Integration Manual

for S32K14X Mcem Driver

Document Number: IM2MCEMASR4.2 Rev0002R1.0.2
Rev. 1.0

Contents

Section number **Title** **Page**

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	26/04/2019	NXP MCAL Team	Updated version for ASR 4.2.2S32K14XR1.0.2

Chapter 2
Introduction

This integration manual describes the integration requirements for MCEM Driver for S32K14X microcontrollers.

2.1 Supported Derivatives

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

Table 2-1. S32K14X Derivatives

NXP Semiconductors	s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64, s32k142_lqfp48, s32k144_lqfp48, s32k148_lqfp100

All of the above microcontroller devices are collectively named as S32K14X .

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
ASM	Assembler
BSMI	Basic Software Make file Interface
DEM	Diagnostic Event Manager
DET	Development Error Tracer
C/CPP	C and C++ Source Code
VLE	Variable Length Encoding
N/A	Not Applicable
MCEM	Microcontroller Error Manager
MCU	Micro-Controller Unit
ISR	Interrupt Service Routine
NMI	Non-maskable Interrupt

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
EIRM	Error Injection and Reporting Module
NCF	Non-Critical Fault
EIM	Error Injection Module
ERM	Error Reporting Module

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	Specification of MCEM Driver	AUTOSAR Release 4.2.2
2	S32K14X Reference Manual	Reference Manual, Rev. 9, 9/2018
3	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	30/11/2017
4	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	30/11/2017
5	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	30/11/2017
6	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	25/10/2018
7	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	07/01/2019

Chapter 3
Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar MCEM driver for NXP SemiconductorsS32K14X . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The MCEM driver files are compiled using

- Green Hills Multi 7.1.4 / Compiler 2017.1.4
- (Linaro GCC 6.3-2017.06~dev) 6.3.1 20170509 (Wed Jan 24 16:21:45 CST 2018 build.sh rev=g27a1317 s=L631 Earmv7 -V release_g27a1317_build_Fed_Earmv7)
- IAR: V8.11.2

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

Note

The TS_T40D2M10I2R0 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 = 40 identifies CORTEXM architecture and Derivative_Id = 2 identifies the S32K14X)

3.1.1 GHS Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-ansi	Specifies ANSI C with extensions. This mode extends the ANSI X3.159-1989 standard with certain useful and compatible constructs.
-Osize	Optimize for size.
-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
-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
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
--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
--short_enum	Store enumerations in the smallest possible type
-c	Produces an object file (called input-file.o) for each source file.
--no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup.
-keeptempfiles	Prevents the deletion of temporary files after they are used. If an assembly language file is created by the compiler, this option will place it in the current directory instead of the temporary directory. Produces an object file (called input-file.o) for each source file.
-list	Creates a listing by using the name of the object file with the .lst extension. Assembler option
-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
-DDISABLE_MCAL_INTERMODULE_ASR_CHECK	-D defines a preprocessor symbol to disable the inter-module version check for AR_RELEASE versions. DISABLE_MCAL_INTERMODULE_ASR_CHECK: By default in the package, drivers are compiled to perform the inter-module version check as per Autosar BSW004. When the inter-module version check needs to be disabled then the DISABLE_MCAL_INTERMODULE_ASR_CHECK global define must be added to the list of compiler options.
-DGHS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GHS preprocessor symbol.

Table 3-2. Assembler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-c	Produces an object file (called input-file.o) for each source file.
-preprocess_assembly_files	Preprocesses assembly files
-asm=list	Creates a listing by using the name of the object file with the .lst extension. Assembler option

Table 3-3. Linker Options

Option	Description
-Mn	Map file numeric ordering
-delete	Removal from the executable of functions that are unused and unreferenced
-v	Display removed unused functions
-ignore_debug_references	Ignores relocations from DWARF debug sections when using -delete.
-map	Creates a detailed map file
-keepmap	Keep the map file in the event of a link error
-lstartup	Link libstartup library -Run-time environment startup routines
-lsys	Link libsys library -Run-time environment system routines
-larch	Link libarch library -Target-specific run-time support. Any file produced by the Green Hills Compiler may depend on symbols in this library.
-lansi	Link libansi library -the standard C library
-L(/lib/thumb2)	Link thumb2 library
-lutf8_s32	Include utf8_s32.a to use the Wide Character Functions

