Elektrobit

EB tresos® AutoCore Generic 8 Memory Stack documentation

release notes update for the Fee module

product release 8.8.7

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential information.

Table of Contents

1. Overview

This document provides you with the release notes to accompany an update to the `Fee` module. Refer to the changelog Section 2.1, “Change log” for details of changes made for this update.

Release notes details

- EB tresos AutoCore release version: 8.8.7
- EB tresos Studio release version: 29.2.1
- AUTOSAR R4.0 Rev 3
- Build number: B628559

2. Fee module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 2.0.0
- Module version: 6.14.24.B628559
- Supplier: Elektrobit Automotive GmbH

2.1. Change log

This chapter lists the changes between different versions.

Module version 6.14.24

2023-04-13

- ASCFEE-757 Fixed known issue: Blocks are not cached after a block has been invalidated

Module version 6.14.21

2022-10-26

- ASCFEE-725 Fixed known issue: Compiling error when SetMode is not available and VendorId is configured
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.14.20

2022-07-04

- Internal module improvement. This module version update does not affect module functionality.
- New Feature: Added support for MemAcc module.

Module version 6.14.19

2022-03-09

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.14.18

2021-12-15

- ASCFEE-681 Fixed known issue: Fee loses last block written in a section.

Module version 6.14.15

2021-12-03

- ASCFEE-669 New feature: Immediate reserved space
- ASCFEE-677 Fixed known issue: Fee does not work if the virtual page size is 512 bytes

Module version 6.14.14

2021-10-08

- ASCFEE-650 New feature: Blank check
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.14.13

2021-06-25

- Changed custom APIs: reject jobs for configured blocks
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.14.12

2021-03-05

- ASCFEE-632 New feature: Added Support for drivers that contain vendorId and vendorApiInfix
- ASCFEE-641 New feature: Added Small Section Support
- ASCFEE-611 New feature: Added Initialize in loop
- Behavior improvement: Fee_WriteCustom() accepts a different size for non configured block than the already exisiting size.
- Internal module improvement. This module version update does not affect module functionality

Module version 6.14.11

2020-10-23

- Internal module improvement. This module version update does not affect module functionality
- Internal module improvement. This module version update does not affect module functionality
- ASCFEE-606 New feature: Added Dynamic block length
- ASCFEE-612 Fee Blocks can be lost when an immediate block is erased in section 0

Module version 6.14.10

2020-06-19

- Internal module improvement. This module version update does not affect module functionality
- Internal module improvement. This module version update does not affect module functionality

Module version 6.14.8

2020-04-24

- ASCFEE-606 New feature: Added Fee_ReadCustom API
- ASCFEE-607 New feature: Caching reconfigured not-configured blocks

Module version 6.14.7

2020-02-21

- ASCFEE-585 Compatibility with Fls that requires only 32b aligned addresses. Internal module improvement. This module version update does not affect module functionality ASCFEE-603 Fixed known issue: Fee_WriteCustom() API does not work when the maximum number of blocks is used

Module version 6.14.6

2020-01-31

- ASCFEE-596 Fixed known issue: Fee Config ID is not generated correctly

Module version 6.14.5

2019-12-12

- ASCFEE-458 Write emergency block feature.
- Internal module improvement. This module version update does not affect module functionality
- ASCFEE-592 Fixed known issue: Fee generates incorrect configurations that trigger continuous switches at run-time.

Module version 6.14.4

2019-10-11

- ASCFEE-547 API extension to cancel ongoing page erase.
- Internal module improvement. This module version update does not affect module functionality

Module version 6.14.3

2019-06-14

- ASCFEE-542 Fee API to write non-configured blocks
- Behavior improvement: If section erase fails, allow requested jobs to be executed, then retry.
- Internal module improvement. This module version update does not affect module functionality

Module version 6.14.2

2019-04-18

- Internal module improvement. This module version update does not affect module functionality
- ASCFEE-486 Fixed known issue: FEE and FLS are no longer compatible due to AUTOSAR 4.0.3 / 4.-2.2 differences.

Module version 6.14.1

2019-02-15

- Internal module improvement. This module version update does not affect module functionality

Module version 6.14.0

2018-10-24

- Flexibility of memory layout (New features: Configurable number of sections, Switch not configured blocks, Freeze activities).

