Integration Manual

for MPC5634M FEE Driver

Document Number: IM14FEEASR3.0R2.0.0
Rev. 2.6

freescale™

Contents

Section Number **Title** **Page**

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
2.5	03-Feb-2011	Gaetano Stabile	Update for Monaco automatic documentation
2.6	06-Dec-2011	Gaetano Stabile	Added tips for fee integrator ch

Chapter 2
Introduction

This integration manual describes the integration requirements for FEE Driver for MPC5634M microcontrollers.

AUTOSAR FEE driver requirements and APIs are described in the AUTOSAR 3.0FEE Driver Software Specification Document (version V2.2.0 R3.0 Rev 0001) [Reference List]

The roadmap for the document is as follows:

[Build Options] This section gives a brief overview of the build procedure (compiler, linker options and source files) and EB tresos Studio plugin setup.

[Function Calls during Start-up] This section lists the various function calls to modules during Start-up, Shutdown and Wake-up.

[Exclusive areas to be defined in BSW scheduler]This section specifies the various module requirements related to:

- Exclusive areas to be defined in SchM module
- Peripheral Hardware Requirements
- ISR to configure within OS
- Specific interface to other modules
- Dependencies with other modules

[Main functions calls within BSW scheduler] This section specifies the requirements related to to the main API and gives a brief overview of the main functions calls within SchM module, and calls to notification functions, callbacks, callouts.

[Sections to be defined in MemMap.h] This section describes the memory allocation requirements namely the sections to be defined in MemMap.h file and the linker command file.

[Configuration Parameters] This section lists the FEE module configuration parameters and their variants/classes.

[Files required for Compilation] This section describes in brief the steps for integrating FEE module.

2.1 Supported Derivatives

The software described in this document is intented to be used with the following microcontroller devices of Freescale Semiconductor .

Table 2-1. MPC5634M Derivatives

Freescale Semiconductor	mpc5634m_bga208, mpc5634m_qfp144, mpc5634m_qfp176

All of the above microcontroller devices are collectively named as MPC5634M .

2.2 Overview

AUTOSAR (AUTomotive Open System ARchitecture) is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

AUTOSAR

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.
- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".
- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

Boldface type: Bold is used for important terms, notes and warnings.

Italic font: Italic typeface is used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

Note

This is a note.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
FEE	Flash EEPROM Emulation
FLS	Flash
MCU	Micro Controller Unit
VLE	Variable Length Encoding
XML	Extensible Markup Language

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	AUTOSAR AUTOSAR_SWS_Flash_EEPROM_Emulation.pdf FEE Driver Software Specification Document.	3.0 Rev0002
2	MPC5634M Reference Manual	Rev. 6, 4 October 2011

Chapter 3
Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar FEE driver for Freescale SemiconductorMPC5634M . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The FEE driver files are compiled using

- GHS 5.2.4
- DIAB 5_8_0_02 wind00198363 20100511 123238
- CW Version 4.3 build 182

The compiler, linker flags used for building the driver are explained below:

Note

The TS_T2D14M20I0R0 plugin name is composed as follow:

TS_T = Target_Id

D = Derivative_Id

M = SW_Version_Major

I = SW_Version_Minor

R = Revision

(i.e. Target_Id = 2 identifies PowerPC architecture and Derivative_Id = 14 identifies the MPC5634M)

3.1.1 CW Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-proc Zen	Generates and links object code for Zen processor. The compiler uses unsigned as the default parameter for the -char switch
-lang c	Expects source code to conform to the language specified by the ISO/IEC 9899-1990 ("C90") standard
-opt all	This option is selected all optimization (the same as -opt speed,level=4,intrinsics,noframe)
-common off	Disables moving uninitialized data into a common section
-sdatathreshold 0	Specifies the threshold size (in bytes) for an item considered by the linker to be small data. (The linker stores small data items in the Small Data address space. The compiler can generate faster code to access this data.)
-sdata2threshold 0	Specifies the threshold size (in bytes) for an item considered by the linker to be small constant data. (The linker stores small constant data items in the Small Constant Data address space.)
-vle	Tells the compiler and linker to generate and lay out Variable Length Encoded (VLE) instructions, available on Zen variants of Power Architecture processors
-use_lmw_stmw on	Enables the use of multiple load and store instructions for function prologues and epilogues
-ppc_asm_to_vle	Converts regular Power Architecture assembler mnemonics to equivalent VLE (Variable Length Encoded) assembler mnemonics in the inline assembler
-cpp_exceptions off	When on, generates executable code for C++ exceptions. When off, generates smaller, faster executable code
-func_align 4	Specifies alignment of functions in executable code
-sym dwarf-2,full	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format)
-gdwarf-2	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format). The linker ignores debugging information that is not in the Dwarf 1, Dwarf 2 format
-w on	Turns on most warning messages
-r	Compiler should expect function prototypes
-w undefmacro	Issues warning messages on the use of undefined macros in #if and #elif conditionals
-char unsigned	Controls the default sign of the char data type: char data items are unsigned
-nosyspath	Performs a search of both the user and system paths, treating #include statements of the form #include xyz the same as the form #include "xyz"
-fp none	No floating point code generation
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DUSE_SW_VECTOR_MODE	-D defines a preprocessor symbol and optionally can set it to a value. USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode.

