

Name	`FeeQsBlockInstances`		
Description	Specifies the number of block instances in case of parent QS block. Multiple QS blocks can exist together and each of the QS block can have more than one instance. This parameter is used for configuring the number of instances within one QS block. For example, if there are 4 instances of QS block with id = 0x400, 0x401, 0x402 and 0x403 then the number of instances for block id = 0x400 is 4. Also, for the rest of the blocks with id = 0x401, 0x402 and 0x403, the number of instances = 0. The block numbers for the instances should be configured sequentially as illustrated. It is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both. The minimum value is chosen as the default value.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - (Fls Total Size) / 4K		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.8 FeeQuasiStaticManager

Table 20 Specification for FeeQuasiStaticManager

Name	`FeeQuasiStaticManager`
Description	Specifies the user of the configured block. It is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both. The default block configuration is double sector data , therefore FeeQuasiStaticManager parameter is disabled by default.

(table continues...)

Table 20 (continued) Specification for FeeQuasiStaticManager

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	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4 Container: FeeDemEventParameterRefs

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.4.1 FEE_E_GC_ERASE

Table 21 Specification for FEE_E_GC_ERASE

Name	FEE_E_GC_ERASE		
Description	Existence of this parameter decides if the FEE driver would raise this particular production error. The short name of the referenced DEM parameter is used as the symbol that is passed to the Mcal_Wrapper_Dem_ReportErrorStatus function for the ASR 4.2.2 and Mcal_Wrapper_Dem_SetEventStatus function for the ASR 4.4.0 while reporting the error. Error description: error while executing erase operation during GC. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL

(table continues...)

Table 21 (continued) Specification for FEE_E_GC_ERASE

Dependency	FeeBlockTypeConfigured
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.4.2 FEE_E_GC_INIT

Table 22 Specification for FEE_E_GC_INIT

Name	FEE_E_GC_INIT		
Description	Existence of this parameter decides if the FEE driver would raise this particular production error. The short name of the referenced DEM parameter is used as the symbol that is passed to the Mcal_Wrapper_Dem_ReportErrorStatus function for the ASR 4.2.2 and Mcal_Wrapper_Dem_SetEventStatus function for the ASR 4.4.0 while reporting the error. Error description: error during the Init GC activity. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.3 FEE_E_GC_READ

Table 23 Specification for FEE_E_GC_READ

Name	FEE_E_GC_READ		
Description	Existence of this parameter decides if the FEE driver would raise this particular production error. The short name of the referenced DEM parameter is used as the symbol that is passed to the Mcal_Wrapper_Dem_ReportErrorStatusfunction for the ASR 4.2.2 and Mcal_Wrapper_Dem_SetEventStatus function for the ASR 4.4.0 while reporting the error. Error description: error while executing the read operation during GC. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef

(table continues...)

Table 23 (continued) Specification for FEE_E_GC_READ

Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.4 FEE_E_GC_TRIG

Table 24 Specification for FEE_E_GC_TRIG

Name	`FEE_E_GC_TRIG`		
Description	Existence of this parameter decides if the FEE driver would raise this particular production error. The short name of the referenced DEM parameter is used as the symbol that is passed to the Mcal_Wrapper_Dem_ReportErrorStatus function for the ASR 4.2.2 and Mcal_Wrapper_Dem_SetEventStatus function for the ASR 4.4.0 while reporting the error. Error description: error due to insufficient space in the new sector for copying data blocks or state block. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.5 FEE_E_GC_WRITE

Table 25 Specification for FEE_E_GC_WRITE

Name	FEE_E_GC_WRITE		
Description	Existence of this parameter decides if the FEE driver would raise this particular production error. The short name of the referenced DEM parameter is used as the symbol that is passed to the Mcal_Wrapper_Dem_ReportErrorStatus function for the ASR 4.2.2 and Mcal_Wrapper_Dem_SetEventStatus function for the ASR 4.4.0 while reporting the error. Error description: error while executing the write operation during GC. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.6 FEE_E_INVALIDATE

Table 26 Specification for FEE_E_INVALIDATE

Name	FEE_E_INVALIDATE		
Description	Existence of this parameter decides if the FEE driver would raise this particular production error. The short name of the referenced DEM parameter is used as the symbol that is passed to the Mcal_Wrapper_Dem_ReportErrorStatus function for the ASR 4.2.2 and Mcal_Wrapper_Dem_SetEventStatus function for the ASR 4.4.0 while reporting the error. Error description: error while executing the user-triggered invalidate request. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE

(table continues...)

Table 26 (continued) Specification for FEE_E_INVALIDATE

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

1.3.1.4.7 FEE_E_READ

Table 27 Specification for FEE_E_READ

Name	FEE_E_READ		
Description	Existence of this parameter decides if the FEE driver would raise this particular production error. The short name of the referenced DEM parameter is used as the symbol that is passed to the Mcal_Wrapper_Dem_ReportErrorStatus function for the ASR 4.2.2 and Mcal_Wrapper_Dem_SetEventStatus function for the ASR 4.4.0 while reporting the error. Error description: uncorrectable ECC error while executing the user read request. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.8 FEE_E_UNCONFIG_BLK_EXCEEDED

Table 28 Specification for FEE_E_UNCONFIG_BLK_EXCEEDED

Name	FEE_E_UNCONFIG_BLK_EXCEEDED

(table continues...)

Table 28 (continued) Specification for FEE_E_UNCONFIG_BLK_EXCEEDED

Description	Existence of this parameter decides if the FEE driver would raise this particular production error. The short name of the referenced DEM parameter is used as the symbol that is passed to the Mcal_Wrapper_Dem_ReportErrorStatus function for the ASR 4.2.2 and Mcal_Wrapper_Dem_SetEventStatus function for the ASR 4.4.0 while reporting the error. Error description: error due to exceeding the un configured number of blocks. Refer to the FeeMaxBlockCount configuration parameter. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.9 FEE_E_WRITE

Table 29 Specification for FEE_E_WRITE

Name	`FEE_E_WRITE`		
Description	Existence of this parameter decides if the FEE driver would raise this particular production error. The short name of the referenced DEM parameter is used as the symbol that is passed to the Mcal_Wrapper_Dem_ReportErrorStatus function for the ASR 4.2.2 and Mcal_Wrapper_Dem_SetEventStatus function for the ASR 4.4.0 while reporting the error. Error description: error while executing the user write request. It is applicable when FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE

(table continues...)

Table 29 (continued) Specification for FEE_E_WRITE

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

1.3.1.4.10 FEE_E_WRITE_CYCLES_EXHAUSTED

Table 30 Specification for FEE_E_WRITE_CYCLES_EXHAUSTED

Name	`FEE_E_WRITE_CYCLES_EXHAUSTED`		
Description	Existence of this parameter decides if the FEE driver would raise this particular production error. The short name of the referenced DEM parameter is used as the symbol that passed to the Mcal_Wrapper_Dem_ReportErrorStatus function for the ASR 4.2.2 and Mcal_Wrapper_Dem_SetEventStatus function for the ASR 4.4.0 while reporting the error. Error description: error due to exceeding the configured limit of the write cycles for the given block. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5 Container: FeeGeneral

Container for general parameters. These parameters are not specific to a block.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.5.1 FeeBlockTypeConfigured

Table 31 Specification for FeeBlockTypeConfigured

Name	`FeeBlockTypeConfigured`		
Description	Pre-processor switch to indicate the type of block configuration out of 3 allowed classifications, that is, whether only NVM data, or NVM and Quasi-Static data or only Quasi-Static data. Default value is set to Double sector data only as it is the most common configuration used.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA : FEE_DOUBLE_SECTOR_DATA_ONLY : FEE_QUASI_STATIC_DATA_ONLY :		
Default value	FEE_DOUBLE_SECTOR_DATA_ONLY		
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 FeeDevErrorDetect

Table 32 Specification for FeeDevErrorDetect

Name	`FeeDevErrorDetect`		
Description	Switches the DET detection and notification ON or OFF. TRUE: enabled (ON). FALSE: disabled (OFF). It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 32 (continued) Specification for FeeDevErrorDetect

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

1.3.1.5.3 FeeInitCheckApi

Table 33 Specification for FeeInitCheckApi

Name	`FeeInitCheckApi`		
Description	Enables the user to avail the functionality to check if FEE initialization is correct. True: Fee_17_InitCheck() API is enabled for use. False: Fee_17_InitCheck() API is disabled for use. The default value is set to FALSE for the optional feature to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.4 FeeMainFunctionPeriod

Table 34 Specification for FeeMainFunctionPeriod

Name	`FeeMainFunctionPeriod`		
Description	Period between successive calls to the main function (in seconds). 10ms is the widely used function period therefore, it is kept as the default value.		
Multiplicity	1..1	**Type**	EcucFloatParamDef
Range	0.001 - 1s		

(table continues...)

Table 34	(continued) Specification for FeeMainFunctionPeriod		
Default value	10ms		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.5 FeeNvmJobEndNotification

Table 35	Specification for FeeNvmJobEndNotification		
Name	`FeeNvmJobEndNotification`		
Description	Mapped to the job end notification routine provided by the upper layer module (NvM_JobEndNotification). It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both. By default, the optional notification is disabled to minimize the executable code size.		
Multiplicity	0..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.6 FeeNvmJobErrorNotification

Table 36	Specification for FeeNvmJobErrorNotification
Name	`FeeNvmJobErrorNotification`

(table continues...)

