

---

# User Manual

for S32K14X Mcem Driver

Document Number: UM2MCEMASR4.2 Rev0002R1.0.2  
Rev. 1.0





# Contents

Section number	Title	Page
<b>Chapter 1</b>		
<b>Revision History</b>		
<b>Chapter 2</b>		
<b>Introduction</b>		
2.1	Supported Derivatives.....	9
2.2	Overview.....	9
2.3	About this Manual.....	10
2.4	Acronyms and Definitions.....	10
2.5	Reference List.....	11
<b>Chapter 3</b>		
<b>Driver</b>		
3.1	Requirements.....	13
3.2	Driver Design Summary.....	13
3.3	Hardware Resources.....	13
3.4	Deviation from Requirements.....	13
3.5	Driver limitations.....	14
3.6	Driver usage and configuration tips.....	14
3.7	Runtime Errors.....	15
3.8	Software specification.....	15
3.8.1	Define Reference.....	15
3.8.1.1	Define MCEM_INIT_ID.....	15
3.8.1.2	Define MCEM_INJECT_FAULT_ID.....	16
3.8.1.3	Define MCEM_GET_ERRORS_ID.....	16
3.8.1.4	Define MCEM_CLEAR_FAULTS_ID.....	16
3.8.1.5	Define MCEM_GETVERSIONINFO_ID.....	16
3.8.1.6	Define MCEM_ALARMROUTINE_ID.....	16
3.8.1.7	Define MCEM_E_UNINIT.....	16
3.8.1.8	Define MCEM_E_PARAM_CONFIG.....	16

Section number	Title	Page
3.8.1.9	Define MCEM_E_PARAM_VINFO.....	17
3.8.1.10	Define MCEM_E_PARAM_FAULT.....	17
3.8.1.11	Define MCEM_E_CFG_LOCK.....	17
3.8.1.12	Define MCEM_E_PREEMPTION.....	17
3.8.1.13	Define MCEM_E_FORBIDDEN_INVOCATION.....	17
3.8.1.14	Define MCEM_FAULT_PENDING.....	17
3.8.1.15	Define MCEM_GETERRORS_STATUS.....	18
3.8.1.16	Define MCEM_DEV_ERROR_DETECT.....	18
3.8.1.17	Define MCEM_GET_VERSION_INFO_API.....	18
3.8.1.18	Define MCEM_ALARM_NOTIFICATION_API.....	18
3.8.2	Enum Reference.....	18
3.8.2.1	Enumeration Mcem_FunctionStateType.....	18
3.8.2.2	Enumeration Mcem_StateType.....	19
3.8.3	Function Reference.....	19
3.8.3.1	Function Mcem_Init.....	19
3.8.3.2	Function Mcem_CheckNMI.....	20
3.8.3.3	Function Mcem_GetErrors.....	20
3.8.3.4	Function Mcem_ClearFaults.....	21
3.8.3.5	Function Mcem_InjectFaults.....	21
3.8.3.6	Function Mcem_GetVersionInfo.....	22
3.8.4	Structs Reference.....	22
3.8.4.1	Structure Mcem_ConfigType.....	22
3.8.4.2	Structure Mcem_FaultContainerType.....	23
3.8.5	Types Reference.....	23
3.8.5.1	Typedef Mcem_FaultType.....	23
3.8.5.2	Typedef Mcem_ErrorNotifyType.....	23
3.8.6	Variables Reference.....	24
3.8.6.1	Variable Mcem_DriverState.....	24
3.8.6.2	Variable Mcem_FunctionState.....	24

Section number	Title	Page
3.8.6.3	Variable Mcem_pConfigPtr.....	24
3.8.7	Fault Management.....	24
3.8.7.1	Fault Status Manipulation Macros.....	25
3.8.7.1.1	Define MCEM_FAULT_PENDING.....	25
3.8.7.1.2	Define MCEM_GETERRORS_STATUS.....	25
3.8.7.2	Fault Names Reference.....	25
3.9	Symbolic Names DISCLAIMER.....	26

## Chapter 4

### Tresos Configuration Plug-in

4.1	Configuration elements of MCEM.....	27
4.2	Form IMPLEMENTATION_CONFIG_VARIANT.....	27
4.3	Form McemGeneral.....	27
4.3.1	McemDevErrorDetect (McemGeneral).....	28
4.3.2	McemVersionInfoApi (McemGeneral).....	28
4.3.3	McemAlarmNotificationApi (McemGeneral).....	29
4.3.4	McemEnableUserModeSupport (McemGeneral).....	29
4.3.5	McemEnableIsrSingleErrorBit (McemGeneral).....	30
4.3.6	McemEnableIsrDoubleErrorBit (McemGeneral).....	30
4.4	Form McemConfigSet.....	30
4.4.1	FaultTimeout (McemConfigSet).....	31
4.4.2	EOUT_FaultOutputMode (McemConfigSet).....	31
4.4.3	McemHardlockedConfiguration (McemConfigSet).....	32
4.4.4	McemSoftLockedConfiguration (McemConfigSet).....	32
4.4.5	EOUT_SwitchingMode (McemConfigSet).....	32
4.4.6	EOUT_PolaritySelection (McemConfigSet).....	33
4.4.7	McemErrorNotificationWithAddressAPI (McemConfigSet).....	33
4.4.8	McemTcmLowerAddr (McemConfigSet).....	33
4.4.9	McemTcmUpperAddr (McemConfigSet).....	34
4.4.10	McemInjectBit1 (McemConfigSet).....	34

Section number	Title	Page
4.4.11	McemInjectBit2 (McemConfigSet).....	36
4.4.12	DefaultFaultConfig Collection (McemConfigSet).....	38
4.4.12.1	FaultIrqEnabled (DefaultConfig).....	38
4.4.13	Form Fault.....	39
4.4.13.1	SignalDesc (Fault).....	39
4.4.13.2	FaultDisabled (Fault).....	40
4.4.13.3	FaultIrqEnabled (DefaultConfig).....	40
4.5	Form CommonPublishedInformation.....	41
4.5.1	ArReleaseMajorVersion (CommonPublishedInformation).....	41
4.5.2	ArReleaseMinorVersion (CommonPublishedInformation).....	42
4.5.3	ArReleaseRevisionVersion (CommonPublishedInformation).....	42
4.5.4	ModuleId (CommonPublishedInformation).....	42
4.5.5	SwMajorVersion (CommonPublishedInformation).....	43
4.5.6	SwMinorVersion (CommonPublishedInformation).....	43
4.5.7	SwPatchVersion (CommonPublishedInformation).....	44
4.5.8	VendorApiInfix (CommonPublishedInformation).....	44
4.5.9	VendorId (CommonPublishedInformation).....	45

# 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 User Manual describes NXP Semiconductors AUTOSAR Microcontroller Error Manager ( MCEM ) for S32K14X .