3.1.2 IAR Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--cpu_mode=thumb	Selects generating code that executes in Thumb state.
--endian=little	Specifies the endianess of core: little endian.
-Ohz	Sets the optimization level to High, favoring size.
-c	Produces an object file (called input-file.o) for each source file.
--no_clustering	Disables static clustering optimizations.
--no_mem_idioms	Makes the compiler to not optimize code sequences that clear, set, or copy a memory region.
--no_explicit_zero_opt	Places the zero initialized variables in data section instead of bss.
--debug	Makes the compiler include information in the object modules.

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
--diag_suppress=Pa050	Suppresses diagnostic messages (warnings) about non-standard line endings.
-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
-DIAR	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the IAR preprocessor symbol.
--require_prototypes	Forces the compiler to verify that all functions have proper prototypes.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by compiler.
--no_system_include	Disables the automatic search for system include files.
-e	Enables language extensions. This option is needed by FLS driver which uses _packed structures.

Table 3-5. Assembler Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--cpu_mode=thumb	Selects generating code that executes in Thumb state.
-g	Use this option to disable the automatic search for system include files.

Table 3-6. Linker Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--map filename	Produces a map file.
--no_library_search	Disables automatic runtime library search.
--entry _start	Treats the symbol _start as a root symbol and as the start of the application.
--enable_stack_usage	Enables stack usage analysis.
--skip_dynamic_initialization	Suppress dynamic initialization during system startup.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by linker.
--config	Specifies the configuration file to be used by the linker.

3.1.3 GCC Compiler/Linker/Assembler Options

Table 3-7. Compiler Options

Option	Description
-c	Produces an object file (called input-file.o) for each source file.
-Os	Use optimization for size.
-ggdb3	Produce debugging information for use by GDB. Level 3 includes extra information, such as all the macro definitions present in the program.
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
-ansi	Specifies ANSI C with extensions.
-mlittle-endian	Generate code for a processor running in little-endian mode.
-fomit-frame-pointer	Removes the frame pointer for all functions, which might make debugging harder.
-msoft-float	Use software floating-point instructions.
-fno-common	Specifies that the compiler should place uninitialized global variables in the data section of the object file, rather than generating them as common blocks.
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
-Wextra	Enables some extra warning flags that are not enabled by '-Wall'.
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types.
-Wno-sign-compare	Do not warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned.
-fstack-usage	Geneates an extra file that specifies the maximum amount of stack used, on a per-function basis.
-fdump-ipa-all	Enables all inter-procedural analysis dumps.
-Werror=implicit-function-declaration	Generates an error when the prototype of the function is not defined..
-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
-DGCC	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GCC preprocessor symbol.

Table 3-8. Assembler Options

Option	Description
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-c	Produces an object file (called input-file.o) for each source file.
-mthumb	This option specifies that the assembler should start assembling Thumb instructions.
-x assembler-with-cpp	Indicates that the assembly code contains C directives and the C preprocessor must be run.

Table 3-9. Linker Options

Option	Description
-Map=filename	Print a link map to the file mapfile.
-T scriptfile	Use scriptfile as the linker script. This script replaces ld's default linker script(rather than adding to it), so commandfile must specify everything necessary to describe the output file.

3.2 Files Required for Compilation

This section describes the include files required to compile, assemble (if assembler code is used), and link the MCEM driver for S32K14X 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.