Table 36 (continued) Specification for FeeNvmJobErrorNotification

Description	Mapped to the job error notification routine provided by the upper layer module (NvM_JobErrorNotification). It is applicable when FEE module is configured as Double-Sector only or Double-Sector and Quasi-Static. The optional notification is disabled by default to minimize the executable code size.		
Multiplicity	0..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.7 FeePollingMode

Table 37 Specification for FeePollingMode

Name	`FeePollingMode`		
Description	Pre-processor switch to enable or disable the polling mode for this module. This parameter is set to FALSE and is non-editable. TRUE: Polling mode enabled, callback functions (provided to the FLS module) are disabled. FALSE: Polling mode disabled, callback functions (provided to the FLS module) are enabled. Infineon implementation of FEE works only in the non-polling mode. Therefore the default value is set to FALSE and is non-editable.		
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 continues...)

Table 37 (continued) Specification for FeePollingMode

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

1.3.1.5.8 FeeQsJobEndNotification

Table 38 Specification for FeeQsJobEndNotification

Name	`FeeQsJobEndNotification`		
Description	Mapped to the job end notification routine for the completed QS jobs provided by the upper layer module that uses the FEE services for QS blocks. The configured function is invoked by the FEE when a QS job completes successfully. The user is expected to provide a function of type void FeeQsJobEndNotification (void) if a notification for a completed QS job is required. If no notification is required, this parameter should be configured as NULL_PTR. The notification function provided is expected to be synchronous. This parameter is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both. By default the optional notification is disabled by default to minimize the executable code size.		
Multiplicity	0..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.9 FeeQsJobErrorNotification

Table 39 Specification for FeeQsJobErrorNotification

Name	`FeeQsJobErrorNotification`

(table continues...)

Table 39 (continued) Specification for FeeQsJobErrorNotification

Description	Mapped to the job error notification routine for the failed QS jobs provided by the upper layer module that uses the FEE services for QS blocks. The configured function is invoked by the FEE when a QS job fails to complete successfully. The user is expected to provide a function of type void FeeQsJobErrorNotification (void) if a notification for a failed QS job is required. If no notification is required, this parameter should be configured as NULL_PTR. The notification function provided is expected to be synchronous. It is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both. By default he optional notification is disabled to minimize the executable code size.		
Multiplicity	0..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.10 FeeSafetyEnable

Table 40 Specification for FeeSafetyEnable

Name	`FeeSafetyEnable`		
Description	Enables the user to avail the functionality to detect and report the safety errors. By default the detection of safety related errors is enabled to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-

(table continues...)

Table 40 (continued) Specification for FeeSafetyEnable

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

1.3.1.5.11 FeeSetModeSupported

Table 41 Specification for FeeSetModeSupported

Name	`FeeSetModeSupported`		
Description	Compiler switch to enable or disable the SetMode functionality of the FEE module. TRUE: setMode functionality supported FALSE: setMode functionality not supported *Note:* *This configuration setting should be consistent with that of all underlying Flash device drivers (configuration parameter FlsSetModeApi).* It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both. The default value is set to disable for the optional feature to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.12 FeeVersionInfoApi

Table 42 Specification for FeeVersionInfoApi

Name	`FeeVersionInfoApi`

(table continues...)

Table 42 (continued) Specification for FeeVersionInfoApi

Description	Pre-processor switch to enable or disable the API to read the version information of the module. TRUE: Version information API enabled. FALSE: Version information API disabled. The default value is set to disable for the optional feature to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.13 FeeVirtualPageSize

Table 43 Specification for FeeVirtualPageSize

Name	`FeeVirtualPageSize`		
Description	Size in bytes to which logical blocks will be aligned. The value of this parameter is 8 and contrary to AUTOSAR this value cannot be changed and is fixed to 8. According to the hardware the page size is 8, therefore, the default value of FeeVirtualPageSize is 8.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	8 - 8		
Default value	8		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		

(table continues...)

Table 43	(continued) Specification for FeeVirtualPageSize
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.6 Container: FeeIfxSpecificConfig

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.6.1 FeeBlocksScannedPerCycle

Table 44 Specification for FeeBlocksScannedPerCycle

Name	`FeeBlocksScannedPerCycle`		
Description	This parameter allows the user to configure the number of blocks to be scanned in one Fee_MainFunction cycle during initialization. When this parameter is configured to 0, then all the blocks present in DFALSH will be scanned in one Fee_MainFunction cycle. In this case, the peak execution time of the Fee_MainFunction will be high. When this parameter is configured to a non-zero positive value, then the main function Fee_MainFunction will process only the configured number of blocks in each cycle. In this case, building the cache table will take several cycles of the Fee_MainFunction, however, the peak execution time of the Fee_MainFunction will be reduced. *Note: The user is advised to consider this effect while choosing a value for this configuration parameter.* *- The parameter is applicable only for the NVM section where double sector algorithm is used and not applicable for QS block region.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535 -		
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	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.2 FeeCancelAllApi

Table 45 Specification for FeeCancelAllApi

Name	`FeeCancelAllApi`

(table continues...)

Table 45 (continued) Specification for FeeCancelAllApi

Description	Pre-processor switch to enable or disable the Fee_17_CancelAll API. It is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both. By default, the Fee_17_CancelAll() API is disabled to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.3 FeeEccErrorInfoApi

Table 46 Specification for FeeEccErrorInfoApi

Name	`FeeEccErrorInfoApi`		
Description	Enables or Disables the service to read the information on most recent ECC error. By default, the Fee_17_GetEccErrorInfo API is disabled to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.4 FeeEraseAllEnable

Table 47 Specification for FeeEraseAllEnable

Name	`FeeEraseAllEnable`		
Description	Allows the user to configure if the sectors should be erased when FEE identifies an invalid sector state during initialization. When configured to TRUE, both the FEE sectors are erased. In such a case, the FEE resume to normal state but the previous data (if any) cannot be recovered. When configured to FALSE, the FEE settles in the illegal state and cannot continue to operate. However, data is kept intact in the DFlash and may be recovered. *Note: This is applicable only for the NVM section where double sector is used and not applicable for QS block region.* *It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.* By default FeeEraseAllEnable is disabled to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.5 FeeGcRestart

Table 48 Specification for FeeGcRestart

Name	`FeeGcRestart`

(table continues...)

Table 48 (continued) Specification for FeeGcRestart

Description	Specifies the GC restarting point. It is applicable when the FEE module is configured as Double-Sector only or Double-Sector and Quasi-Static both. Note: *Determination of NVM sector states, QS blocks dirty state and QS sector(area allocated for the QS blocks) virgin state is performed independent of this parameter during initialization.* FEE_GC_RESTART_INIT: the following operations are started after FEE initialization is completed: 1. Handling of NVM sector virgin or invalid or normal state. 2. Handling of QS sector virgin or QS blocks dirty state. 3. Cache build. 4. Interrupted GC. FEE_GC_RESTART_WRITE: the following operations are started when user read/QS read job is requested: 1. Handling of NVM sector virgin or invalid or normal state. 2. Handling of QS sector virgin or QS blocks dirty state. 3. Cache build. Interrupted GC will be started on the first write or invalidate or QS erase request.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	FEE_GC_RESTART_INIT: FEE_GC_RESTART_WRITE:		
Default value	FEE_GC_RESTART_INIT		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.6 FeeGetCycleCountApi

Table 49 Specification for FeeGetCycleCountApi

Name	`FeeGetCycleCountApi`
Description	Pre-processor switch to enable or disable the Fee_17_GetCycleCount() API. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both. By default, the get cycle count is disabled to minimize the executable code size.

(table continues...)

Table 49 (continued) Specification for FeeGetCycleCountApi

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	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.7 FeeGetPrevDataApi

Table 50 Specification for FeeGetPrevDataApi

Name	`FeeGetPrevDataApi`		
Description	Pre-processor switch to enable or disable the Fee_17_GetPrevData() API. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both. By default, the Fee_17_GetPrevData() API is disabled to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.8 FeeMaxBlockCount

Table 51 Specification for FeeMaxBlockCount

Name	`FeeMaxBlockCount`		
Description	Specifies the total number of blocks throughout all configurations. Note that the blocks which are shared across configurations are counted only once. This parameter decides the cache table size at pre-compile time (cache table size = FeeMaxBlockCount +10. The size of the cache table is increased by 10 as a safety margin). The cache table holds the information about configured, un-configured and QS blocks. Configuration of this parameter is carried out carefully. Higher value implies higher RAM area consumption for the cache table. The lower value would result in the loss of un-configured block(s) during GC (if FeeUnConfigBlkOverflowHandle is FEE_CONTINUE) or illegal state of FEE (if FeeUnConfigBlkOverflowHandle is FEE_STOP_AT_GC). Example 1: Configuration sets having mutually exclusive blocks: Configuration A: Number of blocks configured is 10. Configuration B: Number of blocks configured is 25. Then, FeeMaxBlockCount = 35. Example 2: Five blocks are shared/used by both Configuration A and Configuration B: Configuration A: Number of blocks configured is 10. Configuration B: Number of blocks configured is 25. Then, FeeMaxBlockCount = 30 (that is, shared blocks are counted only once).		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - Fee.NumberOfBlocks		
Default value	1		
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.		

1.3.1.6.9 FeeMaxBytesPerCycle

Table 52 Specification for FeeMaxBytesPerCycle

Name	`FeeMaxBytesPerCycle`

(table continues...)

Table 52 (continued) Specification for FeeMaxBytesPerCycle

Description	Specifies the maximum number of data bytes that are processed in one FEE main function call during the read, write and compare operations. The size is inclusive of the block overhead like header, consecutive page id, and so on. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both. Default value is set to the maximum bytes which can be processed in a cycle.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	FEE_MAX_BYTES_128: FEE_MAX_BYTES_256: FEE_MAX_BYTES_512: FEE_MAX_BYTES_64:		
Default value	FEE_MAX_BYTES_512		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.10 FeeNvmIllegalStateNotification

Table 53 Specification for FeeNvmIllegalStateNotification

Name	`FeeNvmIllegalStateNotification`		
Description	This parameter is a pointer to a notification API which is called when the FEE (NVM part) reaches an Illegal state. Illegal state means that the FEE is not able to proceed and the user should perform a power-on reset. NVM illegal notification can also be raised due to hardware errors during internal activities of FEE such as GC, initialisation of GC. It is applicable when FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both. By default, the optional notification is disabled to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		

(table continues...)