Table continues on the next page...

Table 3-1. Compiler Options (continued)

Option	Description
-DMWERKS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the CWpreprocessor symbol.

Table 3-2. Assembler Options

Option	Description
-proc Zen	Generates and links object code for Zen processor. The compiler uses unsigned as the default parameter for the -char switch
-vle	Tells the compiler and linker to generate and lay out Variable Length Encoded (VLE) instructions, available on Zen variants of Power Architecture processors
-sym dwarf-2,full	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format)
-gdwarf-2	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format). The linker ignores debugging information that is not in the Dwarf 1, Dwarf 2 format.

Table 3-3. Linker Options

Option	Description
-proc Zen	Generates and links object code for Zen processor. The compiler uses unsigned as the default parameter for the -char switch
-code_merging all	Removes duplicated functions to reduce object code size
-far_near_addressing	Simplifies address computations to reduce object code size and improve performance
-vle_enhance_merging	Removes duplicated functions that are called by functions that use VLE instructions to reduce object code size
-listdwarf	DWARF debugging information in the linker's map file
-sym dwarf-2,full	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format)
-char unsigned	Controls the default sign of the char data type: char data items are unsigned.

3.1.2 DIAB Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
-tPPCE200Z3VEG:simple	Sets target processor to PPCE200Z3, generates ELF using EABI conventions, All Single Hardware Floating Point (Single precision uses hardware, double precision is mapped to single precision), selects simple environment settings for Startup Module and Libraries
-Xdialect-ansi	Follow the ANSI C standard with some additions
-XO	Enables extra optimizations to produce highly optimized code
-Xsize-opt	Optimize for size rather than speed when there is a choice
-Xsmall-data=0	Set Size Limit for "small data" Variables to zero.

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
-Xsmall-const=0	Set Size Limit for “small const” Variables to zero.
-Xno-common	Disable use of the “COMMON” feature so that the compiler or assembler will allocate each uninitialized public variable in the .bss section for the module defining it, and the linker will require exactly one definition of each public variable
-Xnested-interrupts	Allow nested interrupts
-Xalign-functions=4	Align each function on an address boundary divisible by 4
-g	Generate symbolic debugger information. Do most target-independent optimizations. Also, disable most target-dependent optimizations: option -g2 also disables basic reordering and all peephole optimizations.
-Xdebug-dwarf2	Generate symbolic debug information in dwarf2 format
-Xdebug-local-all	Force generation of type information for all local variables
-Xdebug-local-cie	Create common information entry per module
-Xdebug-struct-all	Force generation of type information for all typedefs, struct, union and class types
-Xforce-declarations	Generates warnings if a function is used without a previous declaration
-ee1481	Generate an error when the function was used before it has been declared
-Xforce-prototypes	Generate warnings if a function is used without a previous prototype declaration
-Xmacro-undefined-warn	Generates a warning when an undefined macro name occurs in a #if preprocessor directive
-Xlink-time-lint	Enable the checking of object and function declarations across compilation units, as well as the consistency of compiler options used to compile source files
-Xlint	Generate warnings when suspicious and non-portable C code is encountered. Enables all warnings
-ei1604	Suppress the warning messages 1604.
-W:as:,-l	Pass the option “-l” (lower case letter L) to the assembler to get an assembler listing file
-Wa,-Xisa-vle	Instruct the assembler to expect and assemble VLE (Variable Length Encoding) instructions rather than BookE instructions.
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DUSE_SW_VECTOR_MODE	-D defines a preprocessor symbol and optionally can set it to a value. USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode.
-DDIAB	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the DIAB preprocessor symbol.