MCEM Header Files

- Mcem_TS_T40D2M10I2R0\include\CDD_Mcem.h
- Mcem_TS_T40D2M10I2R0\include\CDD_Mcem_Types.h
- Mcem_TS_T40D2M10I2R0\include\CDD_Mcem_IrqHandler.h
- Mcem_TS_T40D2M10I2R0\include\Mcem_IPW.h
- Mcem_TS_T40D2M10I2R0\include\Mcem_Eirm.h
- Mcem_TS_T40D2M10I2R0\include\Mcem_Eirm_Types.h
- Mcem_TS_T40D2M10I2R0\include\Reg_eSys_Eirm.h

MCEM Source Files

- Mcem_TS_T40D2M10I2R0\src\CDD_Mcem.c
- Mcem_TS_T40D2M10I2R0\src\Mcem_IPW.c
- Mcem_TS_T40D2M10I2R0\src\Mcem_Eirm.c
- Mcem_TS_T40D2M10I2R0\src\Mcem_Eirm_Irq.c

MCEM Generated Files for parameters without variation points

- CDD_Mcem_Cfg.h - MCEM Pre-compile configuration file. This file should be generated by the user using a configuration tool.
- CDD_Mcem_Cfg.c- MCEM Pre-compile configuration file. This file should be generated by the user using a configuration tool.

MCEM Generated Files for variant aware parameters

- CDD_Mcem_<VariantName>_PBcfg.c - MCEM Post-build configuration file. This file should be generated by the user using a configuration tool.The file contains the definition of the init pointer for the respective variant.

As a deviation from standard:

- CDD_Mcem_<VariantName>_PBcfg.c files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB)
- CDD_Mcem_Cfg.c file will contain the definition for all configuration structures containing only variables that are not variant aware, configured and generated only once. This file alone does not contain the whole structure needed by MCEM_Init function to configure the driver. Based on the number of variants configured in the EcuC, there can be more than one configuration structure for one module even for PreCompile variant.

Files from Base common folder

- Base_TS_T40D2M10I2R0\include\Compiler.h
- Base_TS_T40D2M10I2R0\include\Compiler_Cfg.h
- Base_TS_T40D2M10I2R0\include\ComStack_Types.h
- Base_TS_T40D2M10I2R0\include\Mcem_MemMap.h
- Base_TS_T40D2M10I2R0\include\Mcal.h
- Base_TS_T40D2M10I2R0\include\Platform_Types.h
- Base_TS_T40D2M10I2R0\include\Std_Types.h
- Base_TS_T40D2M10I2R0\include\Reg_eSys.h
- Base_TS_T40D2M10I2R0\include\Soc_Ips.h
- Base_TS_T40D2M10I2R0\include\Reg_Macros.h

Files from Det folder:

- Det_TS_T40D2M10I2R0\include\Det.h
- Det_TS_T40D2M10I2R0\src\Det.c

3.3 Setting up the Plug-Ins

The MCEM driver was designed to be configured by using the EB Tresos Studio (version EB tresos Studio 23.0.0 b170330-0431 or later).

Location of various files inside the MCEM module folder:

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\Mcem_TS_T40D2M10I2R0\config\Mcem.xdm
  - ..\Resource_TS_T40D2M10I2R0\config\Resource.xdm
  - ..\Base_TS_T40D2M10I2R0\config\Base.xdm
  - ..\Det_TS_T40D2M10I2R0\config\Det.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\Mcem_TS_T40D2M10I2R0\autosar\Mcem*.epd

    - ..\Resource_ TS_T40D2M10I2R0\autosar\Resource.epd
    - ..\Base_ TS_T40D2M10I2R0\autosar\Base.epd
    - ..\Det_ TS_T40D2M10I2R0\autosar\Det.epd

- Code Generation Templates for parameters without variation points:
    - ..\Mcem_TS_T40D2M10I2R0\generate_PC\src\CDD_Mcem_Cfg.c
    - ..\Mcem_TS_T40D2M10I2R0\generate_PC\include\CDD_Mcem_Cfg.h

- Code Generation Templates for variant aware parameters:
    - ..\Mcem_TS_T40D2M10I2R0\generate_PB\src \CDD_Mcem_PBcfg_[VariantName].c
    - ..\Mcem_TS_T40D2M10I2R0\generate_PB\include \CDD_Mcem_PBcfg_[VariantName].h

Steps to generate the configuration:

1. Copy the module folders Mcem_TS_T40D2M10I2R0, Det_ TS_T40D2M10I2R0, Base_ TS_T40D2M10I2R0, Resource_ TS_T40D2M10I2R0, Ecuc_ TS_T40D2M10I2R0 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.