Table 53 (continued) Specification for FeeNvmIllegalStateNotification

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

1.3.1.6.11 FeeOngoingWriteCancelSupport

Table 54 Specification for FeeOngoingWriteCancelSupport

Name	`FeeOngoingWriteCancelSupport`		
Description	Configuration parameter to enable/disable cancel feature of an ongoing write and invalidate block operation in Fee_Cancel() API. If FeeOngoingWriteCancelSupport is enabled, an ongoing write and invalidate block operation can be cancelled by calling Fee_Cancel() API except GC else, an ongoing write and invalidate block operation cannot be cancelled by calling Fee_Cancel() API including GC . *- The parameter is applicable only for the NVM section where double sector algorithm is used and not applicable for QS block region.* *-This configuration parameter is recommended to be enabled only in scenarios when high priority write/invalidate job is expected during an ongoing operation from NVM. As there is a possibility of not detecting a word line error due to the skip of the Robustness algorithm during the cancellation of the ongoing write/invalidate job.*		
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	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.12 FeeQsHardenErrorNotification

Table 55 Specification for FeeQsHardenErrorNotification

Name	`FeeQsHardenErrorNotification`		
Description	Mapped to the hardening error notification routine provided by the upper layer module that uses the FEE services for the QS blocks. The configured function is invoked by the FEE when an error is encountered while performing a hardening operation. The user is expected to provide a function of type void FeeQsHardenErrorNotification (void) when a notification for a failed hardening operation is required. If no notification is required, this parameter should be configured as NULL_PTR. The notification function provided is expected to be synchronous. This parameter is applicable when the FEE module is configured as Double-Sector and Quasi-Static. By default, the optional notification is disabled to minimize the executable code size.		
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	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.13 FeeQsIllegalStateNotification

Table 56 Specification for FeeQsIllegalStateNotification

Name	`FeeQsIllegalStateNotification`		
Description	This parameter is a pointer to a notification API which is called when the FEE (QS part) reaches an Illegal state. Illegal notification is raised when the QS area happens to be in the virgin state and FeeVirginFlashIllegalState is not set. It is applicable when the FEE module is configured as Quasi-Static only or as Double-Sector and Quasi-Static both. By default, the optional notification is disabled to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucFunctionNameDef
Range	String		

(table continues...)

Table 56 (continued) Specification for FeeQsIllegalStateNotification			
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	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.14 FeeRunTimeErrorDetect

Table 57 Specification for FeeRunTimeErrorDetect			
Name	`FeeRunTimeErrorDetect`		
Description	This parameter enables or disables the Runtime errors reporting. When this parameter is set to TRUE, this enables the runtime errors reporting. The default value of this parameter is set to TRUE to ensure the runtime error detection during the product lifecycle. *Note: When FeeSafetyEnable is TRUE, this parameter must be set to TRUE.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.6.15 FeeStateVarStructure

Table 58 Specification for FeeStateVarStructure	
Name	`FeeStateVarStructure`

(table continues...)

Table 58 (continued) Specification for FeeStateVarStructure

Description	This parameter is a pointer to a structure which would contain all the global variables of the FEE driver. Using this, the user can allocate the space for the variables at his best to avoid any possible linking problems.		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node:		
Default value	Fee_StateVar		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.16 FeeThresholdValue

Table 59 Specification for FeeThresholdValue

Name	`FeeThresholdValue`		
Description	Describes the threshold value (in bytes before the end of the FEE sector) for triggering garbage collection/sector change. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - Fee Threshold Size		
Default value	200		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.17 FeeUnConfigBlkOverflowHandle

Table 60 Specification for FeeUnConfigBlkOverflowHandle

Name	`FeeUnConfigBlkOverflowHandle`

(table continues...)

Table 60 (continued) Specification for FeeUnConfigBlkOverflowHandle

Description	Specifies the behavior of the FEE driver when the cache table overflow occurs, that is, insufficient space in the cache table due to wrongly configured value of FeeMaxBlockCount (more number of blocks were detected during cache build, which cannot be accommodated in the cache table). FEE_CONTINUE: un-configured blocks which could not be accommodated in the cache table are lost after the GC. The FEE continues as expected for the currently active configuration. FEE_STOP_AT_GC: During the GC, the FEE enters a pseudo illegal state where only read operatipon is allowed but write is not allowed. Note: *If FeeUnConfigBlock is set to FEE_UNCONFIG_BLOCK_IGNORE then this parameter is irrelevant.* It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	FEE_CONTINUE: FEE_STOP_AT_GC:		
Default value	FEE_CONTINUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	FeeBlockTypeConfigured		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.18 FeeUnConfigBlock

Table 61 Specification for FeeUnConfigBlock

Name	`FeeUnConfigBlock`		
Description	Specifies whether unconfigured blocks should be copied to the new sector or ignored during the GC. It is applicable when the FEE module is configured as Double-Sector only or as Double-Sector and Quasi-Static both.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	FEE_UNCONFIG_BLOCK_IGNORE: FEE_UNCONFIG_BLOCK_KEEP:		
Default value	FEE_UNCONFIG_BLOCK_IGNORE		

(table continues...)

Table 61 (continued) Specification for FeeUnConfigBlock

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

1.3.1.6.19 FeeUseEraseSuspend

Table 62 Specification for FeeUseEraseSuspend

Name	`FeeUseEraseSuspend`		
Description	Specifies the usage of the erase-suspend feature provided by the hardware. If it is configured as TRUE, then user read, write and invalidate requests are serviced during the erase operation of the GC or QS block. If it is configured as FALSE, then user requests during erase operations are not accepted. By default, the erase-suspend feature is disabled to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.20 FeeVirginFlashIllegalState

Table 63 Specification for FeeVirginFlashIllegalState

Name	`FeeVirginFlashIllegalState`

(table continues...)

Table 63 (continued) Specification for FeeVirginFlashIllegalState

Description	Allows the user to configure the behavior of the FEE upon detection of virgin Flash (DFlash0). It can either be configured to call the configured illegal state notification function or to program the state blocks and perform normal operation. Values: FALSE: upon detection of virgin Flash (DFlash0) - FEE programs the initial state blocks and is available for user requests. TRUE: upon detection of virgin Flash (DFlash0) - FEE reaches illegal state and calls the configured illegal state notification function, if configured. By default, the FeeVirginFlashIllegalState is disabled to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7 Container: FeePublishedInformation

This container holds additional published parameters not covered by the CommonPublishedInformation container.

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

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.7.1 FeeBlockOverhead

Table 64 Specification for FeeBlockOverhead

Name	`FeeBlockOverhead`

(table continues...)

Table 64 (continued) Specification for FeeBlockOverhead

Description	Management overhead per logical block in bytes. - Block management overhead per logical NVM block in bytes is given by: (i) If ((FeeBlockSize % 7) == 0) then, ((FeeBlockSize / 7) * 8) + 16 - FeeBlockSize (ii) If ((FeeBlockSize % 7) != 0) then, (((FeeBlockSize / 7) + 1) * 8) + 16 - FeeBlockSize *Note: Integer arithmetic is used in the calculation.* - Block management overhead per logical QS block in bytes is minimum 36 bytes and can be more depending on the block size as minimum size allowed is 4k bytes Since the default FEE block size is 1 therefore, 17 is taken as the block overhead by default.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	17 - 17		
Default value	17		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.2 FeePageOverhead

Table 65 Specification for FeePageOverhead

Name	FeePageOverhead		
Description	Management overhead per page in bytes. This value is applicable only for the pages containing logical block data bytes for NVM (that is, not applicable for header and marker). For QS blocks, minimum block size to be configured is sector size which is 4k and this is not relevant. Overhead per page is 1 byte which is set as the default value.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - 1		
Default value	1		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL

(table continues...)

Table 65	(continued) Specification for FeePageOverhead
Dependency	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.2 Functions - Type definitions

1.3.2.1 Fee_BlockType

Table 66 Specification for Fee_BlockType

Syntax	`Fee_BlockType`	
Type	Structure	
File	`Fee.h`	
Range	unsigned_int CycleCountLimit : 24	FEE Block Cycle Count Range – [0…..16777215]
	unsigned_int FeeImmediateData : 8	Determine FEE Data FEE_NORMAL_DATA or FEE_IMMEDIATE_DATA Range: 0 - FEE_NORMAL_DATA 1 - FEE_IMMEDIATE_DATA
	unsigned_int BlockNumber : 16	FEE logical block number Range – [1……65534]
	unsigned_int Size : 16	Size of the logical block Range – [0……65535]
	unsigned_int Address : 32	FEE block address for quasi blocks only. This parameter is used for quasi feature only. Range – [0…..max size (device specific)]
	unsigned_int Instances : 16	FEE block instances for quasi blocks only. This parameter is used for quasi feature only. Range – [0……256]
	unsigned_int FeeUser : 8	FEE user type determination FEE_NVM_USER or FEE_QUASI_STATIC_USER Range: 0 - FEE_NVM_USER 1 - FEE_QUASI_STATIC_USER
Description	This type definition contains the types for the logical block configuration data.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.2 Fee_ConfigType

Table 67 Specification for Fee_ConfigType

Syntax	`Fee_ConfigType`	
Type	Structure	
File	`Fee.h`	
Range	-	None
Description	Configuration data structure of the FEE module. Elements of this structure are defined during the design phase.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.3 Fee_DataType

Table 68 Specification for Fee_DataType

Syntax	`Fee_DataType`	
Type	Enumeration	
File	`Fee.h`	
Range	1 - FEE_IMMEDIATE_DATA	None
	0 - FEE_NORMAL_DATA	None
Description	This type definition contains enumerations for the logical block types. FEE NVM block type: - Normal Type - Immediate Type	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.4 Fee_NotifFunctionPtrType

