User Manual

for S32K14X MCEM Driver

Document Number: UM2MCEMASR4.2 Rev0002R1.0.1
Rev. 1.0

Contents

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

Chapter 1
Revision History

Chapter 2
Introduction

Chapter 3
Driver

Chapter 4
Tresos Configuration Plug-in

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	13/07/2018	NXP MCAL Team	Updated version for ASR 4.2.2S32K14X1.0.1 Release

Chapter 2
Introduction

This User Manual describes the NXP SemiconductorsAUTOSAR complex driver *Microcontroller Error Manager* (MCEM) for S32K14X.

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

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

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

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
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	S32K14X Reference Manual	Reference Manual, Rev. 7, 4/2018
2	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	30/11/2017
3	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	30/11/2017
4	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	30/11/2017
5	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	30/11/2017
6	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	26/02/2018

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
MCEM_SW011	N/A	The initialization structure shall contain among other parameters: -fault timeout enable/disable - fault timeout value	Not applicable for S32K14X
MCEM_SW020	N/A	The initialization function shall handle the Configuration Timeout IRQ.	Not applicable for S32K14X
MCEM_SW013	N/A	MCEM driver shall provide an interface to collect latched faults. The faults include - All faults handled by FCCU - All memory-related faults handled by MEMU (RAM, peripheral memory, flash memory)	Not applicable for S32K14X
MCEM_SW038	N/A	If the Mcem_GetErrors function failed to collect latched faults, an error labeled Mcem_E_OperationFailed shall be reported to SERR.	Ticket AAI-282 is created to reject this requirement
MCEM_SW018	N/A	MCEM driver shall provide interrupt routine for ALARM IRQ.	Not applicable for S32K14X
MCEM_SW019	N/A	If the allarm callback is NULL, the alarm ISR shall invoke SERR notification with an error labeled MCEM_E_ALARM_ISR.	Ticket AAI-282 is created to reject this requirement
MCEM_SW040	N/A	If the allarm callback is not NULL, the alarm ISR shall invoke the alarm callback defined in the configuration.	Not applicable for S32K14X
MCEM_SW042	N/A	If Mcem_ClearFaults did not cleared the fault, an error labeled Mcem_E_OperationFailed has to be reported to SERR.	Ticket AAI-282 is created to reject this requirement
MCEM_SW048	N/A	Fccu ALARM ISR shall check on invalid error codes and shall notify the SERR with a non critical error labeled Mcem_E_Forbidden_Invocation. The ALARM ISR shall return without calling the alarm notification handler.	Ticket AAI-282 is created to reject this requirement

Table continues on the next page...

Table 3-2. MCEM Deviations (continued)

Req.	Status	Description	Notes
MCEM_SW049	N/A	MCEM faults (Mcem_FaultType) shall be mapped to a set of platform specific symbolic values (indices) covering both FCCU and MEMU/ECSM faults. A dedicated value MCEM_FCCU_MULTIPLE_FAULTS is reserved for the ALARM ISR/handler to indicate the fact the alarm has been triggered due to multiple faults which FCCU cannot distinguish.	Not applicable for S32K14X
MCEM_SW053	N/A	The initialization structure shall contain among other parameters fault enable/disable configuration for all faults handled by the FCCU.	Not applicable for S32K14X
MCEM_SW054	N/A	The initialization structure shall contain among other parameters configuration of EOUT interface protocol.	Not applicable for S32K14X
MCEM_SW055	N/A	The initialization structure shall contain among other parameters configuration of polarity of EOUT interface signals.	Not applicable for S32K14X
MCEM_SW056	N/A	The initialization structure shall contain among other parameters configuration of switching mode of EOUT interface signals.	Not applicable for S32K14X
MCEM_SW057	N/A	The MCEM module shall provide a public API named Mcem_CheckNMI for the NMI handler to be able to confirm the FCCU has triggered a valid NMI request. These checks include also detection of documented spurious NMI requests due to internal FCCU causes.	Not applicable for S32K14X
MCEM_SW058	N/A	If the MCEM module has been initialized, the Mcem_CheckNMI() function shall return - E_FCCU_SPURIOUS_NMI if the FCCU was identified as possible originator of a spurious NMI IRQ, - E_OK if the FCCU NMI IRQ flag is asserted.	Not applicable for S32K14X
MCEM_SW059	N/A	If one of the following conditions is not met: - The MCEM module has been initialized, - The FCCU has been identified as a source of regular NMI request or suspect spurious NMI request, the Mcem_CheckNMI() function shall return E_NOT_OK.	Not applicable for S32K14X
MCEM_SW060	N/A	The check NMI API prototype shall be Std_ReturnType Mcem_CheckNMI(void)	Not applicable for S32K14X
SMCAL_SW164	N/A	The MCEM module shall define a boolean post-build configuration parameter named McemHardlockedConfiguration	Not applicable for S32K14X