Module version 6.13.3

2018-10-23

- Internal module improvement. This module version update does not affect module functionality
- ASCFEE-479 Fixed known issue: FEE shall support a virtual page size of 256 bytes configured.
- ASCFEE-469 Fixed known issue: User write failure occurs during section switch, which leads to an over-writing of already existing flash
- ASCFEE-463 ASCFEE-463 Removed license check for Fee.ConsistencyPatterns
- ASCFEE-497 Fixed known issue: Failure of an immediate user write request during a sections switch causes the Fee to get stuck
- ASCFEE-485 Support for FEE configuration to use only a part of the FLS address space configured in the FLS driver.
- ASCFEE-456 Switch not configured blocks.
- Internal module improvement. This module version update does not affect module functionality
- Internal module improvement. This module version update does not affect module functionality
- ASCFEE-510 Internal module improvement. This module version update does not affect module functionality

Module version 6.13.2

2018-05-25

Module version 6.13.1

2018-02-16

- Internal module improvement. This module version update does not affect module functionality
- Internal module improvement. This module version update does not affect module functionality

Module version 6.13.0

2017-09-22

- Update Job handling when internal operations are ongoing.
- Switch to MISRA-C:2012. It does not affect module functionality.

Module version 6.12.0

2017-03-31

- ASCFEE-412 Fixed known issue: Fee could perform an out of bounds read access
- Internal module improvement. This module version update does not affect module functionality

Module version 6.11.1

2016-12-14

- ASCFEE-407 Fixed known issue: Fee could not compile with certain compilers

Module version 6.11.0

2016-11-04

- Internal module improvement. This module version update does not affect module functionality

Module version 6.10.1

2016-07-13

- Implemented data consistency pattern mechanism and additional robustness improvements

Module version 6.10.0

2016-05-25

- Added support for Debug & Trace with custom header file configurable via parameter `BaseDbgHeaderFile`

Module version 6.9.0

2016-02-08

- Improved the configuration parameters descriptions

Module version 6.8.0

2015-11-06

- Improved Fee robustness during start-up. Removed callout function in case the status of the sections can not be read from flash memory and implemented erase reaction in Fee
- ASCFEE-323 Fixed known issue: An already erased section may be erased again
- Added support for code flash sectors according to Renesas vendor specific parameter
- Internal module improvement. This module version update does not affect module functionality
- Implemented AUTOSAR Bugzilla RfC #58294: MEMIF_BLOCK_INCONSISTENT is returned for blocks that are not found in memory

Module version 6.7.0

2015-06-19

- Improved the Fee robustness if flash failure occurs during a write / invalidate request. Improved the Fee robustness if the section headers are FEE_SECTION_INCONSISTENT and FEE_SECTION_FULL
- ASCFEE-297 Fixed known issue: Fee remains in IDLE state without notifying upper layers if flash driver is corrupted
- ASCFEE-305 Fixed known issue: The Fee module doesn't check the module status before a mode switch is performed
- Improved the Fee robustness regarding the detection of erased sections. Implemented the flash memory erase counter functionality
- Improved Fee robustness in case reading the internal management data of a block fails during start-up

Module version 6.6.0

2014-10-02

- Updated module version for ACG-7.2.0 release

Module version 6.5.0

2014-04-25

- Added support for function tracing and variable debugging via AUTOSAR Debugging

Module version 6.4.0

2013-09-13

- Moved `DBG_FEE_STATE()` in `Fee_Init()` to just before initialization is completed

Module version 6.3.0

2013-06-25

- ASCFEE-249 Fixed known issue: The module behavior may be undefined if `Fee_Init()` is preempted by `Fee_MainFunction()`
- ASCFEE-262 Fixed known issue: The function `Fee_JobErrorNotification()` incorrectly sets the Fee flash job result to `MEMIF_JOB_FAILED` if the job result is not equal to `MEMIF_JOB_CANCELED` or `MEMIF_JOB_PENDING`

Module version 6.2.1

2013-02-08

- Internal module improvement. This module version update does not affect module functionality

Module version 6.2.0

2012-10-12

- Changed `Fee_Init()` to an asynchronous interface
- ASCFEE-196 Fixed known issue: Compiler warnings may occur due to incorrect size of `FEE_SECTION_STATUS_ERASED` when the Fls published value `FlsErasedValue` is larger than one byte
- Changed the top-level structure of the software-component description in the ARXML files from `/AUTOSAR/Fee` to `/AUTOSAR_Fee`