Table 3-5. Assembler Options

Option	Description
-tPPCE200Z3VEN:simple	Selects target processor: PPCE200Z3, generates ELF using EABI conventions, NO floating point support, selects simple environment settings for Startup Module and Libraries.
-g	Dump the symbols in the global symbol table in each archive file.
-Xisa-vle	Expect and assemble VLE (Variable Length Encoding) instructions rather than Book E instructions. The default code section is named .text_vle instead of .text, and the default code section fill "character" is set to 0x44444444 instead of 0. The .text_vle code section will have ELF section header flags marking it as VLE code, not Book E code.
-Xasm-debug-on	Generate debug line and file information

Table 3-6. Linker Options

Option	Description
-tPPCE200Z3VEN:simple	Selects target processor: PPCE200Z3, generates ELF using EABI conventions, NO floating point support, selects simple environment settings for Startup Module and Libraries.
-Xelf	Generates ELF object format for output file
-m6	Generates a detailed link map and cross reference table
-lc	Specifies to linker to search for libc.a
-Xlink-time-lint	Enable the checking of object and function declarations across compilation units, as well as the consistency of compiler options used to compile source files.
-Xlibc-old	Enables usage of legacy (pre-release 5.6) libraries

3.1.3 GHS Compiler/Linker/Assembler Options

Table 3-7. Compiler Options

Option	Description
-cpu=ppc563xm	Selects target processor: ppc563xm
-ansi	Enforces strict ANSI mode (C89 standard)
-noSPE	Disables the use of SPE and vector floating point instructions by the compiler.
-Ospace	Optimize for size
-sda=0	Enables the Small Data Area optimization with a threshold of 0.
--no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup. This may improve optimizations by giving the compiler optimizer more information about the location of the variable.
-vle	Enables VLE code generation
-dual_debug	Enables the generation of DWARF, COFF, or BSD debugging information in the object file
-G	Generates source level debugging information and allows procedure call from debugger's command line.
--no_exceptions	Disables support for exception handling

Table continues on the next page...

Table 3-7. Compiler Options (continued)

Option	Description
-Wundef	Generates warnings for undefined symbols in preprocessor expressions
-Wimplicit-int	Issues a warning if the return type of a function is not declared before it is called
-Wshadow	Issues a warning if the declaration of a local variable shadows the declaration of a variable of the same name declared at the global scope, or at an outer scope
-Wtrigraphs	Issues a warning for any use of trigraphs
--prototype_errors	Generates errors when functions referenced or called have no prototype
--incorrect_pragma_warnings	Valid #pragma directives with wrong syntax are treated as warnings
-noslashcomment	C++ like comments will generate a compilation error
-preprocess_assembly_files	Preprocesses assembly files
-nostartfile	Do not use Start files
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DUSE_SW_VECTOR_MODE	-D defines a preprocessor symbol and optionally can set it to a value. USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode.
-DGHS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GHS preprocessor symbol.

Table 3-8. Assembler Options

Option	Description
-cpu=ppc563xm	Selects target processor: ppc563xm

Table 3-9. Linker Options

Option	Description
-cpu=ppc563xm	Selects target processor: ppc563xm
-nostartfiles	Do not use Start files.
-vle	Enables VLE code generation
-linker_warnings	Display linker warnings

3.2 Files required for Compilation

This section describes the include files required to compile, assemble (if assembler code) and link the FEE driver for MPC5634M microcontrollers.

To avoid integration of incompatible files, all the include files from other modules shall have the same AR_MAJOR_VERSION and AR_MINOR_VERSION, i.e. only files with the same AUTOSAR major and minor versions can be compiled.

FEE Files

- ..\Fee_TS_T2D14M20I0R0\src\Fee.c
- ..\Fee_TS_T2D14M20I0R0\include\Fee_Cbk.h
- ..\ Fee _ TS_T2D14M20I0R0 \include\Fee.h
- ..\ Fee _ TS_T2D14M20I0R0 \include\Fee_Api.h
- ..\ Fee _ TS_T2D14M20I0R0 \include\Fee_InternalTypes.h
- ..\ Fee _ TS_T2D14M20I0R0 \include\Fee_Types.h
- ..\ Fee _ TS_T2D14M20I0R0 \include\Fee_Version.h

- Fee_Cfg.h - this file should be generated by the user using a configuration/generation tool
- Fee_Cfg.c - this file should be generated by the user using a configuration/generation tool

Other includes files:

Files from MemIf folder:

- ..\MemIf_TS_T2D14M20I0R0\include\MemIf_Types.h

Files from Base common folder

- ..\Base_TS_T2D14M20I0R0 \include\Compiler.h
- ..\Base_TS_T2D14M20I0R0 \include\Compiler_Cfg.h
- ..\Base_TS_T2D14M20I0R0 \include\MemMap.h
- ..\Base_TS_T2D14M20I0R0\include\Mcal.h
- ..\Base_TS_T2D14M20I0R0 \include\Platform_Types.h
- ..\Base_TS_T2D14M20I0R0 \include\Std_Types.h

Files from Det folder:

- ..\Det_TS_T2D14M20I0R0 \include\Det.h

Files from Fls folder:

- ..\Fls_TS_T2D14M20I0R0 \include\Fls.h

3.3 Setting up the Plug-ins

The FEE driver was designed to be configured by using the EB Tresos Studio (version Tresos 2010a.sr4 20100415-release2010a-sr4 or later.)

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format: .. \Fee_TS_T2D14M20I0R0\config\Fee.xdm
- VSMD (Vendor Specific Module Definition) file in AUTOSAR compliant EPD format: ..\Fee_TS_T2D14M20I0R0\autosar\Fee.epd
- Code Generation Templates for Pre compile time configuration parameters:
  - ..\Fee_TS_T2D14M20I0R0 \generate\include\Fee_Cfg.h
  - ..\Fee_TS_T2D14M20I0R0 \generate\src\Fee_Cfg.c

Steps to generate the configuration:

1. Copy the module folders Fee_TS_T2D14M20I0R0, Fls_TS_T2D14M20I0R0, Base_TS_T2D14M20I0R0 , Resource_ TS_T2D14M20I0R0, into the Tresos plugins folder.
2. Set the desired Tresos Output location folder for the generated sources and header files.
3. Use the EB tresos Studio GUI to modify ECU configuration parameters values.
4. Generate the configuration files.

Chapter 4
Function calls to module

4.1 Function Calls during Start-up

FEE shall be initialized during STARTUP phase of EcuM initialization. The API to be called for this is Fee _Init(). The MCU module should be initialized before the FEE is initialized.

Notes:

- FEE module is the upper layer module which works on FLS module
- Fls_Init API must be called before calling Fee_Init API.
- Fls_MainFunction API and Fee_MainFunction API must be called periodically for FEE module initialization (after Fee_Init the: Fls_MainFunction and Fee_MainFunction must be called until Fee_GetStatus does not return MEMIF_IDLE) and its operation.

4.2 Function Calls during Shutdown

None.

4.3 Function Calls during Wake-up

None.

Chapter 5
Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

None.

5.2 Peripheral Hardware Requirements

The FEE module is hardware independent module and only depends on underlying FLS module and its configuration.

5.3 ISR to configure within OS – dependencies

None.

5.4 ISR Macro

None.

5.5 Other AUTOSAR modules - dependencies

- **Base**
- **Dem:** This module is necessary for enabling reporting of production relevant error status. The API function used is Dem_ReportErrorStatus().

- **Det:** This module is necessary for enabling Development error detection. The API function used is Det_ReportError(). The activation/deactivation of Development error detection is configurable using 'CanDevErrorDetect' configuration parameter.
- **MemIf:** Memory Interface
- **Resource:** Sub-Derivative model is selected from Resource configuration.
- **Flash:** The flash driver provides services for reading, writing and erasing flash memory and a configuration interface for setting / resetting the write / erase protection if supported by the underlying hardware.

Chapter 6
Main API Requirements

6.1 Main functions calls within BSW scheduler

None.

6.2 API Requirements

Before calling the function "Fee_Write" for immediate data, the function "Fee_EraseImmediateBlock" must be called to pre-erase the flash area

6.3 Calls to Notification Functions, Callbacks, Callouts

None.

6.4 Tips for FEE integration

Task scheduling

Make sure that no FEE/FLS functions are interrupted by another FEE/FLS functions (excluded Cancel operation)

Example 1: Fee_MainFunction() is called from 10 ms OS task and Fee_Write() function is called from 20 ms OS task. It has to be ensured that Fee_Write() function is not interrupted by the Fee_MainFunction().

Example2: Fee_MainFunction() is called from several places in the application. It has to be ensured that Fee_MainFunction() is not interrupted by another Fee_MainFunction().

Time consumption

1. Be aware of the maximum time consumption of the FEE/FLS functions (the best would be to use your actual configuration for performance analysis)

   Example 1: If Fee_MainFunction() takes maximum of 5ms (e.g. when swap occurs) it could be risky to call it from 1ms task.