AUTOSAR MCEM driver configuration parameters and deviations from the specification are described in MCEM Driver chapter of this document. AUTOSAR MCEM driver requirements and APIs are described in the AUTOSAR MCEM driver software specification document.

## 2.1 Supported Derivatives

The software described in this document is intended 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

*Table continues on the next page...*

**Table 2-2. Acronyms and Definitions (continued)**

Term	Definition
MCEM	Microcontroller Error Manager
MCU	Micro-Controller Unit
ISR	Interrupt Service Routine
NMI	Non-maskable Interrupt
FCCU	Fault Collection and Control Unit
NCF	Non-Critical Fault
MEMU	Memory Error Management Unit

## 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 Driver

### 3.1 Requirements

Requirements for this driver are detailed in the AUTOSAR MCAL MCEM Traceability Matrix document.

### 3.2 Driver Design Summary

The MCEM driver configures how to react to the faults in the system.

The MCEM driver has the following major features:

- Enable interrupt service routine for each fault.
- Set the User Callback for Error Address.

### 3.3 Hardware Resources

MCEM does not strickly depend on Resources. There are only 2 output signal which already fixed in the hardware. It is not required to configure these pins from PORT module.

### 3.4 Deviation from Requirements

The Mcem Driver limitations, deviations from the Autosar specification and additional requirements to the upper software layer(s) are described in the following text.

The driver deviates from the Autosar Mcem Driver software specification in some places. The table [Table 3-2](#) identifies the Autosar requirements that are not fully implemented, implemented differently, or out of scope for the Mcem Driver. The table [Table 3-1](#) provides the Status column description.

**Table 3-1. Deviations Status Column Description**

Term	Definition
N/A	Not Available
N/T	Not Testable
N/S	Out of Scope
N/F	Not Fully Implemented
N/I	Not Implemented

**Table 3-2. MCEM Deviations**

Req.	Status	Description	Notes

## 3.5 Driver limitations

### None

With S32K11X, The Error Injection Module only supports for EIM channel 0 and the ECC Protection only supports for the SRAM\_U. So Mcem\_InjectFaults() function only inject faults for channel 0 and test the reaction for block SRAM\_U.

## 3.6 Driver usage and configuration tips

Here are some tips for configure MCEM module:

- To enable IRQ for ECC or non-ECC error, it is required to enable McemEnableIsrSingleErrorBit or McemEnableIsrDoubleErrorBit. It is recommended to define User notification in McemErrorNotificationWithAddressAPI.
- For quick configuration, all faults with similar configuration could be done by DefaultFaultConfig
- It is possible to select the address (McemTcmLowerAddr, McemTcmUpperAddr) that MCEM will read for injecting RAM error.

- To select bit for inject fault, it is possible to select different value for McemInjectBit1 and McemInjectBit2. However, it is better to select all to DATA bit or CHECKBIT because there is possibility that 1 DATA bit and 1 CHECKBIT do not work properly for injecting non-ECC case.
- After exit from ISR, the fault which enable IRQ processing will be cleared. So recording or doing reaction for these faults should be done within User Notification.

## 3.7 Runtime Errors

The driver generates the following DEM errors at runtime.

**Table 3-3. Runtime Errors**

Error Code	Condition triggering the error
MCEM_E_FORBIDDEN_INVOCATION	Function called when MCEM driver is in wrong state, a function preempts itself when not allowed, or the alarm ISR is invoked with invalid value in Frozen Status registers.
MCEM_E_PARAM_CONFIG	The initialization configuration is not correct.
MCEM_E_PARAM_FAULT	A function is called with an invalid parameter.
MCEM_E_INIT_FAILED	Issued when the initialization function fails.
MCEM_E_OPERATION_FAILED	Issued when a read or clear fault fails.
MCEM_E_CORRUPTED_HW_CONFIG	The HW Configuration periodic check failed.
MCEM_E_ALARM_ISR	Reference to the DemEventParameter which shall be issued through SERR when there is no alarm interrupt notification callback configured.

## 3.8 Software specification

The following sections contains driver software specifications.

### 3.8.1 Define Reference

Constants supported by the driver are as per AUTOSAR MCEM Driver software specification Version 4.2 Rev0002 .

#### 3.8.1.1 Define MCEM\_INIT\_ID

API service ID for Mcem\_Init function.

**Definition:** `#define MCEM_INIT_ID`

### 3.8.1.2 Define MCEM\_INJECT\_FAULT\_ID

API service ID for Mcem\_InjectFaults function.

**Definition:** `#define MCEM_INJECT_FAULT_ID`

### 3.8.1.3 Define MCEM\_GET\_ERRORS\_ID

API service ID for Mcem\_GetErrors function.

**Definition:** `#define MCEM_GET_ERRORS_ID`

### 3.8.1.4 Define MCEM\_CLEAR\_FAULTS\_ID

API service ID for Mcem\_ClearFaults function.

**Definition:** `#define MCEM_CLEAR_FAULTS_ID`

### 3.8.1.5 Define MCEM\_GETVERSIONINFO\_ID

API service ID for Mcem\_GetVersionInfo function.

**Definition:** `#define MCEM_GETVERSIONINFO_ID`

### 3.8.1.6 Define MCEM\_ALARMROUTINE\_ID

API service ID for Mcem\_AlarmRoutine function.