Module version 6.1.0

2012-06-15

- Added new Development Error `FEE_E_INVALID_CANCEL` in case module status is `MEMIF_BUSY` from `Fee_Cancel`
- ASCFEE-183 Fixed known issue: `Fee_MainFunction` is not invoked by Rte

Module version 6.0.0

2012-03-16

- Updated naming scheme for #defines for symbolic name values to AUTOSAR 4.0 Rev 3 naming scheme
- Updated Fee with respect to AUTOSAR R4.0 Rev 2
- Provided BSWMD for Fee

Module version 5.0.0

2011-09-30

- Initial AUTOSAR 4.0 version

2.2. New features

- No new features have been added since the last release

2.3. Elektrobit-specific enhancements

This chapter lists the enhancements provided by the module.

- Jump table and bootloader support

  Description:

  The global and static variables of the Fee are implemented as elements of a single structure. The Integrator is able to map these variables to a dedicated memory or Flash section of his choice.

  Rationale:

  This enhancement implements the HIS requirements HisFee0003 and HisFee0004 which enable the EB Fee to be used from multiple applications by using jump tables, e.g. for the use in a bootloader.

- Configuration parameter `FeeMainFunctionPeriod`

  Description:

  The vendor-specific configuration parameter `FeeMainFunctionPeriod` is implemented to specify the period with which `Fee_MainFunction` shall be invoked by the BSW scheduler.

  Rationale:

  The BSW scheduler of the Rte requires the `PERIOD` attribute of a timing event which triggers a schedulable entity in the Basic Software Module description. The Rte calculates the activation time of the OS schedule table and the execution time of a BSW main function within an OS task.

- Function tracing support via AUTOSAR Debugging

  Description:

  The module Fee supports tracing of function entry and exit via the EB Dbg module.

  Function tracing records following parameters for each function:

- function name
- values of the function arguments
- point in time of function invocation
- point in time of function termination
- return value of the function

- Variable Debugging

  Description:

  The Fee module supports debugging of variables containing the module's state and the next action to be performed.

  Variables available for debugging:

  - `Fee_State` (of type `Fee_State_t`)
  - `Fee_NextState` (of type `Fee_State_t`)

  `Fee_State_t` is defined as an enum with the following possible states:

  - `FEE_UNINIT`
  - `FEE_IDLE`
  - `FEE_INIT_READ_HEADER0`
  - `FEE_INIT_READ_HEADER1`
  - `FEE_INIT_ERASE`
  - `FEE_INIT_FILL_CACHE`
  - `FEE_INIT_CHK_LASTBLOCK`
  - `FEE_READ_BLOCKDATA`
  - `FEE_WRITE_BLOCK_INFO`
  - `FEE_WRITE_BLOCK_DATA`
  - `FEE_WRITE_SECTION_ACTIVE`
  - `FEE_WRITE_SECTION_FULL`
  - `FEE_WRITE_SECTION_EMPTY`
  - `FEE_WRITE_SECTION_NOT_EMPTY`
  - `FEE_WRITE_SECTION_ERASE_COUNTER`
  - `FEE_INVALIDATE`
  - `FEE_SECTION_SWITCHING`
  - `FEE_SS_READ_DATA`

- FEE_SS_COPY_INFO
- FEE_SS_COPY_DATA
- FEE_SS_WRITE_COPIED
- FEE_SS_ERASE_SECTION
- FEE_HANDLE_SECTION_HEADER_WRITE_FAILURE
- FEE_WRITE_BLOCK_INFO_START_PATTERN
- FEE_WRITE_BLOCK_DATA_START_PATTERN
- FEE_WRITE_BLOCK_INFO_END_PATTERN
- FEE_WRITE_BLOCK_DATA_END_PATTERN
- FEE_WRITE_SECTION_ACTIVE_END_PATTERN
- FEE_WRITE_SECTION_FULL_END_PATTERN
- FEE_WRITE_SECTION_COPIED_END_PATTERN
- FEE_WRITE_SECTION_EMPTY_END_PATTERN
- FEE_WRITE_SECTION_ERASE_COUNTER_END_PATTERN
- FEE_NUM_STATES
- FEE_STATE_INVALID

- Production error reporting

  Description:

  The vendor-specific parameter `FeeProdErrorDetect` allows enabling or disabling the detection of production errors. In case production error detection is enabled and Fee internal operations fail production errors can be reported to DEM. For details regarding the production errors please refer to the EB user documentation.