Dependencies

- **RESOURCE** is required to select processor derivative. The current MCEM driver has support for the following derivatives, which have their respective resource files: s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64, s32k142_lqfp48, s32k144_lqfp48, s32k148_lqfp100.
- **DET** is required for signaling the development error detection (parameters out of range, null pointers, etc).
- **ECUC** is required for configuring the variant handling in Tresos.
- **BASE** is a basic module which represents a foundation of all other drivers.

Chapter 4
Function calls to module

4.1 Function Calls During Start-Up

The MCEM module shall be initialized during STARTUP phase of the EcuM initialization. The API to be called is

```
Mcem_Init( ConfigPtr )
```

4.2 Function Calls During Shutdown

N/A

4.3 Function Calls During Wake-up

N/A

Chapter 5
Module requirements

5.1 Exclusive Areas to Be Defined in the BSW Scheduler

The MCEM driver does not use any EXCLUSIVE area.

5.2 Peripheral Hardware Requirements

The EIM and ERM IPs of the S32K14X are used by the MCEM driver.

5.3 ISR to Configure within the OS – Dependencies

The following ISR are used by the MCEM driver:

The ISR table is presented below. Depending on the derivative used, some of the ISRs may not be available. For complete details please consult the Reference Manual:

Table 5-1. EIRM Interrupt Service Routines

EIRM Interrupt Service Routine	NVIC ID
MCEM_EIRM_FAULT_SINGLEBIT_ISR	44
MCEM_EIRM_FAULT_DOUBLEBIT_ISR	45

5.4 ISR Macro

MCAL drivers use the ISR macro to define the functions that will process hardware interrupts. Depending on whether the OS is used or not, this macro can have different definitions:

1. OS is not used - AUTOSAR_OS_NOT_USED is defined:
   - If USE_SW_VECTOR_MODE is defined:

     ```
     #define ISR(IsrName) void IsrName(void)
     ```

     In this case, drivers interrupt handlers are normal C functions and the prolog/ epilog handle the context save and restore.

   - If USE_SW_VECTOR_MODE is not defined:

     ```
     #define ISR(IsrName) INTERRUPT_FUNC void IsrName(void)
     ```

     In this case, drivers interrupt handlers must save and restore the execution context.

2. NXP Semiconductors OS is used – AUTOSAR_OS_NOT_USED is not defined

   ```
   #define ISR(IsrName) void OS_isr_##IsrName()
   ```

   In this case, OS is handling the execution context when an interrupt occurs. Drivers' interrupt handlers are normal C functions.
3. Other vendor's OS is used – AUTOSAR_OS_NOT_USED is not defined. Please refer to the OS documentation for description of the ISR macro.

5.5 Other AUTOSAR Modules - Dependencies

- **BASE:** The BASE module contains the common files/definitions needed by all MCAL modules.
- **DET:** The DET module is used for enabling Development error detection. The API function used is Det_ReportError(). The activation / deactivation of Development error detection is configurable using the 'McemDevErrorDetect' configuration parameter.
- **ECUC:** The ECUC module is used for ECU configuration. MCAL modules need ECUC to retrieve the variant information.
- **MCU:** The MCU driver provides services for basic microcontroller initialization, power down functionality, reset and microcontroller specific functions required by other MCAL software modules. The clocks need to be initialized prior to using the MCEM driver.
- **RESOURCE:** Resource module is used to select microcontroller's derivatives.

5.6 User Mode Support

The MCEM can be run in user mode if the following steps are performed:

- Enable McemEnableUserModeSupport from the configuration
- Call the following functions as trusted functions:
  - Mcem_Eirm_Init for initializing driver.
  - Mcem_Eirm_InjectFault for injecting a fault to driver.
  - Mcem_Eirm_GetErrors for geting error status.
  - Mcem_Eirm_ClearFault for clearing specific fault Id.

Chapter 6
Main API Requirements

6.1 Main Function Calls within the BSW Scheduler

None.

6.2 API Requirements

None.

6.3 Calls to Notification Functions, Callbacks, Callouts

Call-back Notifications:

None.