Table 69 Specification for Fee_NotifFunctionPtrType

Syntax	`Fee_NotifFunctionPtrType`
Type	Pointer to a function of type void Function_Name (void)
File	`Fee.h`
Description	Defines the function pointer type for the call back functions (job completion, job failure, illegal state).
Source	IFX
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.2.5 Fee_PageType

Table 70 Specification for Fee_PageType

Syntax	`Fee_PageType`	
Type	uint16	
File	`Fee.h`	
Range	uint16	
Description	Number of DFlash pages read/written.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.6 Fee_QsBlock_StateType

Table 71 Specification for Fee_QsBlock_StateType

Syntax	`Fee_QsBlock_StateType`	
Type	Enumeration	
File	`Fee.h`	
Range	0 - FEE_QS_PROG_STATE_ERASE_STARTED	Internal state to indicate an erase job has been initialized
	1 - FEE_QS_PROG_STATE_DESTROY	Internal state to indicate programmed state to be destroyed
	2 - FEE_QS_PROG_STATE_ERASE_COMPLETE	Internal state to indicate erase is completed
	3 - FEE_QS_PROG_STATE_WRITE_COMPLETE	Internal state to indicate write is complete
	4 - FEE_QS_PROG_STATE_WRITE_STARTED	Internal state to indicate write is started
	5 - FEE_QS_START_ERASE	Internal state to indicate an erase is requested
	6 - FEE_QS_START_BCC_WRITE	Internal state to indicate a write of the BCC
	7 - FEE_QS_START_BLOCK_WRITE	Internal state to indicate a start of the writing operation of the block
	8 - FEE_QS_ERASE_COMPLETE	Internal state to indicate Erase complete and erase complete state is properly set
	9 - FEE_QS_DIRTY_ERASE	Internal state to indicate Erase complete and erase complete state is properly set
	10 - FEE_QS_WRITE_COMPLETE	Internal state to indicate write complete and write state is properly set
	11 - FEE_QS_DIRTY_WRITE	Internal state to indicate write complete and write state is incomplete and nonzero

(table continues...)

Table 71 (continued) Specification for Fee_QsBlock_StateType

	12 - FEE_QS_ERASE_STARTED	Internal state to indicate erase started
	13 - FEE_QS_WRITE_STARTED	Internal state to indicate write started state is set and block partially written
	14 - FEE_QS_DESTROY	State in which QS marker pages are written with 0xFF
	25 - FEE_QS_INVALID	All states are invalid
Description	This type definition contains enumerations for QS Block State type.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.7 Fee_QuasiStaticBlockInfoType

Table 72 Specification for Fee_QuasiStaticBlockInfoType

Syntax	`Fee_QuasiStaticBlockInfoType`	
Type	Structure	
File	`Fee.h`	
Range	uint16 Bcc	Block Cycle count Range - [0.....65535]
	Fee_QsBlock_StateType State	Block state Range - Refer to the enum Fee_QsBlock_StateType for range
Description	This type definition contains the types for holding the block information about Block cycle count and Block state (for Quasi blocks only).	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.8 Fee_StateDataType

Table 73 Specification for Fee_StateDataType

Syntax	`Fee_StateDataType`	
Type	Structure	
File	`Fee.h`	
Range	To be elaborated in Design	None
Description	This type definition contains the types for holding the FEE driver status data.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.9 Fee_UserType

Table 74 Specification for Fee_UserType

Syntax	`Fee_UserType`	
Type	Enumeration	
File	`Fee.h`	
Range	0 - FEE_NVM_USER	None
	1 - FEE_QUASI_STATIC_USER	None
Description	FEE feature type selection: NVM OR Quasi.	
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 FEE driver.

1.3.3.1 Fee_17_CancelAll

Table 75 Specification for `Fee_17_CancelAll` API

Syntax	`void Fee_17_CancelAll ( void )`	
Service ID	0x28	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

(table continues...)

Table 75 (continued) Specification for `Fee_17_CancelAll` API

Description	Service to cancel any ongoing internal/user read or write job. However, the ongoing erase in the hardware cannot be cancelled. This API is expected to be called before a high priority QS data write is requested. When a user requested job (pending or ongoing) is cancelled, a job error notification is generated. If an internal operation such as garbage collection is cancelled, then no job error notification is generated. *Note:* *1) This API is not available when FEE is configured to support only Double Sector data (that is NVM data only).* *2) This API is not available if FeeCancelAllApi is disabled (FALSE).* *3) This API will not cancel module initialization related activities and no safety error(FEE_E_INVALID_CANCEL) will be raised.*
Source	IFX
Error handling	FEE_SE_UNINIT, FEE_SE_INVALID_CANCEL, FEE_E_INVALID_CANCEL
Configuration dependencies	FeeBlockTypeConfigured,FeeCancelAllApi
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.2 Fee_17_DisableGcStart

Table 76 Specification for `Fee_17_DisableGcStart` API

Syntax	`void Fee_17_DisableGcStart ( void )`	
Service ID	0x22	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

(table continues...)

Table 76 (continued) Specification for `Fee_17_DisableGcStart` API

Description	The user can call this API to prevent the GC from being started in case the threshold is crossed in the active FEE sector. This API does not stop an ongoing GC but only prevents the GC from being triggered by the write/invalidate request issued by the user. *Note: This API is applicable only for Double-Sector data.*
Source	IFX
Error handling	FEE_SE_UNINIT
Configuration dependencies	FeeBlockTypeConfigured
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.3 Fee_17_EnableGcStart

Table 77 Specification for `Fee_17_EnableGcStart` API

Syntax	`void Fee_17_EnableGcStart ( void )`	
Service ID	0x21	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This service allows enabling the trigger of GC, if GC is disabled earlier by calling the Fee_17_DisableGcStart() API. After this API is called, if the sector is filled up to the threshold level and additional write / invalidate request is issued, then GC is initiated. *Note: This API is applicable only for the Double-Sector data.*	
Source	IFX	
Error handling	FEE_SE_UNINIT	

(table continues...)

Table 77 (continued) Specification for `Fee_17_EnableGcStart` API

Configuration dependencies	FeeBlockTypeConfigured
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.4 Fee_17_EraseQuasiStaticData

Table 78 Specification for `Fee_17_EraseQuasiStaticData` API

Syntax	`Std_ReturnType Fee_17_EraseQuasiStaticData ( const uint16 BlockNumber, const uint16 Instances )`	
Service ID	0x25	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber Instances	Logical block number Number of block instances
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: the requested job has been accepted by the module. E_NOT_OK: the requested job has not been accepted by the module because the request is either pending or the QS is not initialized or the GC is running or the block is not found or the given block instance is incorrect
Description	Service to request an erase job for one or multiple consecutive instances of a Quasi-Static data block. *Note: This API is applicable only for the QS data block type.*	
Source	IFX	
Error handling	FEE_SE_INVALID_BLOCK_INSTANCES, FEE_SE_BUSY, FEE_SE_UNINIT, FEE_SE_INVALID_BLOCK_NO, FEE_E_BUSY	
Configuration dependencies	FeeBlockTypeConfigured	
User hints	-	

(table continues...)

Table 78 (continued) Specification for `Fee_17_EraseQuasiStaticData` API

SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.5 Fee_17_GetCycleCount

Table 79 Specification for `Fee_17_GetCycleCount` API

Syntax	`Std_ReturnType Fee_17_GetCycleCount ( const uint16 BlockNumber, uint32 * const CountPtr )`	
Service ID	0x20	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber	Logical block number
Parameters (out)	CountPtr	Pointer to the variable to which the cycle count is to be updated
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: the cycle count was read and returned successfully. E_NOT_OK: the function could not read the cycle count as FEE was busy or a read error occurred or the cache update was not complete or GC was ongoing.
Description	When called for a configured non-QS BlockNumber, the write cycle count of the given block is returned. When called with BlockNumber = 0, this routine delivers the FEE sector erase cycle count. *Note: This API is not applicable for the Quasi-Static data blocks.*	
Source	IFX	
Error handling	FEE_SE_INVALID_BLOCK_NO, FEE_SE_PARAM_POINTER, FEE_SE_UNINIT, FEE_SE_BUSY, FEE_E_BUSY	
Configuration dependencies	FeeBlockTypeConfigured,FeeGetCycleCountApi	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.6 Fee_17_GetEccErrorInfo

Table 80 Specification for `Fee_17_GetEccErrorInfo` API

Syntax	`Std_ReturnType Fee_17_GetEccErrorInfo ( uint16 * const BlockNumberPtr, uint32 * const PageAddressPtr )`	
Service ID	0x32	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	BlockNumberPtr PageAddressPtr	Block number with ECC error Address of page with ECC error
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: The requested job has been executed E_NOT_OK: The requested job has not been executed. Following condition will result in E_NOT_OK - - Fee is busy or underlying flash driver is busy - Data pointers passed are null
Description	Service to get the last ECC error information (Block number and page address where last ECC error is detected) Default value - If there is no ECC error detected, then default value for BlockNumber = 0xFFFF PageAddress = 0xFFFFFFFF - For any of the following condition if the block number is unknown, the default value of BlockNumber = 0xFFFF is used. 1. If the ECC error occurred on state page. 2. If the ECC error occurred during cache build ECC error detected on the block header page or marker page. 3. If the ECC error occurred during the GC copy phase. 4. If the ECC error occurred the GC erase verification phase. 5. If the ECC error occurred on un-configured blocks. *Note: The information about ECC error is not stored, only last ECC error information is retained.*	
Source	IFX	
Error handling	FEE_SE_UNINIT, FEE_SE_BUSY, FEE_SE_PARAM_POINTER, FEE_E_BUSY	

(table continues...)