**Definition:** `#define MCEM_ALARMROUTINE_ID`

### 3.8.1.7 Define MCEM\_E\_UNINIT

API service is called while driver not initialized.

**Definition:** `#define MCEM_E_UNINIT`



### 3.8.1.8 Define MCEM\_E\_PARAM\_CONFIG

API service Mcem\_Init is called with wrong parameter.

**Definition:** `#define MCEM_E_PARAM_CONFIG`

### 3.8.1.9 Define MCEM\_E\_PARAM\_VINFO

API service Mcem\_GetVersionInfo is called with wrong parameter.

**Definition:** `#define MCEM_E_PARAM_VINFO`

### 3.8.1.10 Define MCEM\_E\_PARAM\_FAULT

API service is called with wrong parameter.

**Definition:** `#define MCEM_E_PARAM_FAULT`

### 3.8.1.11 Define MCEM\_E\_CFG\_LOCK

API service is called while driver is locked.

**Definition:** `#define MCEM_E_CFG_LOCK`

### 3.8.1.12 Define MCEM\_E\_PREEMPTION

API service is called while same non-reentrant service is running.

**Definition:** `#define MCEM_E_PREEMPTION`

### 3.8.1.13 Define MCEM\_E\_FORBIDDEN\_INVOCATION

API service is called while driver does not permit the service to run.

**Definition:** `#define MCEM_E_FORBIDDEN_INVOCATION`

### 3.8.1.14 Define MCEM\_FAULT\_PENDING

Access macro for querying collected fault status.

**Definition:** `#define MCEM_FAULT_PENDING( pError, nFault )`

### 3.8.1.15 Define MCEM\_GETERRORS\_STATUS

Access macro for stored Mcem\_Geterrors() return status.

**Definition:** `#define MCEM_GETERRORS_STATUS( pError )`

### 3.8.1.16 Define MCEM\_DEV\_ERROR\_DETECT

Enable/Disable Development Error Detection.

**Definition:** `#define MCEM_DEV_ERROR_DETECT`

### 3.8.1.17 Define MCEM\_GET\_VERSION\_INFO\_API

Function Mcem\_GetVersionInfo() enable switch.

**Definition:** `#define MCEM_GET_VERSION_INFO_API`

### 3.8.1.18 Define MCEM\_ALARM\_NOTIFICATION\_API

Function Mcem\_AlarmNotification enable switch.

**Definition:** `#define MCEM_ALARM_NOTIFICATION_API`

## 3.8.2 Enum Reference

Enumeration of all constants supported by the driver are as per AUTOSAR MCEM Driver software specification Version 4.2 Rev0002 .

### 3.8.2.1 Enumeration Mcem\_FunctionStateType

MCEM Driver Public functions State.

Each MCEM public function that is not re-entrant shall use this states to protect from preempting itself.

**Table 3-4. Enumeration Mcem\_FunctionStateType Values**

Value	Description
MCEM_S_IDLE = 0	Function can be executed.
MCEM_S_BUSY = 1	Function was invoked and has not finished yet.

### 3.8.2.2 Enumeration Mcem\_StateType

MCEM Driver State Enum.

Enumeration covering possible states of the driver.

**Table 3-5. Enumeration Mcem\_StateType Values**

Value	Description
MCEM_S_UNINIT = 0	MCEM module has not been initialized.
MCEM_S_INITIALIZED = 1	MCEM module has been initialized without hard-lock.
MCEM_S_LOCKED = 2	MCEM module has been initialized with hard-lock.

## 3.8.3 Function Reference

Functions of all functions supported by the driver are as per AUTOSAR MCEM Driver software specification Version 4.2 Rev0002 .

### 3.8.3.1 Function Mcem\_Init

Initialize the MCEM driver.

**Prototype:** Std\_ReturnType Mcem\_Init(const Mcem\_ConfigType \*pConfigPtr);

**Table 3-6. Mcem\_Init Arguments**

Type	Name	Direction	Description
const Mcem_ConfigType *	pConfigPtr	input	<ul style="list-style-type: none"> <li>Configuration pointer referring to the configuration structure. This pointer should always be a non-null, valid pointer.</li> </ul>

**Return:** Std\_ReturnType

**Table 3-7. Mcem\_Init Returns**

Value	Description
E_OK	Initialization of the driver succeeded. The driver is functional and its state is MCEM_S_LOCKED or MCEM_S_INITIALIZED, depending whether the configuration uses hard-locking or not.
E_NOT_OK	Initialization of the driver did not succeed. Configuration timeout expired, or the register configuration failed. Driver state is MCEM_S_UNINIT.

The function initializes the MCEM driver according to the configuration passed as parameter.

### 3.8.3.2 Function Mcem\_CheckNMI

Check whether an NMI has originated from the MCEM driver.

**Prototype:** Std\_ReturnType Mcem\_CheckNMI(void);

**Return:** Std\_ReturnType (extended)

**Table 3-8. Mcem\_CheckNMI Returns**

Value	Description
E_FCCU_SPURIOUS_NMI	The initialized MCEM (FCCU) might trigger a spurious interrupt.
E_OK	The NMI request originates in initialized MCEM (FCCU).
E_NOT_OK	Otherwise (MCEM not initialized, non-FCCU related NMI, FCCU malfunction).

Function checks whether the initialized MCEM module is the source of the NMI request.

### 3.8.3.3 Function Mcem\_GetErrors

Return logged error status.

**Prototype:** Std\_ReturnType Mcem\_GetErrors(Mcem\_FaultContainerType \*pErrorContainer);

**Table 3-9. Mcem\_GetErrors Arguments**

Type	Name	Direction	Description
Mcem_FaultContainerType *	pErrorContainer	output	Error container pointer.

**Return:** Std\_ReturnType

**Table 3-10. Mcem\_GetErrors Returns**

Value	Description
E_OK	No fault is pending.
E_FAULT_DETECTED	There is at least one logged fault.
E_NOT_OK	Otherwise (wrong driver state, error occurred while querying faults in the hardware).

Function stores status of each NCF into a container provided by the application.