User Notification:

The MCEM Driver provides a notification for the ALARM event. The ISRs are responsible for resetting the interrupt flags (if needed by hardware) and calling the corresponding upper layer that will invoke notification functions. The notifications can be configured as pointers to user defined functions. If notification is not desired, `NULL_PTR` shall be configured.

If there is at least one fault configured with ALARM interrupt enabled, it must be handled by user provided notification routine (then the notification API must be enabled and the notification pointer properly configured).

The prototype of the notification function is as follows:

```
void NotificationName( uint8 nFaultId , uint32 u32ErrorAddr);
```

Chapter 7
Memory Allocation

7.1 Sections to be defined in Mcem_MemMap.h

Table 7-1. MemMap sections present in the MCEM driver code

Section name	Section type	Description
MCEM_START_SEC_CODE	Code	Start of Memory Section for Code
MCEM_STOP_SEC_CODE	Code	End of Memory Section for Code
MCEM_START_SEC_CONST_UNSPECIFIED	Configuration Data/Const	Start of Memory Section for Const with undefined sized
MCEM_STOP_SEC_CONST_UNSPECIFIED	Configuration Data/Const	End of Memory Section for Const Data
MCEM_START_SEC_VAR_INIT_UNSPECIFIED	Variables	Used for variables, structures, arrays, when the SIZE (alignment) does not fit the criteria of 8,16 or 32 bit. These variables are initialized with values after every reset.
MCEM_STOP_SEC_VAR_INIT_UNSPECIFIED	Variables	End of above section.
MCEM_START_SEC_VAR_INIT_BOOLEAN	Variables	Used for variables of type boolean. These variables are initialized with values after every reset.
MCEM_STOP_SEC_VAR_INIT_BOOLEAN	Variables	End of above section.
MCEM_START_SEC_VAR_NO_INIT_32	Variables	Used for variables of type uint32. These variables are not initialized with values after every reset.
MCEM_STOP_SEC_VAR_NO_INIT_32	Variables	End of above section.

7.2 Linker command file

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

Chapter 8
Configuration parameters considerations

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

8.1 Configuration Parameters

Configuration parameter class for the MCEM driver fall into the following variants:

Table 8-1. Configuration Parameters

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
Mcem	IMPLEMENTATION_CONFIG_VARIANT		
McemConfigSet	FaultTimeout	VariantPC or VariantPB	Post Build
	EOUT_FaultOutputMode	VariantPC or VariantPB	Post Build
	McemHardlockedConfiguration	VariantPC or VariantPB	Post Build
	McemSoftLockedConfiguration	VariantPC or VariantPB	Post Build
	EOUT_SwitchingMode	VariantPC or VariantPB	Post Build
	EOUT_PolaritySelection	VariantPC or VariantPB	Post Build
	McemErrorNotificationWithAddressAPI	VariantPC or VariantPB	Post Build
	McemTcmLowerAddr	VariantPC or VariantPB	Post Build
	McemTcmUpperAddr	VariantPC or VariantPB	Post Build
	McemInjectBit1	VariantPC or VariantPB	Post Build
	McemInjectBit2	VariantPC or VariantPB	Post Build
McemConfigSet/ DefaultFaultConfig	FaultIrqEnabled	VariantPC or VariantPB	Post Build
McemConfigSet/Fault	SignalDesc	VariantPC or VariantPB	Post Build
	FaultDisabled	VariantPC or VariantPB	Post Build
	FaultIrqEnabled	VariantPC or VariantPB	Post Build

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
McemGeneral	McemDevErrorDetect	Pre Compile parameter for all Variants of Configuration	Pre Compile
	McemVersionInfoApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	McemAlarmNotificationApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	McemEnableUserModeSupport	Pre Compile parameter for all Variants of Configuration	Pre Compile
	McemEnableIsrSingleErrorBit	Pre Compile parameter for all Variants of Configuration	Pre Compile
	McemEnableIsrDoubleErrorBit	Pre Compile parameter for all Variants of Configuration	Pre Compile
CommonPublishedInformation	ArReleaseMajorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	ArReleaseMinorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	ArReleaseRevisionVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	ModuleId	Pre Compile parameter for all Variants of Configuration	Pre Compile
	SwMajorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	SwMinorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	SwPatchVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	VendorApiInfix	Pre Compile parameter for all Variants of Configuration	Pre Compile
	VendorId	Pre Compile parameter for all Variants of Configuration	Pre Compile