3.5 Driver limitations

None

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**	• Configuration pointer referring to the configuration structure. This pointer should always be a non-null, valid pointer.

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	Othewise (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 occured 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_UINIT 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

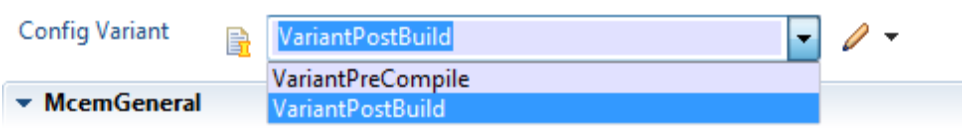

Figure 4-1. Tresos Plugin snapshot for IMPLEMENTATION_CONFIG_VARIANT form.

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.

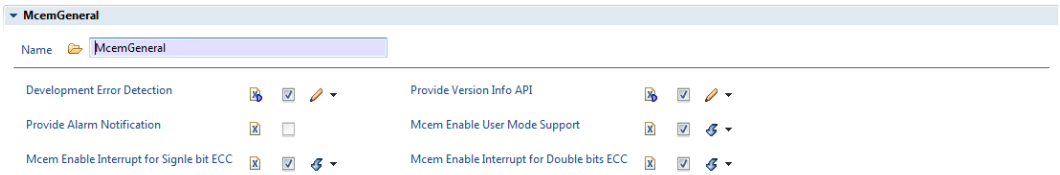

Figure 4-2. Tresos Plugin snapshot for McemGeneral form.

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

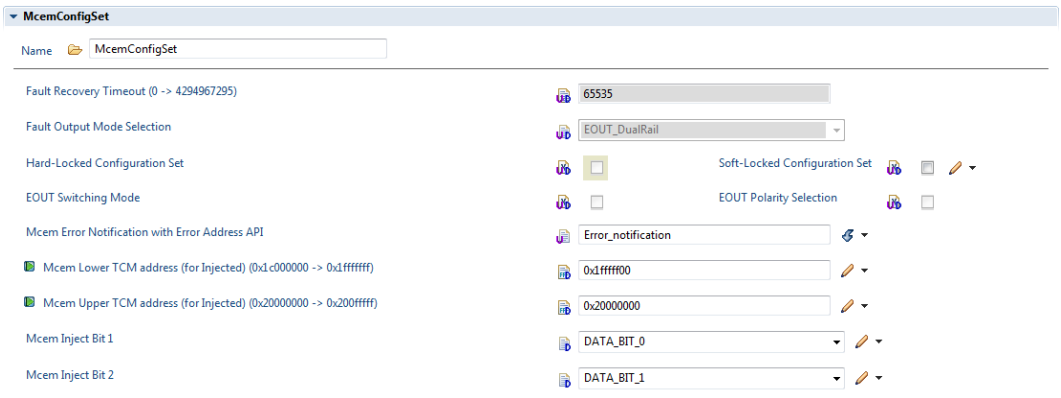

Figure 4-3. Tresos Plugin snapshot for McemConfigSet form.