Table 80 (continued) Specification for `Fee_17_GetEccErrorInfo` API

Configuration dependencies	FeeEccErrorInfoApi
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.7 Fee_17_GetPrevData

Table 81 Specification for `Fee_17_GetPrevData` API

Syntax	`Std_ReturnType  Fee_17_GetPrevData`
	`(`
	`  const uint16 BlockNumber,`
	`  const uint16 BlockOffset,`
	`  uint8 * const DataBufferPtr,`
	`  const uint16 Length`
	`)`	

Service ID	0x23	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber BlockOffset Length	Logical block number Address offset within the block Number of bytes to be read
Parameters (out)	DataBufferPtr	Pointer to data buffer
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: the read job is accepted E_NOT_OK: the read job is not accepted as there is a pending request (module is busy).
Description	This API reads one preceding occurrence of data (that is most recent one in history) of the given block. This API accepts the request and updates the FEE internal variables. However the actual reading of the data is done by the Fee_MainFunction after the cache is built. *Note: This is a Non-Autosar API and is applicable only for non-QS blocks.*	
Source	IFX	
Error handling	FEE_SE_UNINIT, FEE_SE_BUSY, FEE_SE_INVALID_BLOCK_LEN, FEE_SE_PARAM_POINTER, FEE_SE_INVALID_BLOCK_NO, FEE_SE_INVALID_BLOCK_OFS , FEE_E_BUSY	
Configuration dependencies	FeeGetPrevDataApi	

(table continues...)

Table 81 (continued) Specification for `Fee_17_GetPrevData` API

User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.8 Fee_17_GetQuasiStaticBlockInfo

Table 82 Specification for `Fee_17_GetQuasiStaticBlockInfo` API

Syntax	`Std_ReturnType Fee_17_GetQuasiStaticBlockInfo ( const uint16 BlockNumber, Fee_QuasiStaticBlockInfoType * const BlockInfoPtr )`	
Service ID	0x26	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber	Logical block number
Parameters (out)	BlockInfoPtr	Constant pointer to the BlockInfo structure
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK : QS block information is read successfully E_NOT_OK : QS block information is not available
Description	Service to read the Block State and the Block Cycle Counter of the given Quasi-Static data block instance. This service can be called to know the status of the QS block instance before a QS read, write or erase operation is requested. This service provides an opportunity to ascertain if a block instance was erased before a write request is made, or, if a previous block instance write was interrupted before a new read request is made. *Note 1: The status of the Fee needs to be idle before calling this api.* *Note 2: The QS block state values returned by this API are :* *FEE_QS_ERASE_STARTED : If the block erase was started* *FEE_QS_ERASE_COMPLETE: If the block erase was completed* *FEE_QS_WRITE_STARTED : If the write to the block started* *FEE_QS_WRITE_COMPLETE: If the writ to the block completed* *FEE_QS_INVALID : If the block was invalid*	
Source	IFX	

(table continues...)

Table 82 (continued) Specification for `Fee_17_GetQuasiStaticBlockInfo` API

Error handling	FEE_SE_INVALID_BLOCK_NO, FEE_SE_BUSY, FEE_SE_PARAM_POINTER, FEE_SE_UNINIT, FEE_E_BUSY
Configuration dependencies	FeeBlockTypeConfigured
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.9 Fee_17_GetQuasiStaticJobResult

Table 83 Specification for `Fee_17_GetQuasiStaticJobResult` API

Syntax	`MemIf_JobResultType Fee_17_GetQuasiStaticJobResult ( void )`	
Service ID	0x27	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	MemIf_JobResultType	MEMIF_JOB_OK : The last job has been finished successfully and there is no job pending MEMIF_JOB_PENDING : The last job is waiting for execution or currently being executed MEMIF_JOB_CANCELED : The last job has been cancelled MEMIF_JOB_FAILED: 1. The FEE driver has not been initialized (Fee_Init not called) 2. The last read/write/erase job failed.
Description	Service to query the result of the last accepted job issued by the QS Manager This API is applicable only if the FEE is configured to support the Quasi-Static data blocks. *Note: This API is applicable only for the QS data block type.*	
Source	IFX	
Error handling	FEE_SE_UNINIT	

(table continues...)

Table 83 (continued) Specification for `Fee_17_GetQuasiStaticJobResult` API

Configuration dependencies	FeeBlockTypeConfigured
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.10 Fee_17_InitCheck

Table 84 Specification for `Fee_17_InitCheck` API

Syntax	`Std_ReturnType  Fee_17_InitCheck`
	`(`
	`  const Fee_ConfigType * const ConfigPtr`
	`)`	

Service ID	0x30	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the selected Configuration set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Module is initialized properly. E_NOT_OK: Module is not initialized properly due to - Fee_CfgPtr is NULL - Fee_CfgPtr is not matching with given ConfigPtr. - Fee is not yet completely initialized.
Description	This function verifies the initialization of the FEE driver. *Note: The application should follow the following calling sequence:* *1. Call Fee_Init* *2. Call Fee_17_InitCheck* *Note: The Fee_17_InitCheck() API should be called before any other FEE API or Fee_MainFunction is called.*	
Source	IFX	
Error handling	-	
Configuration dependencies	FeeInitCheckApi	

(table continues...)

Table 84 (continued) Specification for `Fee_17_InitCheck` API

User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.11 Fee_Cancel

Table 85 Specification for `Fee_Cancel` API

Syntax	`void Fee_Cancel ( void )`	
Service ID	0x04	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

(table continues...)

Table 85 (continued) Specification for `Fee_Cancel` API

Description	Service to call the cancel function of the underlying flash driver. When an ongoing user requested read job is cancelled, the Infineon specific FLS service to cancel non-erase jobs is called. Ongoing user requested write and invalidate block jobs except GC is cancelled only If configuration parameter 'FeeOngoingWriteCancelSupport' is enabled. When an ongoing user requested write and invalidate block job is cancelled, then the FLS service for cancelling requested job shall be called. If a user requested read, write or invalidate job is not started and is held in pending state due an ongoing internal operation, these jobs can be cancelled by the Fee_Cancel API. Ongoing internal read, write and erase operations are not cancelled. Fee_Cancel should not be called for QS data. The service to cancel all ongoing jobs should be used instead. *Note: This API is applicable only for double sector data (NVM) block.* *Note: There are deviations taken for Aurix2G with respect to this AUTOSAR API.* *Note: When Fee_Cancel is called near to the block Write/ invalidate request completion, there is a possibility of having a valid write/ invalidate block still present in the flash.* *Note: As ongoing write request in hardware (command already requested from Flash) cannot be cancelled by Fee_Cancel API, user has to consider the write time as described below before issuing Fee_Write after Fee_Cancel. This will ensure that write request after cancel will not fail due to hardware busy.* *- 154 microseconds (Fee/Fls running only in HOST)* *- 5154 microseconds (Fee/Fls running in HOST and HSM parallelly)*
Source	AUTOSAR
Error handling	FEE_E_INVALID_CANCEL, FEE_E_UNINIT
Configuration dependencies	FeeOngoingWriteCancelSupport
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.12 Fee_EraseImmediateBlock

Table 86 Specification for `Fee_EraseImmediateBlock` API

Syntax	`Std_ReturnType  Fee_EraseImmediateBlock`
	`(`
	`  const uint16 BlockNumber`
	`)`

Service ID	0x09
Sync/Async	Synchronous

(table continues...)

Table 86 (continued) Specification for `Fee_EraseImmediateBlock` API

Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber	Logical Block Number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	-
Description	Service to erase an immediate logical block. Since the double-sector algorithm is used with the threshold limit for triggering GC, write requests of immediate block during GC can be accommodated within the pre-erased threshold area of the active FEE sector. Hence, this API is implemented as an empty function returning E_NOT_OK always.	
Source	AUTOSAR	
Error handling	-	
Configuration dependencies	-	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.13 Fee_GetJobResult

Table 87 Specification for `Fee_GetJobResult` API

Syntax	`MemIf_JobResultType Fee_GetJobResult ( void )`	
Service ID	0x06	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-

(table continues...)

Table 87 (continued) Specification for `Fee_GetJobResult` API

Return	MemIf_JobResultType	MEMIF_JOB_OK: the last job has been finished successfully and there is no pending job pending MEMIF_JOB_PENDING: the last job is waiting for execution or is being executed currently. This can happen in the following cases: 1.When there is a pending request waiting to be executed (GC/ Init on-going) 2.When there is a request being executed MEMIF_JOB_CANCELED : The last job has been cancelled MEMIF_JOB_FAILED : - The FEE driver has not been initialized (Fee_Init not called) - The last read/write/invalidate job failed. MEMIF_BLOCK_INCONSISTENT : - The requested block is inconsistent, it may contain corrupted data. - The requested block to be read is present in the configuration but is not written in DFlash yet MEMIF_BLOCK_INVALID : The requested block has been invalidated; the requested read operation cannot be performed
Description	Service to query the result of the last accepted job issued by the NVM.	
Source	AUTOSAR	
Error handling	FEE_E_UNINIT	
Configuration dependencies	-	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.14 Fee_GetStatus

Table 88 Specification for `Fee_GetStatus` API

Syntax	`MemIf_StatusType Fee_GetStatus ( void )`
Service ID	0x05
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info

(table continues...)