  Rationale:

  If errors occur during the the internal operations, Fee does not have any mechanism to report these error to the upper layer. Production errors have been added to notify the user.

- Flash memory erase counter

  Description:

  The vendor-specific configuration parameter `FeeEraseCounter` enables or disables the `Fee_GetEraseCounterValue()` API. The interface returns the approximate number of times Fee sections have been erased. The counter represents the number of erase of both sections, it is not section specific. Under unexpected working conditions, if the flash data cannot be recovered, the value of the erase counter will also be lost.

Rationale:

This feature gives a possibility to estimate how many times the flash has been erased and the user can get information regarding the lifetime of the flash device.

- Cancel Section Erase

  Description:

  The vendor-specific configuration parameter `FeeCancelSectionErase` enables or disables the `Fee_CancelSectionErase()` API. The interface allows the user to cancel an ongoing section erase if the highest internal priority is not active.

  Rationale:

  This feature gives a possibility to stop an ongoing section erase and free Fee/Fls, if this is allowed by the internal state of Fee.

- Write emergency block

  Description:

  The vendor-specific configuration parameter `FeeCriticalBlock` give highest priority to a write job of the block it referes to. This block, when requested, is written immediately. A successfull write of this block freezes Fee internally. An erase-immediate request for this block unfreezes Fee.

- Fls alignment compatibility

  Description:

  Fls needs aligned addresses: The vendor-specific configuration parameter `FeeUseBufferForJobs` enables or disables the compatibility with Fls drivers that require 32bit alignment for addresses that are passed as API parameters.

- Fee_WriteCustom API

  Description:

  The vendor-specific configuration parameter `FeeWriteCustomApi` enables the API Fee_WriteCustom that allows the user to write not-configured blocks.

- Fee_ReadCustom API

  Description:

  The vendor-specific configuration parameter `FeeReadCustomApi` enables the API Fee_ReadCustom that allows the user to read not-configured blocks.

- Caching reconfigured not-configured blocks

Description:

Newer instances of not-configured blocks that are found with reconfigured size will be cached at initialization.

- Allow Small Section

  Description:

  The vendor-specific parameter `FeeEnableSmallSectionSize` allows enabling or disabling the Fee Section size to be smaller than the total size of the blocks, in case the number of sections configured is more than 3 sections and the total size of the blocks fits in the (number of sections)/2 sections.

  Rationale:

  When the number of sections configured is more than 3 it is not mandatory to have the size of the section larger than the total size of the blocks configured as the large number of sections will compensate for the small size of each section.

- Dynamic block length

  Description:

  With this feature, Fee can detect and switch valid blocks that that have a different size in configuration. This means that for these blocks the configuration changed and the user wants to recover the existing data anyway. Otherwise their data would be lost. There are two ways the block length can be changed: by increasing the size or by decreasing the size. - If the block size is decreased in the current configuration Fee will restore as many data as it's requested, and sets the result to MEMIF_JOB_OK_SIZE_DECREASED. There is some extra data in the flash memory for the block, but the user doesn't have the space for it anymore, and it doesn't need it. - If the block size is increased in the current configuration Fee will restore the entire data from flash memory for this block and will add some padding bytes, in order to fill the user's bigger buffer. The padding byte value is the bitwise complement value of the last byte of data found in flash for this block. The user is responsable to manage this data recovery situation. The first successful write job for a reconfigured block will return everything to standard behavior concerning this feature for that particular block. Fee will switch these blocks always with the size that is written in flash and not by the current configuration. To use the feature, set the configuration parameters FeeDynamicBlockLength.

- Small Section Support

  Description:

  This feature allows enabling the total block size to be larger than smallest configured section when the number of configured sections is more than 3 and the total block size can fit in half of the configured sections.

- Drivers with vendorId and vendorApiInfix support

  Description:

This feature allows the use of flash drivers that use vendorId and vendorApiInfix

- Initialize in loop

  Description:

  In this feature a configuration parameter is added allowing to configure the initialization buffer to be generated with the smallest possible value to give the best performance in case the initialization will be done in a loop without using the OS.