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating Microcontroller Error Manager :

- Generate the required MCEM configurations. For more details refer to section Files Required for Compilation
- Allocate proper memory sections in MCEM_MemMap.h and linker command file. For more details refer to section
- Compile & build the MCEM with all the dependent modules. For more details refer to section Building the Driver

9.1 Fault Names Reference

The following table presents fault symbolic names and their mapping to the IDs of the S32K14X devices:

Table 9-1. EIRM Fault Names

Fault Name	ID	Description
MCEM_EIRM_MEN0_SINGLE_CORRECTION	0	Single error correctable in lower RAM TCM
MCEM_EIRM_MEN0_NON_CORRECTABLE	1	Multipler error noncorrectable in lower RAM TCM
MCEM_EIRM_MEN1_SINGLE_CORRECTION	2	Single error correctable in upper RAM TCM
MCEM_EIRM_MEN1_NON_CORRECTABLE	3	Multipler error noncorrectable in upper RAM TCM

Refer to the S32K14X Reference Manual in the Reference List for details regarding the faults.

The total number of MCEM faults is available through the MCEM_NCF_TOTAL_NUMBER constant.

Chapter 10
ISR Reference

ISR functions exported by the driver:

Table 10-1. MCEM EIRM Interrupts

Internal EIRM Interrupt Service Routines	NVIC ID
MCEM_EIRM_FAULT_SINGLEBIT_ISR	44
MCEM_EIRM_FAULT_DOUBLEBIT_ISR	45

The MCEM_EIRM_FAULT_SINGLEBIT_ISR interrupt will handle the error for single error bit correctable occurs in RAM Lower/Upper.

The MCEM_EIRM_FAULT_DOUBLEBIT_ISR interrupt will handle the error for more than 1 error bit non-correctable occurs in RAM Lower/Upper..

User can define ErrorNotification which will be called when an event with IRQ reaction occurs, to handling errors. After exit from ISR handler, the corresponding fault which notify to user by notification will be automatically cleared due to requirement CPR-MCAL-853.mcem to prevent spurious interrupt. Therefore, it is recomment to do reaction or save fault status to user variable.

Chapter 11
External Assumptions for MCEM driver

The section presents requirements that must be complied with when integrating MCEM driver into the application.

[SMCAL_CPR_EXT55]

<< Mcem_Init() function is non-reentrant. The external application shall ensure that this function is not preempting itself. >>

[SMCAL_CPR_EXT56]

<< Mcem_GetErrors() function is non-reentrant. The external application shall ensure that this function is not preempting itself.

>>

[SMCAL_CPR_EXT57]

<< Mcem_ClearFaults() function is non-reentrant. The external application shall ensure that this function is not preempting itself. >>

[SMCAL_CPR_EXT58]

<< Mcem_InjectFault() function is non-reentrant. The external application shall ensure that this function is not preempting itself.

>>

[SMCAL_CPR_EXT59]

<< The external application shall ensure that the following functions don't preempt one another:

-Mcem_Init

-Mcem_ClearFaults

-Mcem_GetErrors. >>

NOTE

The functions create conflicts on pending operations.

[SMCAL_CPR_EXT154]

<< The application shall be able to detect and react in the proper way for the situation of resets due to continuous FCCU reaction. >>

[SMCAL_CPR_EXT155]

<< MCEM fault injection API shall be used by the application for persistent faults detection on the reaction path.(The external application shall invoke the Mcem_InjectFault() function for ensuring that the reaction path between different IPs (MEMU, FCCU, MC_RGM, INTC) is working correctly.) >>

[SMCAL_CPR_EXT163]

<< If interrupts are locked a centralized function pair to lock and unlock interrupts shall be used. >>

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Document Number IM2MCEMASR4.2 Rev0002R1.0.2
Revision 1.0

AUTOSAR