Table 88 (continued) Specification for `Fee_GetStatus` API

Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	MemIf_StatusType	MEMIF_UNINIT: the FEE driver has not been initialized (Fee_Init not called) MEMIF_IDLE: the FEE driver and the underlying Flash driver are currently idle. This can happen in the following cases: - When there is no pending request and GC is idle - When there is no pending request and GC cannot be started because GC restart point is FEE_GC_RESTART_WRITE - When there is no pending request and GC has entered a fail state - When there is no pending request and Init GC has entered a fail state - When the write has entered a fail state MEMIF_BUSY: the FEE driver is currently busy processing a request. MEMIF_BUSY_INTERNAL: the FEE module is busy with internal management operations
Description	Service to return the status of the driver. Though GC is considered as an internal management operation, the driver status is maintained as MEMIF_BUSY until the GC copy is over (this is for normal block). During GC erase the module state becomes MEMIF_BUSY_INTERNAL. In the case of immediate block the module status is MEMIF_BUSY_INTERNAL, the moment GC is triggered.	
Source	AUTOSAR	
Error handling	-	
Configuration dependencies	-	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.15 Fee_GetVersionInfo

Table 89 Specification for `Fee_GetVersionInfo` API

Syntax	`void Fee_GetVersionInfo ( Std_VersionInfoType * const VersionInfoPtr )`	
Service ID	0x08	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	VersionInfoPtr	Pointer to the standard version information structure.
Parameters (in - out)	-	-
Return	void	-
Description	Service to return the version information of the FEE module. *Note: This API is applicable if the FeeVersionInfoApi is enabled.*	
Source	AUTOSAR	
Error handling	FEE_E_PARAM_POINTER	
Configuration dependencies	FeeVersionInfoApi	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.16 Fee_Init

Table 90 Specification for `Fee_Init` API

Syntax	`void Fee_Init ( const Fee_ConfigType * const ConfigPtr )`
Service ID	0x00
Sync/Async	Asynchronous
Safety Level	Refer to the release notes for the safety related info
Re-entrancy	Non Reentrant

(table continues...)

Table 90 (continued) Specification for `Fee_Init` API

Parameters (in)	ConfigPtr	Pointer to the selected configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to initialize the FEE module.	
Source	AUTOSAR	
Error handling	FEE_E_PARAM_POINTER	
Configuration dependencies	-	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.17 Fee_InvalidateBlock

Table 91 Specification for `Fee_InvalidateBlock` API

Syntax	`Std_ReturnType  Fee_InvalidateBlock`
	`(`
	`  const uint16 BlockNumber`
	`)`	

Service ID	0x07	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber	Logical block number
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: the requested job has been accepted by the module. E_NOT_OK: the requested job has not been accepted by the module.

(table continues...)

Table 91 (continued) Specification for `Fee_InvalidateBlock` API

Description	Service to invalidate a logical block. This service initiates a write job to mark the block as invalid, make the module status as MEMIF_BUSY and set the job result to MEMIF_JOB_PENDING. This API is only available for non-QS data blocks. For QS data blocks, the service to erase QS data blocks is to be used.
Source	AUTOSAR
Error handling	FEE_E_UNINIT , FEE_E_INVALID_BLOCK_NO, FEE_E_BUSY
Configuration dependencies	-
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.18 Fee_Read

Table 92 Specification for `Fee_Read` API

Syntax	`Std_ReturnType  Fee_Read`
	`(`
	`  const uint16 BlockNumber,`
	`  const uint16 BlockOffset,`
	`  uint8 * const DataBufferPtr,`
	`  const uint16 Length`
	`)`	

Service ID	0x02	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber BlockOffset Length	Logical block number. Read offset inside the block. Number of bytes to read
Parameters (out)	DataBufferPtr	Pointer to data buffer
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: the requested job has been accepted by the module. E_NOT_OK: the requested job has not been accepted by the module.

(table continues...)

Table 92 (continued) Specification for `Fee_Read` API

Description	This service initiates the read job for a QS or a non-QS block. If the length to be read is zero, then the function returns E_NOT_OK and no DET is raised.
Source	AUTOSAR
Error handling	FEE_E_INVALID_BLOCK_NO, FEE_E_INVALID_BLOCK_OFS , FEE_E_BUSY, FEE_E_PARAM_POINTER, FEE_E_INVALID_BLOCK_LEN, FEE_E_UNINIT
Configuration dependencies	-
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.19 Fee_SetMode

Table 93 Specification for `Fee_SetMode` API

Syntax	`void Fee_SetMode ( const MemIf_ModeType Mode )`	
Service ID	0x01	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	Mode	Desired mode for the underlying flash driver. The mode value can be MEMIF_MODE_SLOW or MEMIF_MODE_FAST
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to call the Fls_17_Dmu_SetMode function of the underlying Flash driver. *Note: This API is applicable if FeeSetModeSupport is enabled and if the FEE is configured to support the double-sector (NVM) data. If the Mode parameter passed to this function is other than MEMIF_MODE_SLOW or MEMIF_MODE_FAST then the error is detected and reported by the FLS module.*	
Source	AUTOSAR	
Error handling	FEE_E_UNINIT , FEE_E_BUSY	

(table continues...)

Table 93 (continued) Specification for `Fee_SetMode` API

Configuration dependencies	FeeBlockTypeConfigured,FeeSetModeSupported
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar version 4.2.2.

1.3.3.20 Fee_SetMode

Table 94 Specification for `Fee_SetMode` API

Syntax	`void Fee_SetMode ( const MemIf_ModeType Mode )`	
Service ID	0x01	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	Mode	Desired mode for the underlying flash driver. The mode value can be MEMIF_MODE_SLOW or MEMIF_MODE_FAST
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to call the Fls_17_Dmu_SetMode function of the underlying Flash driver. *Note: This API is applicable if FeeSetModeSupport is enabled and if the FEE is configured to support the double-sector (NVM) data. If the Mode parameter passed to this function is other than MEMIF_MODE_SLOW or MEMIF_MODE_FAST then the error is detected and reported by the FLS module.*	
Source	AUTOSAR	
Error handling	FEE_E_BUSY, FEE_E_UNINIT	
Configuration dependencies	FeeBlockTypeConfigured,FeeSetModeSupported	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.3.21 Fee_Write

Table 95 Specification for `Fee_Write` API

Syntax	`Std_ReturnType Fee_Write ( const uint16 BlockNumber, const uint8 * const DataBufferPtr )`	
Service ID	0x03	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	BlockNumber DataBufferPtr	Logical block number Pointer to data buffer
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: the requested job has been accepted by the module. E_NOT_OK: the requested job has not been accepted by the module.
Description	This service initiates the write to the given block number. Note: In case FEE_Write() is called continuously and the previous job is not completed, then FEE driver will return E_NOT_OK. If Safety or DET is enabled then FEE_E_BUSY will be reported.	
Source	AUTOSAR	
Error handling	FEE_E_PARAM_POINTER, FEE_E_UNINIT , FEE_E_INVALID_BLOCK_NO, FEE_E_BUSY	
Configuration dependencies	-	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.4 Notifications and Callbacks

This section lists all the notifications and callbacks of the FEE driver.

1.3.4.1 Fee_17_IllegalStateNotification

Table 96 Specification for `Fee_17_IllegalStateNotification` API

Syntax	`void Fee_17_IllegalStateNotification ( void )`	
Service ID	0x24	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This notification function is called by the underlying FLS driver when FLS driver reaches an illegal state.	
Source	IFX	
Error handling	FEE_SE_UNINIT	
Configuration dependencies	-	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.4.2 Fee_17_JobEraseErrorNotification

Table 97 Specification for `Fee_17_JobEraseErrorNotification` API

Syntax	`void Fee_17_JobEraseErrorNotification ( void )`
Service ID	0x29
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info
Re-entrancy	Non Reentrant

(table continues...)

Table 97 (continued) Specification for `Fee_17_JobEraseErrorNotification` API

Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to report to the FEE module the failure of an erase operation when EVER (Erase Verify) error occurred.	
Source	IFX	
Error handling	FEE_SE_UNINIT	
Configuration dependencies	-	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.4.3 Fee_17_JobProgErrorNotification

Table 98 Specification for `Fee_17_JobProgErrorNotification` API

Syntax	`void Fee_17_JobProgErrorNotification ( void )`	
Service ID	0x31	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to report to the FEE module when the Program Verify Error occurred while programming/writing.	
Source	IFX	

(table continues...)

Table 98 (continued) Specification for `Fee_17_JobProgErrorNotification` API

Error handling	FEE_SE_UNINIT
Configuration dependencies	-
User hints	-
SFR accessed	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.4.4 Fee_JobEndNotification

Table 99 Specification for `Fee_JobEndNotification` API

Syntax	`void Fee_JobEndNotification ( void )`	
Service ID	0x10	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to report to FEE module about the successful end of an asynchronous operation performed by the underlying Flash driver.	
Source	AUTOSAR	
Error handling	FEE_E_UNINIT	
Configuration dependencies	-	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.4.5 Fee_JobErrorNotification

Table 100 Specification for `Fee_JobErrorNotification` API

Syntax	`void Fee_JobErrorNotification ( void )`	
Service ID	0x11	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Service to report to the FEE module that the underlying flash driver (FLS) failed to perform an asynchronous operation.	
Source	AUTOSAR	
Error handling	FEE_E_UNINIT , FEE_E_INVALIDATE, FEE_E_WRITE, FEE_E_READ	
Configuration dependencies	-	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.5 Scheduled functions

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

1.3.5.1 Fee_MainFunction

Table 101 Specification for `Fee_MainFunction` API

Syntax	`void Fee_MainFunction ( void )`
Service ID	0x12
Sync/Async	NA

(table continues...)

Table 101 (continued) Specification for `Fee_MainFunction` API

Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The scheduled function helps to drive asynchronous jobs- read, write, erase and internal management jobs like garbage collection.	
Source	AUTOSAR	
Error handling	FEE_E_GC_ERASE, FEE_E_WRITE, FEE_E_READ, FEE_E_GC_INIT, FEE_E_GC_TRIG, FEE_E_GC_READ, FEE_E_WRITE_CYCLES_EXHAUSTED, FEE_E_UNCONFIG_BLK_EXCEEDED, FEE_E_GC_WRITE, FEE_E_INVALIDATE	
Configuration dependencies	-	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.6 Interrupt service routines

The FEE 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 the FEE driver.