- Blank Check

  Description:

  The Blank Check feature ensures that no read operation is performed on an empty virtual page, risking ECC errors to be reported. At Fee Startup Fls_BlankCheck is called whenever something needs to be read from flash. Due to this fact the Startup time will be increased a little. Fee does not perform any blank check if according to the writing flow it "knows" that the area could not be blank. Therefore, after the Startup is completed Fee will not use blank check any more. Patterns for data consistency are addapted to the feature. Only Section management data needs pattern.

- Abort Erase

  Description:

  The Abort Erase feature ensures that a write request for an immediate block coming during a section erasing will abort the erasing and it's performed right away, as long as the section erasing doesn't have the highest internal priority.

- Immediate Space Reserved

  Description:

  This feature allows the user to configure some space that will be reserved for immediate blocks. The space will be used in case an immediate write request comes while Fee is executing an internal operation with the highest priority. Fee will interrupt the internal operation in the favor of the immediate job, assuring that the immediated data is written as soon as possible. As the reserved space is limited, the immediate blocks that don't fit no more will be written after section switch completion.

2.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- Initialization check in main function

  Description:

If the main function is called while the module is not yet initialized the main function returns immediately without performing any functionality and without raising any Det error. This initialization check is always performed independent of the development error detection setting.

Rationale:

The SchM module may schedule the modules main function before the module is initialized. This would result in lots of Det errors during startup. Therefore the module's main function does not throw a Det error if the module is not yet initialized and simply returns in this case.

- Variable tracing by AUTOSAR Debugging

Description:

Only the module's state is available for debugging. The state gives information about the module's status and the current action. As long as the module's state is only used by the Fee module, it is not declared in the module's header file `Fee.h`. No information related to the job result and the block meta information are available for debugging.

Requirements:

FEE130, FEE131, FEE132

- `DataBufferPtr` type is `const uint8*` in `Fee_Write` API

Description:

In contrast to FEE088 in AUTOSAR Fee SWS V2.0.0, `DataBufferPtr` type is `const uint8*` in `Fee_Write` API.

Rationale:

In NvM R4.0.3 the parameter `NvM_SrcPtr` of `NvM_WriteBlock()` is changed to `const uint8*`, to make `Fee_Write()` compatible with `NvM_WriteBlock()`, `DataBufferPtr` type must be `const uint8*`. http://www.autosar.org/bugzilla/show_bug.cgi?id=55397 has been raised in AUTOSAR Bugzilla to fix the issue in the Fee SWS.

Requirements:

FEE088

- Internal Management Operation Handling

Description:

Fee internal management operation management is according to AUTOSAR Fee SWS V4.3.0. Requirements related to internal management operations handling that are exclusive to previous AUTOSAR specs are not supported any more.

Requirements:

FEE179, FEE180, FEE181, FEE182

- Block data interruption

Description:

Fee detects and handles internally the interruptions of the data writing. There is no need for making the block as corrupted before writing the data. It makes no sense to mark the block as corrupted anyway because otherwise a cancellation will lead to the impossibility of retrieving the previous data. This is an inconsistent behavior and it should be avoided.

Requirements:

FEE153

- Blank Check Enable

Description:

Due to backwards compatibility reasons the Blank Check functionality is OFF by default.

Requirements:

SWS_Fee_00187

- Blank Check Async

Description:

Fls_BlankCheck will not be called from Fee_Read context, but asynchronously from Fee_MainFunction.

Requirements:

SWS_Fee_00187

- MemAcc Usage

Description:

Fee SWS Autosar R21-11 requierments shall only apply if vendor specific parameter 'MemAccUsage' is set to 'true'.

Requirements:

ECUC_Fee_00157, ECUC_Fee_00155

2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- **Integration requirement: EB_INTREQ_Fee_0001**

  Callbacks

  Description:

  The EB memory stack modules NvM, Ea, and Fee make only limited use of the callback calls from their underlying modules. This also means that callbacks from the Fls to the Fee are not synchronously forwarded to the NvM. During the integration one has to make sure that the NvM, Ea, and Fee main functions are only called from the same task context so that they cannot preempt each other.

  Rationale:

  This approach enables a simple and lock-free implementation resulting in smaller code.

- **Integration requirement: EB_INTREQ_Fee_0002**

  Contiguous and ascending Flash sectors

  Description:

  The Flash sectors configured in `Fee_FlsSectorList` of the flash driver configuration should be contiguous if more than one sector is configured and they shall be in the ascending order of their addresses.