### 3.8.3.4 Function Mcem\_ClearFaults

Clear software recoverable fault.

**Prototype:** Std\_ReturnType Mcem\_ClearFaults(Mcem\_FaultType nFaultId);

**Table 3-11. Mcem\_ClearFaults Arguments**

Type	Name	Direction	Description
Mcem_FaultType	nFaultId	input	ID of the fault that shall be cleared. If the fault is not marked as software recoverable, the request is ignored.

**Return:** Std\_ReturnType

**Table 3-12. Mcem\_ClearFaults Returns**

Value	Description
E_OK	Fault was cleared successfully.
E_NOT_OK	Fault was not cleared.

When a software recoverable fault occurs, a software routine shall remove the fault cause and then it shall clear the fault status using this function.

### 3.8.3.5 Function Mcem\_InjectFaults

Inject a fault.

**Prototype:** void Mcem\_InjectFaults(Mcem\_FaultType nFaultId);

**Table 3-13. Mcem\_InjectFaults Arguments**

Type	Name	Direction	Description
Mcem_FaultType	nFaultId	input	ID of the fault that shall be injected.

**Return:** void

Implements mechanism of the MCEM driver to inject faults into the hardware and test the reaction.

**Precondition:** Driver shall be initialized.

### 3.8.3.6 Function Mcem\_GetVersionInfo

Get version of the software module.

**Prototype:** void Mcem\_GetVersionInfo(Std\_VersionInfoType \*pVersionInfo);

**Table 3-14. Mcem\_GetVersionInfo Arguments**

Type	Name	Direction	Description
Std_VersionInfoType *	pVersionInfo	output	Pointer to a structure where the info about the version shall be stored.

**Return:** void

Returns the version information of this module.

## 3.8.4 Structs Reference

Data structures supported by the driver are as per AUTOSAR MCEM Driver software specification Version 4.2 Rev0002 .

### 3.8.4.1 Structure Mcem\_ConfigType

Configuration struct type.

MCEM Driver configuration structure type.

#### Declaration

```
typedef struct
{
    const boolean Lockable;
    const Mcem_Eirm_ConfigType * EirmConfig;
} Mcem_ConfigType;
```

**Table 3-15. Structure Mcem\_ConfigType member description**

Member	Description
Lockable	This configuration will be locked.
EirmConfig	EIRM Configuration values.

### 3.8.4.2 Structure Mcem\_FaultContainerType

Fault container struct type.

MCEM container for storing the errors.

#### Declaration

```
typedef struct
{
    Std_ReturnType nErrorStatus;
    Mcem_IPW_FaultContainerType au32Faults;
} Mcem_FaultContainerType;
```

**Table 3-16. Structure Mcem\_FaultContainerType member description**

Member	Description
nErrorStatus	Stored return value of Mcem_Geterrors().
au32Faults	Aggregated fault status container.

## 3.8.5 Types Reference

Types supported by the driver are as per AUTOSAR MCEM Driver software specification Version 4.2 Rev0002 .

### 3.8.5.1 Typedef Mcem\_FaultType

The type of FaultID.

**Type:** uint8

### 3.8.5.2 Typedef Mcem\_ErrorNotifyType

The notification functions shall have 2 parameters and no return value.

**Type:** void (\*Mcem\_ErrorNotifyType)(Mcem\_FaultType FaultID, uint32 u32FaultAddr);;

## 3.8.6 Variables Reference

Variables supported by the driver are as per MCEM Driver software specification.

### 3.8.6.1 Variable Mcem\_DriverState

Driver state.

The MCEM driver changes state during start-up or runtime from MCEM\_S\_UNIT to MCEM\_S\_INITIALIZED. If the configuration specify a lockable configuration, driver locks its configuration and change its state to MCEM\_S\_LOCKED.

### 3.8.6.2 Variable Mcem\_FunctionState

Public function status.

Each public function that is not re-entrant protects itself against pre-emption. This variables holds the status of each public function and changes each time function is executed from MCEM\_S\_IDLE to MCEM\_S\_BUSY, and back to MCEM\_S\_IDLE when the function exits.

### 3.8.6.3 Variable Mcem\_pConfigPtr

Latest Configuration pointer.

When the MCEM driver is initialized, the initialization pointer is stored.

## 3.8.7 Fault Management

Faults are identified using platform-specific symbolic names defined in chapter [Fault Names Reference](#) below. These names are valid for

- Mcem\_InjectFaults() function
- Mcem\_ClearFaults() function
- Fault status manipulation macros



Depending on the module configuration, faults may be reported to the application by means of alarm notification routine, which is provided by the MCEM. Additionally, a non-maskable interrupt can be triggered as a result of delayed alarm handling.

For interrupt-related details, refer to the **MCEM Integration Manual**.

### 3.8.7.1 Fault Status Manipulation Macros

The following macros are the only means to manipulate fault status information as returned from the `Mcem_GetErrors()` function call.

#### 3.8.7.1.1 Define MCEM\_FAULT\_PENDING

Access macro for querying collected fault status.

**Definition:** `#define MCEM_FAULT_PENDING( pError, nFault )`

#### 3.8.7.1.2 Define MCEM\_GETERRORS\_STATUS

Access macro for stored `Mcem_Geterrors()` return status.

**Definition:** `#define MCEM_GETERRORS_STATUS( pError )`

### 3.8.7.2 Fault Names Reference

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

**Table 3-17. 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.

## 3.9 Symbolic Names DISCLAIMER

All containers having the symbolic name tag set as true in the AUTOSAR schema will generate defines like:

```
#define <Container_Short_Name> <Container_ID>
```

For this reason it is forbidden to duplicate the name of such containers across the MCAL configuration, or to use names that may trigger other compile issues (e.g. match existing `#ifdef` arguments).

## Chapter 4

# Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the MCEM Driver. The most of the parameters are described below.