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
FEE_E_GC_ERASE: Failure during the GC erase	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
FEE_E_GC_INIT: Failure in the GC during initialization	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
FEE_E_GC_READ: Failure during the GC read	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
FEE_E_GC_TRIG: GC triggering GC	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
FEE_E_GC_WRITE: Failure during the GC write	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
FEE_E_INVALIDATE: Failure during the Block invalidate	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
FEE_E_READ: Failure during the Block read	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
FEE_E_UNCONFIG_BLK_EXCEEDED: Unconfigured Block count limit reached	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
FEE_E_WRITE: Failure during the Block write	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
FEE_E_WRITE_CYCLES_EXHAUSTED: Block maximum write count exceeded	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
FEE_E_UNINIT : API service is called when the module is not initialized	AUTOSAR	0x01	DET_SAFETY	0x01	DET_SAFETY
FEE_SE_UNINIT: API service is called when the module is not initialized	IFX	0x01	SAFETY	0x01	SAFETY
FEE_E_INVALID_BLOCK_NO: API service is called with an invalid block number	AUTOSAR	0x02	DET_SAFETY	0x02	DET_SAFETY
FEE_SE_INVALID_BLOCK_NO: API service is called with an invalid block number	IFX	0x02	SAFETY	0x02	SAFETY
FEE_E_INVALID_BLOCK_OFS : API service is called with an invalid block offset	AUTOSAR	0x03	DET_SAFETY	0x03	DET_SAFETY
FEE_SE_INVALID_BLOCK_OFS : API service is called with the invalid block offset	IFX	0x03	SAFETY	0x03	SAFETY
FEE_E_PARAM_POINTER: API service is called with an invalid data pointer	AUTOSAR	0x04	DET_SAFETY	0x04	DET_SAFETY
FEE_SE_PARAM_POINTER: API service is called with an invalid data pointer	IFX	0x04	SAFETY	0x04	SAFETY
FEE_E_INVALID_BLOCK_LEN: API service is called with an invalid length information	AUTOSAR	0x05	DET_SAFETY	0x05	DET_SAFETY

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
FEE_SE_INVALID_BLOCK_LEN: API service is called with an invalid length information	IFX	0x05	SAFETY	0x05	SAFETY
FEE_E_BUSY: API service is called while the module is busy processing a user request. *Note: IFX API will not report this error in AUTOSAR version 4.2.2*	AUTOSAR	0x06	DET_SAFETY	0x06	RUNTIME
FEE_SE_BUSY: API service is called while the module is busy processing a user request	IFX	0x06	SAFETY	NA	NA
FEE_E_INVALID_CANCEL: API service is called while no job is pending. *Note: IFX API will not report this error in AUTOSAR version 4.2.2*	AUTOSAR	0x08	DET_SAFETY	0x08	RUNTIME
FEE_SE_INVALID_CANCEL: API service is called while no job is pending	IFX	0x08	SAFETY	NA	NA
FEE_SE_INVALID_BLOCK_INSTANCES: API service is called with invalid block instances	IFX	0x20	SAFETY	0x20	SAFETY

1.3.9 Deviations and limitations

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

1.3.9.1 Deviations

This section describes the deviation of the FEE driver.

1.3.9.1.1 Software specification deviations

This section describes the deviations from software specifications.

Table 102 Known deviations

Reference	Deviation
FEE as a precompile module	According to AUTOSAR, the FEE driver should be implemented as a pre-compile variant module. However, the Infineon FEE driver is implemented as a post-build variant.
FeeImmediateData	According to AUTOSAR, FeeImmediateData should be implemented as pre-compile variant. However, this configuration parameter is implemented as a post-build variant.

(table continues...)

Table 102 (continued) Known deviations

FeeNumberOfWriteCycles	According to AUTOSAR, FeeNumberOfWriteCycles should be implemented as pre-compile variant. However, this configuration parameter is implemented as a post-build variant.
FeeBlockSize	According to AUTOSAR, FeeBlockSize should be implemented as pre-compile variant. However, this configuration parameter is implemented as a post-build variant.
For all requirements related to Production/Runtime errors	Reporting of Production error: Dem_ReportErrorStatus is done through Mcal_Wrapper_Dem_ReportErrorStatus interface for AUTOSAR 4.2.2 and Dem_SetEventStatus is done through Mcal_Wrapper_Dem_SetEventStatus interface for AUTOSAR 4.4.0. Reporting of Runtime error: Det_ReportRuntimeError is done through Mcal_Wrapper_Det_ReportRuntimeError interface. This is applicable for only AUTOSAR 4.4.0. All production and runtime related datatypes and modified interfaces inclusion shall be done via Mcal_Wrapper.h.
Runtime error	The runtime error reporting is configurable, if user disables the runtime error reporting this is a deviation to AUTOSAR 4.4.0.

1.3.9.1.2 AMDC Violations

The Fee driver does not have any AMDC violations.

1.3.9.1.3 VSMD Violations

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

Table 103 Violation reported by VSMD checker tool for EB03

Rule ID:	EB03
VSMD Node(s):	/AURIX2G/EcucDefs/Fee/FeeGeneral/FeeNvmJobEndNotification /AURIX2G/EcucDefs/Fee/FeeGeneral/FeeNvmJobErrorNotification
Description:	The StMD node has LOWER-MULTIPLICITY=0 and UPPER-MULTIPLICITY=1. The VSMD-node shall get the OPTIONAL-attribute instead of creating a list!
Additional Information:	-

Table 104 Violation reported by VSMD checker tool for EB09

Rule ID:	EB09
VSMD Node(s):	/AURIX2G/EcucDefs/Fee
Description:	EB specific rule to check consistency of parameter postBuildVariantUsed.
Additional Information:	-

Table 105 Violation reported by VSMD checker tool for EcucSws_1014

Rule ID:	EcucSws_1014
VSMD Node(s):	/AURIX2G/EcucDefs/Fee/AURIX2G/EcucDefs/Fee/FeeBlockConfiguration/AURIX2G/EcucDefs/Fee/FeeGeneral
Description:	Additional vendor specific parameter definitions (using ParameterTypes), container definitions and references shall be added to the VSMD according to the alphabetical order.
Additional Information:	-

Table 106 Violation reported by VSMD checker tool for EcucSws_1035

Rule ID:	EcucSws_1035
VSMD Node(s):	"/AURIX2G/EcucDefs/Fee/AURIX2G/EcucDefs/Fee/FeeBlockConfiguration/AURIX2G/EcucDefs/Fee/FeeBlockConfiguration/FeeBlockNumber/AURIX2G/EcucDefs/Fee/FeeBlockConfiguration/FeeBlockSize/AURIX2G/EcucDefs/Fee/FeeBlockConfiguration/FeeDeviceIndex/AURIX2G/EcucDefs/Fee/FeeBlockConfiguration/FeeImmediateData/AURIX2G/EcucDefs/Fee/FeeBlockConfiguration/FeeNumberOfWriteCycles/AURIX2G/EcucDefs/Fee/FeeGeneral/AURIX2G/EcucDefs/Fee/FeeGeneral/FeeDevErrorDetect/AURIX2G/EcucDefs/Fee/FeeGeneral/FeeMainFunctionPeriod/AURIX2G/EcucDefs/Fee/FeeGeneral/FeeNvmJobEndNotification/AURIX2G/EcucDefs/Fee/FeeGeneral/FeeNvmJobErrorNotification/AURIX2G/EcucDefs/Fee/FeeGeneral/FeePollingMode/AURIX2G/EcucDefs/Fee/FeeGeneral/FeeSetModeSupported/AURIX2G/EcucDefs/Fee/FeeGeneral/FeeVersionInfoApi/AURIX2G/EcucDefs/Fee/FeeGeneral/FeeVirtualPageSize/AURIX2G/EcucDefs/Fee/FeePublishedInformation/AURIX2G/EcucDefs/Fee/FeePublishedInformation/FeeBlockOverhead"
Description:	For Containers, Parameters and References elements UUID must be unique (also between StMD and VSMD).
Additional Information:	-

Table 107 Violation reported by VSMD checker tool for EcucSws_2101

Rule ID:	EcucSws_2101
VSMD Node(s):	/AURIX2G/EcucDefs/Fee/POST_BUILD_VARIANT_USED
Description:	For each ConfigurationVariant supported by the ModuleDef, there must be one ImplementationConfigClass element. In VSMD, the ImplementationConfigClass is mandatory.
Additional Information:	-

Table 108 Violation reported by VSMD checker tool for EcucSws_6003

Rule ID:	EcucSws_6003
VSMD Node(s):	/AURIX2G/EcucDefs/Fee
Description:	The SHORT-NAME of the AR-PACKAGEs of StMD and VSMD must be different to ensure a unique SHORT-NAME-path.
Additional Information:	-

Table 109 Violation reported by VSMD checker tool for TpsEcuc_06049

Rule ID:	TpsEcuc_06049
VSMD Node(s):	/AURIX2G/EcucDefs/Fee
Description:	The supported EcucModuleDef.supportedConfigVariant shall be restricted in the VSMD to the actually supported configuration variants of this implementation. This can be a subset of the EcucModuleDef.supportedConfigVariant in the StMD.
Additional Information:	According to AUTOSAR, the FEE driver should be implemented as a pre-compile variant module. However, the Infineon FEE driver is implemented as a post-build variant to support different configuration for boot mode and runtime mode.