  Rationale:

  The logic used for calculating the Fee section size is dependent upon this configuration.

- **Integration requirement: EB_INTREQ_Fee_0003**

  Flash virtual page size re-configuration

  Description:

  If the flash virtual page size is reconfigured, the Fee module cannot retrieve the data blocks which are present in the flash.

  Rationale:

  Internal management information size and aligned block size are dependent on the virtual page size.

- **Integration requirement: EB_INTREQ_Fee_0004**

  Flash erase value limitation

Description:

Only the first byte of the Fls published parameter `FlsErasedValue` is used by the Fee module.

Rationale:

The Fls erased value is not expected to be different from one byte to another.

- **Integration requirement: EB_INTREQ_Fee_0005**

  Main functions

  Description:

  During the integration, the responsible for the MemStack modules has to make sure that the MemStack modules main functions are only called from the same task context and only from one task context so that they cannot preempt each other.

  Rationale:

  This approach enables a simple and lock-free implementation resulting in smaller code.

- **Integration requirement: EB_INTREQ_Fee_0006**

  Emergency job result

  Description:

  When the user requests an erase-immediate job for emergency block it shall wait for job completion by verifying the result of the job returned by `Fee_GetJobResult` API.

  Rationale:

  If other means are used the user shall still call at least once the API `Fee_GetJobResult` after termination of the erase immediate job. The reason for this is to be compatible with NvM behavior.

- **Integration requirement: EB_INTREQ_Fee_0007**

  Emergency freezes Fee

  Description:

  Once the emergency block is requested to be written, the user shall stop the `NvM_MainFunction`'s task until the emergency block is erased, when the task can be resumed.

  Rationale:

  User may read the emergency block any time, once the emergency block is written.

- **Integration requirement: EB_INTREQ_Fee_0008**

Emergency unfreezes Fee

Description:

If the emergency block is active in project, the user shall start the `NvM_MainFunction`'s task at startup after the emergency block is erased.

Rationale:

There is the possibility that the emergency block to be already written in flash by the time the system starts. In this case Fee will remain frozen internally until the user requests an erase-immediate of this block. Meanwile the user might read the emergency block any time.

- **Integration requirement: EB_INTREQ_Fee_0009**

Emergency erase write

Description:

If the user requested an erase-immediate for emergency block and immediately the environment requires the emergency block to be written again, the user shall not wait for the erase job to complete, but it shall request the emergency write job immediately as long as at least one `Fee_MainFunction` had been called after requesting the erase immediate job.

Rationale:

The signal to unfreeze Fee from emergency situation is given by the request of the erase immediate. Therefore after the request Fee is not frozen any more, and in case of another emergency write it should behave consistently. Mind the fact that the code `Fee_EraseImmediate(EBlock); Fee_Write(EBlock,Buff);` will not work since by the time the write job comes Fee would still be frozen internally. It needs on cycle of Fee main function after the erase-immediate request in order to unfreeze.

- **Integration requirement: EB_INTREQ_Fee_0010**

Emergency failing normal job

Description:

The environment shall be aware that if an emergency write is called when NvM has a job already pending in Fee, the NvM job will be reported as failed.

Rationale:

NvM is informed about the failure by either the notification callbacks or by NvM's polling mode. In this case the environment must take measures like retying the jobs.

- **Integration requirement: EB_INTREQ_Fee_0011**

Emergency preemption

Description:

If an emergency block is active in the project and a custom API(Fee_WriteCustom or Fee_ReadCustom) is used, the user shall consider the possibility of an emergency block write request being declined if it preempts a custom API: a retry could be needed, or preemption to be prevented by the system.

Rationale:

When Fee_WriteCustom or Fee_ReadCustom are configured, preemption protection is active for all Fee APIs, in order to prevent collision between NvM calling Fee API and a CDD calling Fee custom API. If the write of an emergency block comes at the same time with a custom API call that had already took the preemption protection lock, the emergency job will not be accepted and it has to be retried. Other measure would be to freeze the CDD that uses the Fee custom APIs in case of an emergency event.

- **Integration requirement: EB_INTREQ_Fee_0012**

  Custom APIs after init

  Description:

  Any call of Fee_ReadCustom/Fee_WriteCustom shall be made after Fee initialization is over(when Fee is Idle and not Busy Internal).