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.

Table 4-10. Attribute McemHardlockedConfiguration (McemConfigSet) detailed description

Property	Value
Label	Hard lock this configuration set
Type	BOOLEAN
Default	False

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 4-11. 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 4-12. 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 4-13. 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 4-14. 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 4-15. Attribute McemTcmLowerAddr (McemConfigSet) detailed description

Property	Value
Label	Mcem Lower TCM address
Type	INTEGER
Default	0x1fffff00
Invalid	Range >=0x1c000000 <=0x1fffffff

4.4.9 McemTcmUpperAddr (McemConfigSet)

Mcem Upper TCM address (for Injected).

Table 4-16. Attribute McemTcmUpperAddr (McemConfigSet) detailed description

Property	Value
Label	Mcem Upper TCM address
Type	INTEGER
Default	0x20000000
Invalid	Range >=0x20000000 <=0x200fffff

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 4-17. 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 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 4-18. Attribute McemInjectBit2 (McemConfigSet) detailed description

Property	Value
Label	Fault Output Mode Selection

Table continues on the next page...

Table 4-18. Attribute McemInjectBit2 (McemConfigSet) detailed description (continued)

Property	Value
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 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.

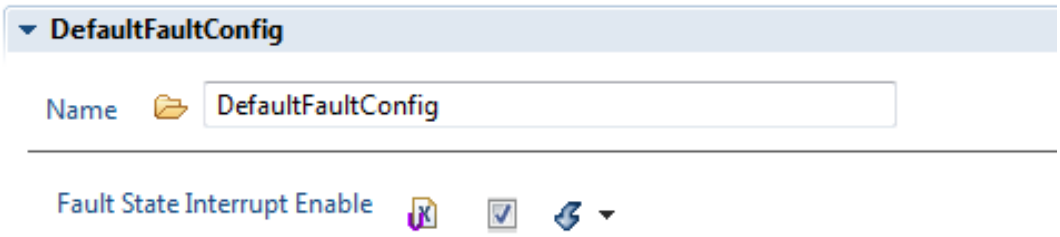

Figure 4-4. Tresos Plugin snapshot for McemConfigSet form.

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 4-19. Attribute FaultIrqEnabled (DefaultConfig) detailed description