### 4.1 Configuration elements of MCEM

Included forms:

- IMPLEMENTATION\_CONFIG\_VARIANT
- McemGeneral
- CommonPublishedInformation
- McemConfigSet

### 4.2 Form IMPLEMENTATION\_CONFIG\_VARIANT

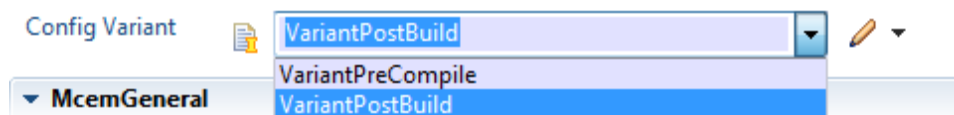


Figure 4-1. Tresos Plugin snapshot for IMPLEMENTATION\_CONFIG\_VARIANT form.

Table 4-1. Attribute IMPLEMENTATION\_CONFIG\_VARIANT detailed description

Property	Value
Label	Config Variant
Default	VariantPreCompile
Enable	true
Range	VariantPreCompile VariantPostBuild

## 4.3 Form McemGeneral

### McemGeneral

All general parameters of the MCEM driver are collected here.

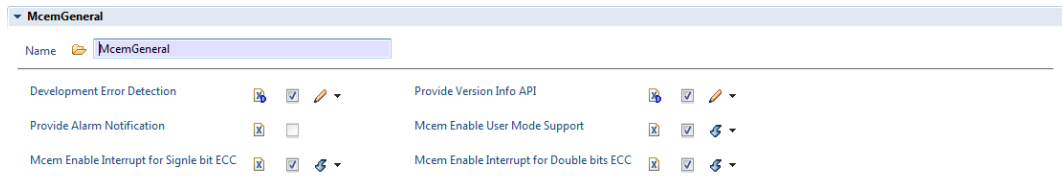


Figure 4-2. Tresos Plugin snapshot for McemGeneral form.

### 4.3.1 McemDevErrorDetect (McemGeneral)

#### Mcem Development Error Detect

Compile switch to enable/disable development error detection for this module.

- Unchecked: Development error detection disabled
- Checked: Development error detection enabled

Table 4-2. Attribute McemDevErrorDetect (McemGeneral) detailed description

Property	Value
Label	Development Error Detection
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.2 McemVersionInfoApi (McemGeneral)

#### MCEM VersionInfo Api

Compile switch to enable/disable the version information API.

- Checked: API enabled
- Unchecked:API disabled

Table 4-3. Attribute McemVersionInfoApi (McemGeneral) detailed description

Property	Value
Label	Provide Version Info API

Table continues on the next page...

**Table 4-3. Attribute McemVersionInfoApi (McemGeneral) detailed description (continued)**

Property	Value
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.3 McemAlarmNotificationApi (McemGeneral)

#### MCEM Alarm Notification

Compile switch to enable/disable the alarm notification .

- Checked: Notification enabled
- Unchecked: Notification disabled

**Table 4-4. Attribute McemAlarmNotificationApi (McemGeneral) detailed description**

Property	Value
Label	Provide Alarm Notification
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.4 McemEnableUserModeSupport (McemGeneral)

#### MCEM Enable User Mode Support

Compile switch to enable/disable User Mode Support.

- Checked: Enabled
- Unchecked: Disabled

**Table 4-5. Attribute McemEnableUserModeSupport (McemGeneral) detailed description**

Property	Value
Label	Enable User Mode support
Type	BOOLEAN
Origin	Vendor specific
Symbolic Name	false
Default	false

### 4.3.5 McemEnableIsrSingleErrorBit (McemGeneral)

#### MCEM Enable ISR for Single Error Bit detected

Compile switch to enable/disable ISR handler for Single Error Bit detected.

- Checked: enabled
- Unchecked: disabled

**Table 4-6. Attribute McemEnableIsrSingleErrorBit (McemGeneral) detailed description**

Property	Value
Label	Provide IRQ handler for Single Error Bit detected
Type	BOOLEAN
Origin	Vendor Specific
Symbolic Name	false
Default	false

### 4.3.6 McemEnableIsrDoubleErrorBit (McemGeneral)

#### MCEM Enable ISR for Multiple Error Bits detected

Compile switch to enable/disable ISR for Multiple Error Bits detected.

- Checked: Enabled
- Unchecked: Disabled

**Table 4-7. Attribute McemEnableIsrDoubleErrorBit (McemGeneral) detailed description**

Property	Value
Label	Provide ISR handler for Multiple Error Bits detected
Type	BOOLEAN
Origin	Vendor Specific
Symbolic Name	false
Default	false

## 4.4 Form McemConfigSet

This container is the base for a multiple configuration set.

## Included forms:

- [Form Fault](#)

The screenshot shows the 'McemConfigSet' configuration window. It includes a 'Name' field with 'McemConfigSet'. Below are several configuration items with their current values and edit icons:

- Fault Recovery Timeout (0 -> 4294967295): 65535
- Fault Output Mode Selection: EOUT\_DualRail
- Hard-Locked Configuration Set: [checkbox]
- EOUT Switching Mode: [checkbox]
- Mcem Error Notification with Error Address API: Error\_notification
- Mcem Lower TCM address (for Injected) (0x1c000000 -> 0x1ffffff): 0x1ffffff0
- Mcem Upper TCM address (for Injected) (0x20000000 -> 0x200fffff): 0x20000000
- Mcem Inject Bit 1: DATA\_BIT\_0
- Mcem Inject Bit 2: DATA\_BIT\_1

**Figure 4-3. Tresos Plugin snapshot for McemConfigSet form.**

### 4.4.1 FaultTimeout (McemConfigSet)

Non-critical fault timeout.

**Table 4-8. Attribute FaultTimeout (McemConfigSet) detailed description**

Property	Value
Label	Fault Recovery Timeout
Type	INTEGER
Default	65535
Invalid	Range >=0 <4294967296

### 4.4.2 EOUT\_FaultOutputMode (McemConfigSet)

Fault Output Mode selection

- Dual-Rail (default state) [fccu\_eout[1:0]= outputs]
- Time Switching [fccu\_eout[1:0]= output to be used]
- Bi-Stable

Note: In Testx mode, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs on the system/safe clock.