   Example 2: Write operation takes 5ms to complete. When swap operation occurs during the write operation, it could take 500ms.

2. Be aware of the maximum number of FEE/FLS main functions or overall time needed for operations (read, write, …) to finish (the best would be to use your actual configuration for performance analysis)

   Example 1: If "Fls erase blank check" is used, 64 kB clusters are used and "Max erase blank check" is set to 256 Bytes, it may take up to 600 FEE/FLS main functions to complete the swap operation. So if the main functions are called within 10 ms task it can take 6s to finish the swap operation.

   Note: presented timing numbers are just an example, please refer to the profiling report to get the real numbers.

FEE050

According to the FEE050 requirement, when the write operation starts, corresponding block is marked as inconsistent. It is marked as consistent after successful write. So it means that when write operation is interrupted (cancel, reset …) application cannot rely anymore on the block data. There is a configuration parameter (FEE Block Always Available) which allows to set the behavior against the FEE050 requirement. So it means that FEE will always provide the last valid information. Valid information means FEE_BLOCK_VALID (block is valid) or FEE_BLOCK_INVALID (block was invalidated or block is not present in the cluster at all).

Example 1: Driver's behavior is set according FEE050. One instance of the block is successfully written to the memory. Another write operation of this block is scheduled but it is interrupted before finish (by Fee_Cancel() or power down). Block can be valid containing old data (operation was interrupted before write process started), block can be valid containing new data (operation was interrupted just before returning success information) or block can be invalid/inconsistent (ongoing write process was interrupted) .

Example 2: Driver behavior is set to violate FEE050 (not supported by all versions). One instance of the block is successfully written to the memory. Another write operation of this block is scheduled but it is interrupted before finish (by Fee_Cancel() or power down). Previous instance of the block is still accessible.

Immediate data usage

Immediate data are used for the fast write, because no swap operation can occur during write (when used properly). Following sequence shall be used for the immediate data usage:

1. Fee_EraseImmediateBlock() – to allocate space for the block in advance. In this case a swap operation can occur.
2. Fee_Cancel() – to interrupt any internal processing in case the driver is not idle.
3. Fee_Write() of immediate block – no swap can occur because space is already allocated.

After the immediate block is written, new space shall be allocated for the further usage (using Fee_EraseImmediateBlock() function). Note that after Fee_EraseImmediateBlock() function is called new space is allocated and previous instance of the immediate block is no longer accessible.

Example of usage 1:

- FEE/FLS init.
- Space reservation for immediate blocks (Fee_EraseImmediateBlock()).
- App execution.
- Abnormal situation occurs.
- All fee operations are stopped (Fee_Cancel()).
- Immediate blocks are written
- Power down/reset …
- FEE/FLS init.
- Check if immediate data are written. If yes, some abnormal situation occurred. Use stored information.
- Space reservation for immediate blocks – all previous data of these blocks are lost.

Example of usage 2:

- Space reservation is done using flash image.
- App is running and writing immediate data in case of some special situation (each immediate blocks are written just once).
- When all immediate blocks are written, unit is not writing any immediate data anymore and report error state.

Code flash erase

Be aware of the read-while-write support when erasing the code flash from code flash. If it is not supported the access code shall be loaded to the RAM ("Fls Load Access Code On Job Start" is turned on). In this case the write/erase operation shall be set as synchronous (if asynchronous write/erase is enabled, access code in the RAM is ignored).

Meaning of the FEE block states (result of the FEE Read operation)

Table 6-1. Fee Job Information

MemIf_JobResultType	Non-immediate block	Immediate block
MEMIF_JOB_OK	block is valid	block is valid
MEMIF_BLOCK_INVALID	block was invalidated, or block is not in the cluster	block was invalidated, or block is not in the cluster, or space for the block is reserved
MEMIF_BLOCK_INCONSISTENT	block has corrupted header (wrong checksum, address, ...), or block doesn’t match with configuration (length, immediate, ...)	block has corrupted header (wrong checksum, address, ...), or block doesn’t match with configuration (length, immediate, ...)