Table 110 Violation reported by VSMD checker tool for TpsEcuc_08036

Rule ID:	TpsEcuc_08036
VSMD Node(s):	/AURIX2G/EcucDefs/Fee/POST_BUILD_VARIANT_USED
Description:	If the valueConfigClass attribute for an EcucParameterDef or an EcucAbstractReferenceDef is not defined in the StMD, it shall be defined in the VSMD for all EcucParameterDefs and EcucAbstractReferenceDefs.
Additional Information:	-

Table 111 Violation reported by VSMD checker tool for TpsEcuc_08041

Rule ID:	TpsEcuc_08041
VSMD Node(s):	/AURIX2G/EcucDefs/Fee
Description:	If the postBuildVariantSupport attribute for an EcucModuleDef is set to false in the StMD, the corresponding VSMD shall also set it to false.
Additional Information:	According to AUTOSAR, the FEE driver should be implemented as a pre-compile variant module. However, the Infineon FEE driver is implemented as a post-build variant to support different configuration for boot mode and runtime mode.

1.3.9.2 Limitations

This section describes the limitation for Fee driver.

Table 112 Known limitations

Reference	Limitation
NVM write request during on going QS erase	If erase-suspend feature is enabled and if a QS block erase is on going, an NVM write request is not allowed. The API Fee_Write/ Fee_InvalidateBlock request made to a non-QS block will return E_NOT_OK in this situation. The reason is that the next NVM write could possibly trigger a GC and the erase as part of the GC cannot be handled by the hardware because a new erase cannot be triggered as there is an already suspended erase request (QS). [cover parentID FEE={956375E0-AFCC-4ea5-A508-7385754F54BD}]
Behavior of Fee_17_CancelAll()	An on going erase be it in NVM (during GC) or QS cannot be cancelled. The Fee_17_CancelAll() API has no effect when called immediately after API Fee_Init() before the execution of scheduled function (time-consuming operations are executed here). Therefore, it is advised to check the FEE module state and issue subsequent requests to FEE only after the module reaches the idle state. [cover parentID FEE={4BCD67F5-4630-4a81-99AC-CE8DB2367320}]
Block Cycle Count overflow	If Fee write cycle limit (FeeNumberOfWriteCycles) for the block is configured to zero then Fee writes are allowed even when the block cycle count rolls over the limit of 2^24. [cover parentID FEE={04AF6E7C-1E1A-4373-BEEE-43BD0CF0D380}]

(table continues...)

Table 112 (continued) Known limitations

Check and hardening time	The check and hardening process checks and hardens 2% of the area of flash memory allocated for QS data. In this area, if there are more pages that need hardening, then the time taken by the check and hardening process will increase, resulting in increased execution time peak. Example : If project has following configuration Device = TC39X (1 MB DFLASH) NVM sector = 4 KB QS sector = 1024-8= 1016 KB 2% of QS sector = 20 KB (approximately) FeeBlockTypeConfigured = FEE_DOUBLE_SECTOR_AND_QUASI_STATIC_DATA then the time taken by check and hardening process will 3.34 ms approximately. [cover parentID FEE={B0F8B663-696C-4456-8690-B6195FB7E929}]
Handling more than 2 word line failures	FEE is designed to handle up to two WL failures. After two WL failures, the Flash is considered to have gone bad and an illegal state notification is raised. [cover parentID FEE={16A0FBC9-1BA5-4dbf-9EA0-55A858FA5163}]
Handling rare state block combinations	State block combinations indicating (dirty valid, dirty erased), (valid, dirty valid) and (dirty erase, valid) are considered as rare cases. These situations can occur if the state blocks reside on a failed WL or due to aging. In these cases, repair is attempted and if repair fails, the state block is written to the next free WL and if this state block write fails, then illegal state notification is generated leading to loss of data. In these rare cases, FEE is not robust up to 2 WL failures. Two state pages in different sectors getting corrupted to indicate a particular invalid state combination AND a repair failing AND a write to the next+1 WL also failing - is expected to be an extremely rare case that the FEE does not handle. [cover parentID FEE={8CBC5576-1AE0-4f99-BB26-D8B156C08615}]
Handling user requests during check and hardening	When the check and hardening process is in progress, user read and write requests will not be accepted if the FEE module state is MEMIF_BUSY. User read write requests will be accepted if the module state is MEMIF_BUSY_INTERNAL but will be serviced with delay. The request will be serviced after the check and hardening process completes. A high priority QS write may be requested after calling Fee_17_CancelAll API. [cover parentID FEE={E9C7680C-4309-4e2e-84E6-969423B8C5DE}]

(table continues...)

Table 112 (continued) Known limitations

No notification for dropped blocks	The garbage collection process copies the latest instance of the data blocks by referring to the information present in the cache. If while reading a block during the copy phase, an un-correctable ECC error is encountered, then the block is dropped and it is not copied to the other sector. No notification is provided. The GC continues with the remaining blocks. This is a limitation mainly arising from the fact that notification interfaces cannot pass the block ID of the dropped block. [cover parentID FEE={01D4D541-E4A4-445b-B6A1-1D2D5EA145FF}]
Behavior of erase suspend resume feature during parallel access to DFlash0 and DFlash1 memory	When the flash memory on DFlash0 by TriCore and DFlash1 by HSM is accessed in parallel, then FSI gets into time-sliced mode of operation to cater both requests. In such scenario resume erase operation request by FEE to IFX FLS driver may lead to timeout failure. During retry there can be a situation where FLS resume erase operation is successful but the erase job end notification is never raised by FLS driver. In this situation FEE driver will hang. Hence, it is recommended not to use the erase suspend feature during simultaneous access of DFlash0 and DFlash1.
Behavior of hardening feature during parallel access to DFlash0 and DFlash1 memory	When the flash memory on DFlash0 by TriCore and DFlash1 by HSM is accessed in parallel, then FSI gets into time-sliced mode of operation to cater both requests. In such scenario hardening request by FEE to IFX FLS driver may lead to failure due to timeout. In such a scenario FEE will not perform hardening check and hardening of the current wordline or pages. The data may be lost if FEE hardening operation is not performed when needed. Hence it is recommended not to use QS and NVM data together during simultaneous access of DFlash0 and DFlash1.
Behavior of Fee_Cancel()	The ongoing user-initiated write and invalidate block requests are not canceled using the Fee_Cancel() API when the configuration parameter 'FeeOngoingWriteCancelSupport' is disabled. It is advisable that the module status be ascertained by making a call to the Fee_GetStatus() API and a new request be made only after the module status reaches MEMIF_IDLE.

Revision history

Table 113 Revision History

Date	Version	Description
2023-07-04	6.0	Document is released.
2023-06-14	5.1	- Dem module removed from upper layer and replaced with Mcal_Wrapper module in "1.1.2 Hardware-software mapping" section in " Figure 1 Mapping of hardware-software interfaces". - Dem.h removed and Mcal_Wrapper.h added in the "1.1.3.1 C file structure" section "Figure 2 Fee_C_File_Structure-1.png" and "Table 2 C file structure". - DEM module removed and Mcal_Wrapper module added in "1.1.4.1 Integration with AUTOSAR stack" section - Runtime error information are removed in DET module and added in Mcal_Wrapper module in "1.1.4.1 Integration with AUTOSAR stack" section - New notes are added in the below section, Note 2 added in "1.1.1 Description" section. Note 1 added in "1.3.3.8 Fee_17_GetQuasiStaticBlockInfo" Table 82 description section Note added in "1.3.3.21 Fee_Write" Table 95 description section - The Fee_17_GetQuasiStaticBlockInfo in Table 82 description corrected in "1.3.3.8 Fee_17_GetQuasiStaticBlockInfo" . - All references to Dem changed to production error, Dem_ReportErrorStatus changed to Mcal_Wrapper_Dem_ReportErrorStatus and Dem_SetEventStatus changed to Mcal_Wrapper_Dem_SetEventStatus in the following sections 1. FEE_E_GC_TRIG Production Error in "1.1.4.7 Example usage" 2. 1.3.1.4 Container: FeeDemEventParameterRefs 3. 1.3.8 Errors Handling - ASIL level field changed to Safety level with value as "refer to release notes" for all APIs under 1.3.3 Functions - APIs - Updated the section 1.3.9.1.1: Software Specification Deviations for Autosar requirements. Updated Reference from "DEM header file" to "For all requirements related to Production/Runtime errors". Updated Description of "DEM header file" to add Mcal_Wrapper Module Information.
2022-06-30	5.0	Document is released.
2022-06-28	4.1	-Added new configuration parameter 'FeeOngoingWriteCancelSupport'. -Updated 'Canceling Write and invalidate requests' AoU. -Description updated for Fee_Cancel() API. - Limitation updated for 'Behavior of Fee_Cancel()'.
2021-11-18	4.0	Document is released.
2021-11-16	3.1	Updated FeeDemEventParameterRefs section, Removed MemIf.h related deviation from software specific deviations section
2021-03-22	3.0	Document is released.

(table continues...)

Table 113 (continued) Revision History

2021-03-05	2.1	Limitation added for Fee_Cancel().
2020-12-07	2.0	Document is released.
2020-11-26	1.1	- Limitation for Cache build time is removed from limitation section. Following points added in intergration hints and example usage :- - `Fee_SetMode()` behavior for AUTOSAR version 4.4.0. - Number of state page processing during cache build for a main cycle - Concurrent access to Dflash 0 - Deviation added for file structure AUTOSAR version 4.2.2 MemIf.h inclusion.
2020-08-14	1.0	Document is released.
2020-07-31	0.1	- Initial version, chapter moved from MC-ISAR_TC3xx_UM_Basic.pdf. - Added following note in description – “The quasi-static data area has a limit of 500 erase/write cycles”. - Removed special character from key architecture in section Handling ECC errors during GC. - Added limitations for behaviour of erase suspend resume feature and hardening feature during parallel access to DFlash0 and DFlash1 by Tricore and HSM respectively. -Added deviation for DEM ASR 440 -Integration hint added 'Evaluation of disturbs in Quasi-Static area in DFLASH'.

Trademarks

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

Edition 2023-07-04

Published by

Infineon Technologies AG

81726 Munich, Germany

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

Email: erratum@infineon.com

Document reference
IFX-ocr1484806431059

Important notice

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

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

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

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

Warnings

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

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