  Rationale:

  The environment must be aware that Fee_ReadCustom/Fee_WriteCustom depends on Fee initialization as the blocks that are not configured can only be detected at initialization.

- **Integration requirement: EB_INTREQ_Fee_0013**

  Custom APIs preemption

  Description:

  The system shall be aware that SCHM_FEE_EXCLUSIVE_AREA_0 is used if Fee_WriteCustom or Fee_-ReadCustom are configured.

  Rationale:

  The interrupts will be disabled only for checking and setting a preemption flag, that prevents call collisions of the custom APIs with the standard APIs called by MemIf.

- **Integration requirement: EB_INTREQ_Fee_0014**

  Big Buffer

  Description:

The user must allocate enough RAM for Fee buffer to accommodate the biggest block if the FeeUseBuffer-ForJobs is enabled.

Rationale:

If the lower layer has alignment requirement Fee will have to make sure that the address given to the lower layer during a job request is aligned. Therefore Fee uses the internal buffer as an intermediary. As Fee is limited in performing a job in one shot, the internal buffer needs to be big enough to fit any configured block.

- **Integration requirement: EB_INTREQ_Fee_0015**

Number of write cycles

Description:

The parameter FeeNumberOfWriteCycles is not used by current Fee implementation and it can be ignored during integration.

Rationale:

As opposed to Ea implementation Fee implementation doesn't use fixed addresses for writing blocks. Therefore FeeNumberOfWriteCycles could only be used as a bad approximation, since the total available space for a particular block depends very much on the writing frequency of other blocks in the project.

- **Integration requirement: EB_INTREQ_Fee_0016**

Read failing during section switch

Description:

In case of flash driver reporting failure during Section switch reading operation Fee will perform a single retry of the reading. If the reading fails again, the block will be skipped by Section switch and the data is lost. The integrator may use means of freezing the system after the first Fls read failure, until the system is steady again.

Rationale:

If the failure happens because of damaged flesh cells, there is no reason to consider that the memory cells will recover if keep retrying to read them. There is nothing to be done in this case. The switch must carry on.

- **Integration requirement: EB_INTREQ_Fee_0017**

Reading non-blank pages

Description:

If the flash technology doesn't guarantee a safe reading by blank-checking the same memory cells, the integrator shall configure 'FeeConsistencyPattern' in order to assure the consistency of the data and the avoidance of ECCs.

Rationale:

If consistency patterns are not used Fee is not able to tell whether the memory cells containing the section headers and footers are safe to be read on RV40F memory technology(used as an example). There are situations(quite probable) when blank checking the section management information is not enough for avoiding ECC errors. An incomplete erase/write can lead to this situation. There are also other situations(less probable) that may lead to data loss during Fee startup. Although EB Fee has mechanisms to assure data consistency up to a point, by using CRC in section management information, there are still situations, when the blank check performed over the section management information can fail because of an incomplete erase of the cells and the data will match the CRC because the hardware returns the previously written data. (By ignoring ECC the probability of this situation increases) Please consult RV40F technology documentation(erata).

- **Integration requirement: EB_INTREQ_Fee_0018**

Blank check and data retention

Description:

If the flash technology doesn't guarantee through blank-checking that a word can safely be written, the integrator shall configure 'FeeConsistencyPattern' in order to assure data retention.

Rationale:

An incompletely erased word may still be seen as blank. Although the writing over that word doesn't lead to any error, the data retention cannot be assured. Please consult RV40F technology documentation(erata).

- **Integration requirement: EB_INTREQ_Fee_0019**

Many sections and startup

Description:

If there are many sections configured (ex.: 16) and there are less frequently written blocks, the Fee startup may take too long. The integrator can mitigate this issue by configuring a lower number of sections (eg.: 4) for the same total flash size.

Rationale:

Section switch is designed to naturally keep half of the sections with data and half empty. Because of this sometimes less frequently written blocks remain hanging in the oldest section. Fee doesn't stop the block searching until they are found, leading to parsing half of the sections during startup.

- **Integration requirement: EB_INTREQ_Fee_00020**

Integrity of block management data

Description:

Integrity of the block's data is ensured by writing management information before and after writing the actual user data. If an unexpected shutdown occurs before the writing of the Written marker, the module retrieves the older block's instance if it exists.

Rationale:

The module ensures the integrity of blocks stored in flash by storing extra block management data.

2.6. Open-source software

Fee does not use open-source software.