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

4.4.13 Form Fault

Is included by form: Form McemConfigSet

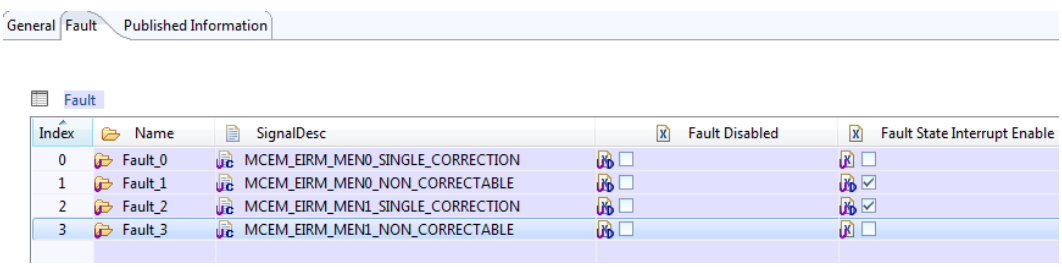

Figure 4-5. Tresos Plugin snapshot for Fault Map

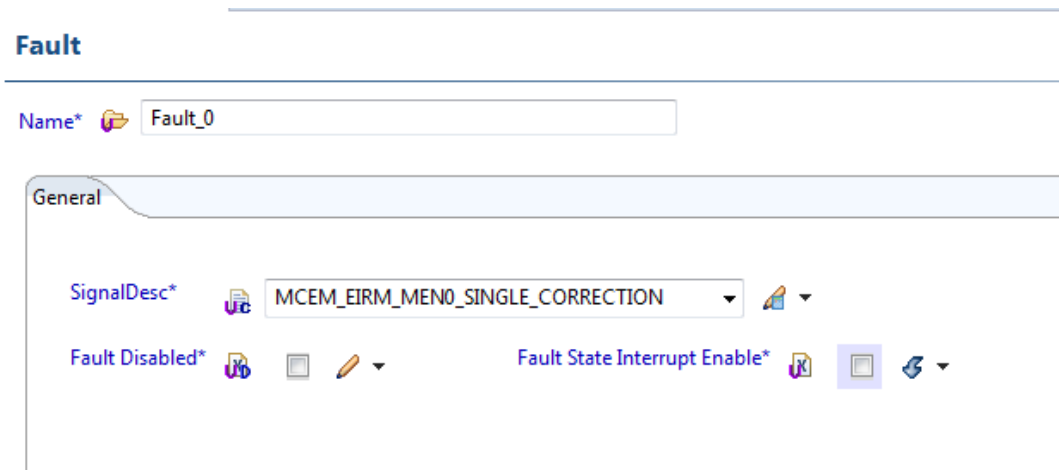

Figure 4-6. Tresos Plugin snapshot for Fault Form.

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 4-20. 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 4-21. 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 4-22. Attribute FaultIrqEnabled (DefaultConfig) detailed description

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

4.5 Form CommonPublishedInformation

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

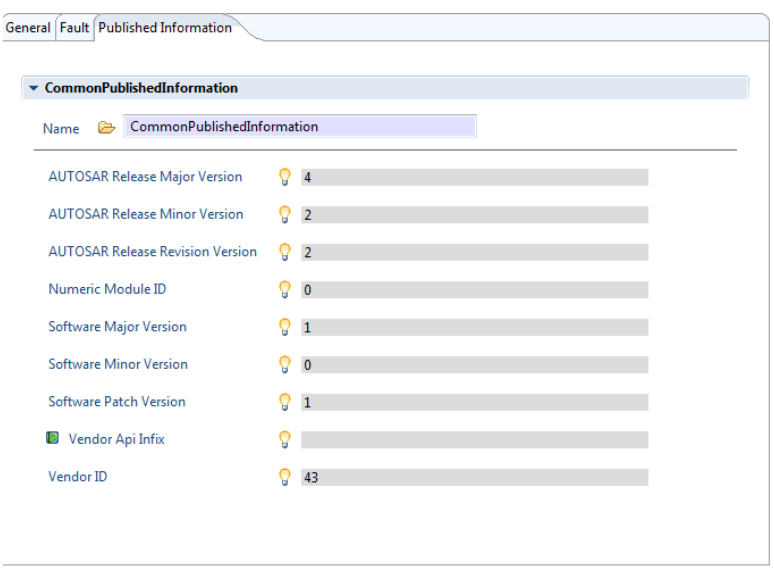

Figure 4-7. Tresos Plugin snapshot for CommonPublishedInformation form.

4.5.1 ArReleaseMajorVersion (CommonPublishedInformation)

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

Table 4-23. Attribute ArReleaseMajorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	AUTOSAR Major Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	4
Invalid	Range >=4 <=4

4.5.2 ArReleaseMinorVersion (CommonPublishedInformation)

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

Table 4-24. 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 4-25. Attribute ArReleaseRevisionVersion (CommonPublishedInformation) detailed description

Property	Value
Label	AUTOSAR Release Revision Version
Type	INTEGER_LABEL

Table continues on the next page...

Table 4-25. Attribute ArReleaseRevisionVersion (CommonPublishedInformation) detailed description (continued)

Property	Value
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 4-26. 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 4-27. 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 4-28. Attribute SwMinorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	Software Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
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 4-29. Attribute SwPatchVersion (CommonPublishedInformation) detailed description

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

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 4-30. 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 4-31. Attribute VendorId (CommonPublishedInformation) detailed description

Property	Value
Label	Vendor Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	43
Invalid	Range >=43 <=43

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number UM2MCEMASR4.2 Rev0002R1.0.1
Revision 1.0

AUTOSAR