**Table 4-9. Attribute EOUT\_FaultOutputMode (McemConfigSet) detailed description**

Property	Value
Label	Fault Output Mode Selection
Type	ENUMERATION
Default	EOUT_DualRail
Range	EOUT_DualRail EOUT_TimeSwitching EOUT_BiStable

### 4.4.3 McemHardlockedConfiguration (McemConfigSet)

Lock this configuration set:

- Unchecked: After init, the configuration is not locked.
- Checked: after init, the configuration is hardlocked.

### 4.4.4 McemSoftLockedConfiguration (McemConfigSet)

Lock this configuration set:

- Unchecked: After init, the configuration is not locked.
- Checked: after init, the configuration is hardlocked.

**Table 4-10. Attribute McemSoftLockedConfiguration (McemConfigSet) detailed description**

Property	Value
Label	Lock this configuration set
Type	BOOLEAN
Default	False

### 4.4.5 EOUT\_SwitchingMode (McemConfigSet)

EOUT Switching mode

- Unchecked: EOUT protocol (dual-rail, time-switching) slow switching mode.
- Checked: EOUT protocol (dual-rail, time-switching) fast switching mode.

Switching Mode has no effect on the bi-stable protocol.



**Table 4-11. Attribute EOUT\_SwitchingMode (McemConfigSet) detailed description**

Property	Value
Label	EOUT Switching Mode
Type	BOOLEAN
Default	False

#### 4.4.6 EOUT\_PolaritySelection (McemConfigSet)

EOUT Polarity Selection

- Unchecked: fcc\_eout[1] active high, fcc\_eout[0] active low.
- Checked: fcc\_eout[1] active low, fcc\_eout[0] active high.

This bit can be read and written by the software. Active state means a state indicating FAULT and applicable for non-toggling protocols only..

**Table 4-12. Attribute EOUT\_PolaritySelection (McemConfigSet) detailed description**

Property	Value
Label	EOUT Polarity Selection
Type	BOOLEAN
Default	False

#### 4.4.7 McemErrorNotificationWithAddressAPI (McemConfigSet)

In case of a failure event, if interrupt is enable, tha fault shall be notified to Application using this API.

**Table 4-13. Attribute McemErrorNotificationWithAddressAPI (McemConfigSet) detailed description**

Property	Value
Label	Mcem Error Notification with Error Address API
Type	FUNCTION-NAME
Default	NULL_PTR

#### 4.4.8 McemTcmLowerAddr (McemConfigSet)

Mcem Lower TCM address (for Injected).

**Table 4-14. Attribute McemTcmLowerAddr (McemConfigSet) detailed description**

Property	Value
Label	Mcem Lower TCM address
Type	INTEGER
Default	0x1ffff00
Invalid	Range >=0x1c000000 <=0x1ffffff

#### 4.4.9 McemTcmUpperAddr (McemConfigSet)

Mcem Upper TCM address (for Injected).

**Table 4-15. Attribute McemTcmUpperAddr (McemConfigSet) detailed description**

Property	Value
Label	Mcem Upper TCM address
Type	INTEGER
Default	0x20000000
Invalid	Range >=0x20000000 <=0x200ffff

#### 4.4.10 McemInjectBit1 (McemConfigSet)

Select bit 1 for use in Mcem\_InjectFault

- CHECK\_BIT\_0
- CHECK\_BIT\_1
- CHECK\_BIT\_2
- CHECK\_BIT\_3
- CHECK\_BIT\_4
- CHECK\_BIT\_5
- CHECK\_BIT\_6
- DATA\_BIT\_0
- DATA\_BIT\_1
- DATA\_BIT\_2
- DATA\_BIT\_3
- DATA\_BIT\_4
- DATA\_BIT\_5

- DATA\_BIT\_6
- DATA\_BIT\_7
- DATA\_BIT\_8
- DATA\_BIT\_9
- DATA\_BIT\_10
- DATA\_BIT\_11
- DATA\_BIT\_12
- DATA\_BIT\_13
- DATA\_BIT\_14
- DATA\_BIT\_15
- DATA\_BIT\_16
- DATA\_BIT\_17
- DATA\_BIT\_18
- DATA\_BIT\_19
- DATA\_BIT\_20
- DATA\_BIT\_21
- DATA\_BIT\_22
- DATA\_BIT\_23
- DATA\_BIT\_24
- DATA\_BIT\_25
- DATA\_BIT\_26
- DATA\_BIT\_27
- DATA\_BIT\_28
- DATA\_BIT\_29
- DATA\_BIT\_30
- DATA\_BIT\_31

**Table 4-16. Attribute McemInjectBit1 (McemConfigSet) detailed description**

Property	Value
Label	Fault Output Mode Selection
Type	ENUMERATION
Default	EOUT_DualRail
Range	CHECK_BIT_0 CHECK_BIT_1 CHECK_BIT_2 CHECK_BIT_3 CHECK_BIT_4 CHECK_BIT_5 CHECK_BIT_6 DATA_BIT_0 DATA_BIT_1 DATA_BIT_2 DATA_BIT_3 DATA_BIT_4

**Table 4-16. Attribute McemInjectBit1 (McemConfigSet) detailed description**

Property	Value
	DATA_BIT_5
	DATA_BIT_6
	DATA_BIT_7
	DATA_BIT_8
	DATA_BIT_9
	DATA_BIT_10
	DATA_BIT_11
	DATA_BIT_12
	DATA_BIT_13
	DATA_BIT_14
	DATA_BIT_15
	DATA_BIT_16
	DATA_BIT_17
	DATA_BIT_18
	DATA_BIT_19
	DATA_BIT_20
	DATA_BIT_21
	DATA_BIT_22
	DATA_BIT_23
	DATA_BIT_24
	DATA_BIT_25
	DATA_BIT_26
	DATA_BIT_27
	DATA_BIT_28
	DATA_BIT_29
	DATA_BIT_30
	DATA_BIT_31