Chapter 7
Memory Allocation

7.1 Sections to be defined in MemMap.h

For precompile:

```
#ifdef FEE_START_CONFIG_DATA_UNSPECIFIED
#undef FEE_START_CONFIG_DATA_UNSPECIFIED
#undef MEMMAP_ERROR
/*no definition -> default compiler settings are used */
#endif
#ifdef FEE_STOP_CONFIG_DATA_UNSPECIFIED
#undef FEE_STOP_CONFIG_DATA_UNSPECIFIED
#undef MEMMAP_ERROR
/*no definition -> default compiler settings are used */
#endif
```

For Code:

```
#ifdef FEE_START_SEC_CODE
#undef FEE_START_SEC_CODE
#undef MEMMAP_ERROR
/*no definition -> default compiler settings are used */
#endif
#ifdef FEE_STOP_SEC_CODE
#undef FEE_STOP_SEC_CODE
#undef MEMMAP_ERROR
/*no definition -> default compiler settings are used */
#endif
```

For Variables:

```
#ifdef FEE_START_SEC_VAR_UNSPECIFIED
#undef FEE_START_SEC_VAR_UNSPECIFIED
#undef MEMMAP_ERROR
/*no definition -> default compiler settings are used */
#endif
#ifdef FEE_STOP_SEC_VAR_UNSPECIFIED
#undef FEE_STOP_SEC_VAR_UNSPECIFIED
#undef MEMMAP_ERROR
/*no definition -> default compiler settings are used */
```

```
#endif
```

For Constant data:

```
#ifdef FEE_START_SEC_CONST_UNSPECIFIED
#undef FEE_START_SEC_CONST_UNSPECIFIED
#undef MEMMAP_ERROR
/*no definition -> default compiler settings are used */
#endif
#ifdef FEE_STOP_SEC_CONST_UNSPECIFIED
#undef FEE_STOP_SEC_CONST_UNSPECIFIED
#undef MEMMAP_ERROR
/*no definition -> default compiler settings are used */
#endif
```

7.2 Linker command file

Memory shall be allocated for every section defined in MemMap.h.

Chapter 8
Configuration parameters considerations

Configuration parameter class for Autosar FEE driver fall into the following variants as defined below:

8.1 Configuration Parameters

Configuration parameter class for Autosar Fee driver fall into the following variants as defined below:

Table 8-1. Configuration Parameters

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
FeeGeneral			
	FeeDevErrorDetect	VariantPreCompile	PreCompile
	FeeIndex	VariantPreCompile	PreCompile
	FeeNvmJobEndNotification	VariantPreCompile	PreCompile
	FeeNvmJobErrorNotification	VariantPreCompile	PreCompile
	FeePollingMode	VariantPreCompile	PreCompile
	FeeVersionInfoApi	VariantPreCompile	PreCompile
	FeeVirtualPageSize	VariantPreCompile	PreCompile
	FeeDataBufferSize	VariantPreCompile	PreCompile
	FeeBlockAlwaysAvailable	VariantPreCompile	PreCompile
FeeSector			
	FeeSectorRef	VariantPreCompile	PreCompile
FeeBlockConfiguration			
	FeeClusterGroupRef	VariantPreCompile	PreCompile
	FeeBlockNumber	VariantPreCompile	PreCompile
	FeeBlockSize	VariantPreCompile	PreCompile
	FeeImmediateData	VariantPreCompile	PreCompile

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	FeeNumberOfWriteCycles	VariantPreCompile	PreCompile
	FeeDeviceIndex	VariantPreCompile	PreCompile

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating FEE Driver :

- Generate the required FEE configurations. For more details refer to Setting up the Plug-ins
- Allocate proper memory sections in MemMap.h and linker command file. For more details refer to section Sections to be defined in MemMap.h
- Make sure all include files for compilation. For more details refer to section Files required for Compilation
- Map the ISRs to their vector locations. For more details refer to section ISR to configure within OS – dependencies
- Compile and build the FEE with all the dependent modules. For more details refer to section Building the Driver

Chapter 10
ISR Reference

None.

10.1 Software specification

The following sections contains driver software specifications.

10.1.1 Define Reference

Constants supported by the driver are as per AUTOSAR FEE Driver software specification Version 3.0 .

10.1.2 Enum Reference

Enumeration of all constants supported by the driver are as per AUTOSAR FEE Driver software specification Version 3.0 .

10.1.3 Function Reference

Functions of all functions supported by the driver are as per AUTOSAR FEE Driver software specification Version 3.0 .

10.1.4 Structs Reference

Data structures supported by the driver are as per AUTOSAR FEE Driver software specification Version 3.0 .

10.1.5 Types Reference

Types supported by the driver are as per AUTOSAR FEE Driver software specification Version 3.0 .

10.1.6 Variables Reference

Variables supported by the driver are as per AUTOSAR FEE Driver software specification Version 3.0 .

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: IM14FEEASR3.0R2.0.0
Rev. 2.6