#### 4.4.11 McemInjectBit2 (McemConfigSet)

Select bit 2 for use in Mcem\_InjectFault

- CHECK\_BIT\_0
- CHECK\_BIT\_1
- CHECK\_BIT\_2
- CHECK\_BIT\_3
- CHECK\_BIT\_4
- CHECK\_BIT\_5
- CHECK\_BIT\_6
- DATA\_BIT\_0
- DATA\_BIT\_1
- DATA\_BIT\_2
- DATA\_BIT\_3
- DATA\_BIT\_4
- DATA\_BIT\_5
- DATA\_BIT\_6

- DATA\_BIT\_7
- DATA\_BIT\_8
- DATA\_BIT\_9
- DATA\_BIT\_10
- DATA\_BIT\_11
- DATA\_BIT\_12
- DATA\_BIT\_13
- DATA\_BIT\_14
- DATA\_BIT\_15
- DATA\_BIT\_16
- DATA\_BIT\_17
- DATA\_BIT\_18
- DATA\_BIT\_19
- DATA\_BIT\_20
- DATA\_BIT\_21
- DATA\_BIT\_22
- DATA\_BIT\_23
- DATA\_BIT\_24
- DATA\_BIT\_25
- DATA\_BIT\_26
- DATA\_BIT\_27
- DATA\_BIT\_28
- DATA\_BIT\_29
- DATA\_BIT\_30
- DATA\_BIT\_31

**Table 4-17. Attribute McemInjectBit2 (McemConfigSet) detailed description**

Property	Value
Label	Fault Output Mode Selection
Type	ENUMERATION
Default	EOUT_DualRail
Range	CHECK_BIT_0 CHECK_BIT_1 CHECK_BIT_2 CHECK_BIT_3 CHECK_BIT_4 CHECK_BIT_5 CHECK_BIT_6 DATA_BIT_0 DATA_BIT_1 DATA_BIT_2 DATA_BIT_3 DATA_BIT_4 DATA_BIT_5

Table 4-17. Attribute McemInjectBit2 (McemConfigSet) detailed description

Property	Value
	DATA_BIT_6
	DATA_BIT_7
	DATA_BIT_8
	DATA_BIT_9
	DATA_BIT_10
	DATA_BIT_11
	DATA_BIT_12
	DATA_BIT_13
	DATA_BIT_14
	DATA_BIT_15
	DATA_BIT_16
	DATA_BIT_17
	DATA_BIT_18
	DATA_BIT_19
	DATA_BIT_20
	DATA_BIT_21
	DATA_BIT_22
	DATA_BIT_23
	DATA_BIT_24
	DATA_BIT_25
	DATA_BIT_26
	DATA_BIT_27
	DATA_BIT_28
	DATA_BIT_29
	DATA_BIT_30
	DATA_BIT_31

4.4.12 DefaultFaultConfig Collection (McemConfigSet)

By default, all faults except reserved ones are enabled. Default behavior is configured in this section. If needed, faults can be disabled or configured individually in the [Form Fault](#) map.

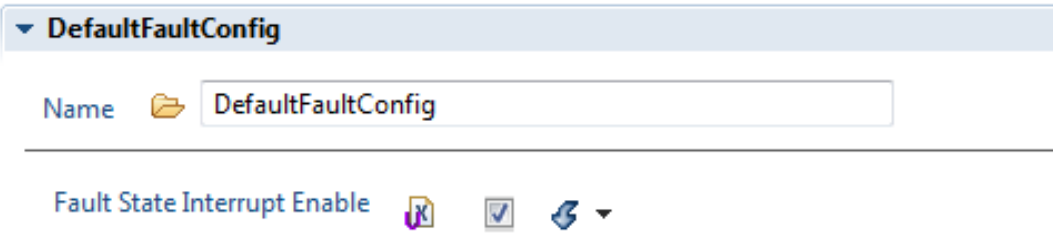


Figure 4-4. Tresos Plugin snapshot for McemConfigSet form.

### 4.4.12.1 FaultIrqEnabled (DefaultConfig)

IRQ enable

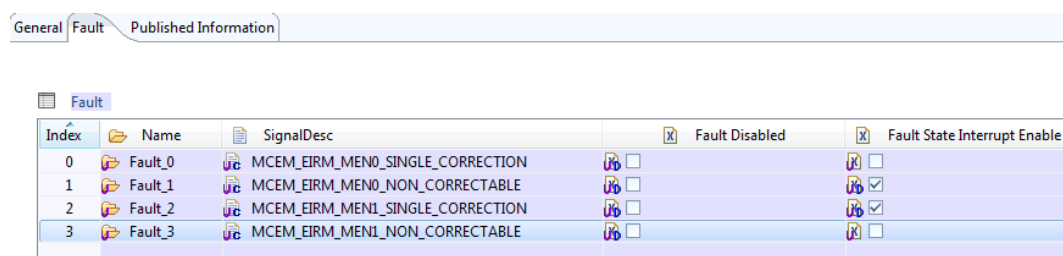
- Unchecked: IRQ is disabled for error source x.
- Checked: IRQ is enabled for error source x.

**Table 4-18. Attribute FaultIrqEnabled (DefaultConfig) detailed description**

Property	Value
Label	Enable Interrupt for default Faults
Type	BOOLEAN
Default	true

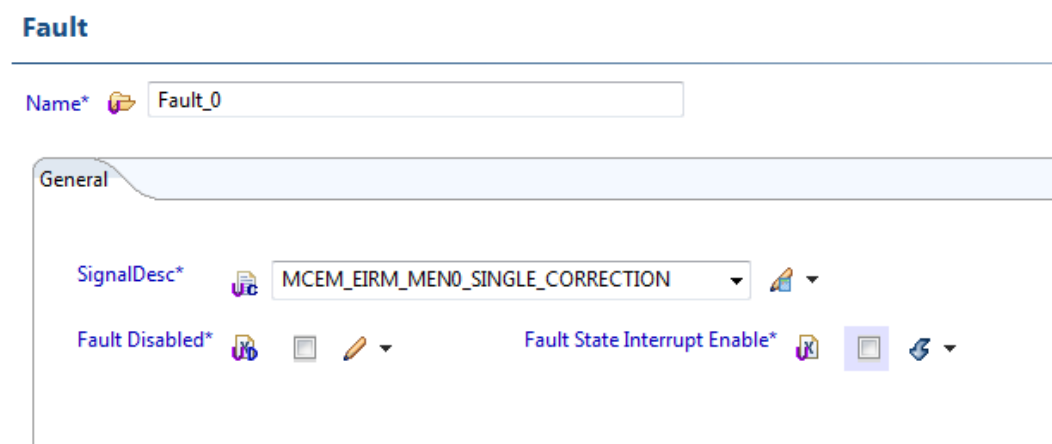
### 4.4.13 Form Fault

Is included by form: [Form McemConfigSet](#)



Index	Name	SignalDesc	Fault Disabled	Fault State Interrupt Enable
0	Fault_0	MCCEM_EIRM_MEN0_SINGLE_CORRECTION	<input type="checkbox"/>	<input type="checkbox"/>
1	Fault_1	MCCEM_EIRM_MEN0_NON_CORRECTABLE	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	Fault_2	MCCEM_EIRM_MEN1_SINGLE_CORRECTION	<input type="checkbox"/>	<input checked="" type="checkbox"/>
3	Fault_3	MCCEM_EIRM_MEN1_NON_CORRECTABLE	<input type="checkbox"/>	<input type="checkbox"/>

**Figure 4-5. Tresos Plugin snapshot for Fault Map**



**Fault**

Name\*

**General**

SignalDesc\*

Fault Disabled\* ☐

Fault State Interrupt Enable\* ☐

**Figure 4-6. Tresos Plugin snapshot for Fault Form.**

### 4.4.13.1 SignalDesc (Fault)

By default, all non-reserved faults are enabled and handled according to the default configurations.

The explicitly created fault is enabled, and the default values for recovery and reaction type are selected. It can be disabled or configured differently.

**Table 4-19. Attribute SignalDesc (Fault) detailed description**

Property	Value
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

### 4.4.13.2 FaultDisabled (Fault)

Fault source disabled

- Unchecked: Specified non-critical fault source is enabled (default).
- Checked: Specified non-critical fault source is disabled.

**Table 4-20. Attribute FaultDisabled (Fault) detailed description**

Property	Value
Label	Fault Disabled
Type	BOOLEAN
Default	false

### 4.4.13.3 FaultIrqEnabled (DefaultConfig)

IRQ enable

- Unchecked: IRQ is disabled for error source x.
- Checked: IRQ is enabled for error source x.

**Table 4-21. Attribute FaultIrqEnabled (DefaultConfig) detailed description**

Property	Value
Label	Enable Interrupt for default Faults
Type	BOOLEAN
Default	true



## 4.5 Form CommonPublishedInformation

Common container, aggregated by all modules. It contains published information about vendor and versions.

Property	Value
AUTOSAR Release Major Version	4
AUTOSAR Release Minor Version	2
AUTOSAR Release Revision Version	2
Numeric Module ID	0
Software Major Version	1
Software Minor Version	0
Software Patch Version	2
Vendor Api Infix	
Vendor ID	43

Figure 4-7. Tresos Plugin snapshot for CommonPublishedInformation form.

### 4.5.1 ArReleaseMajorVersion (CommonPublishedInformation)

Major version number of AUTOSAR specification on which the appropriate implementation is based on.

**Table 4-22. Attribute ArReleaseMajorVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	AUTOSAR Major Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	4
Invalid	Range <div> <div>&gt;=4</div> <div>&lt;=4</div> </div>

## 4.5.2 ArReleaseMinorVersion (CommonPublishedInformation)

Minor version number of AUTOSAR specification on which the appropriate implementation is based on.

**Table 4-23. Attribute ArReleaseMinorVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	AUTOSAR Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	2
Invalid	Range >=2 <=2

## 4.5.3 ArReleaseRevisionVersion (CommonPublishedInformation)

Revision version number of AUTOSAR specification on which the appropriate implementation is based on.

**Table 4-24. Attribute ArReleaseRevisionVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	AUTOSAR Release Revision Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	2
Invalid	Range >=2 <=2

## 4.5.4 ModuleId (CommonPublishedInformation)

Module ID of this module from Module List.

**Table 4-25. Attribute ModuleId (CommonPublishedInformation) detailed description**

Property	Value
Label	Module Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range >=0 <=0

### 4.5.5 SwMajorVersion (CommonPublishedInformation)

Major version number of the vendor specific implementation of the module. The numbering is vendor specific.

**Table 4-26. Attribute SwMajorVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	Software Major Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	1
Invalid	Range >=1 <=1

### 4.5.6 SwMinorVersion (CommonPublishedInformation)

Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

**Table 4-27. Attribute SwMinorVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	Software Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false

*Table continues on the next page...*

**Table 4-27. Attribute SwMinorVersion (CommonPublishedInformation) detailed description (continued)**

Property	Value
Default	0
Invalid	Range >=0 <=0

### 4.5.7 SwPatchVersion (CommonPublishedInformation)

Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

**Table 4-28. Attribute SwPatchVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	Software Patch Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	2
Invalid	Range >=2 <=2

### 4.5.8 VendorApiInfix (CommonPublishedInformation)

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name. This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows:

<ModuleName>\_>VendorId>\_<VendorApiInfix><Api name from SWS>. E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name Can\_Write defined in the SWS will translate to Can\_123\_v11r456Write. This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

**Table 4-29. Attribute VendorApiInfix (CommonPublishedInformation) detailed description**

Property	Value
Label	Vendor Api Infix
Type	STRING_LABEL
Origin	Custom
Symbolic Name	false
Default	
Enable	false

### 4.5.9 VendorId (CommonPublishedInformation)

Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.

**Table 4-30. Attribute VendorId (CommonPublishedInformation) detailed description**

Property	Value
Label	Vendor Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	43
Invalid	Range <div style="margin-left: 20px;">&gt;=43</div> <div style="margin-left: 20px;">&lt;=43</div>



**How to Reach Us:****Home Page:**[nxp.com](http://nxp.com)**Web Support:**[nxp.com/support](http://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](http://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.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2019 NXP B.V.

Document Number UM2MCEMASR4.2 Rev0002R1.0.2  
Revision 1.0