

# User Manual

for S32K1 BASE Driver

Document Number: UM2BASEASR4.4 Rev0000R1.0.1 Rev. 1.0

<b>1 Revision History</b>	<b>2</b>
<b>2 Introduction</b>	<b>3</b>
2.1 Supported Derivatives	3
2.2 Overview	4
2.3 About This Manual	5
2.4 Acronyms and Definitions	6
2.5 Reference List	6
<b>3 Driver</b>	<b>7</b>
3.1 Requirements	7
3.2 Driver Design Summary	7
3.3 Hardware Resources	10
3.4 Deviations from Requirements	10
3.5 Driver Limitations	28
3.6 Driver usage and configuration tips	28
3.6.1 NO_STDINT_H compiler symbol	28
3.7 Runtime errors	29
3.8 Symbolic Names Disclaimer	29
<b>4 Tresos Configuration Plug-in</b>	<b>30</b>
4.1 Module Base	30
4.2 Container OsIfGeneral	31
4.3 Parameter OsIfMulticoreSupport	31
4.4 Parameter OsIfEnableUserModeSupport	32
4.5 Parameter OsIfDevErrorDetect	32
4.6 Parameter OsIfUseSystemTimer	33
4.7 Parameter OsIfUseCustomTimer	33
4.8 Parameter OsIfInstanceId	33
4.9 Reference OsIfEcucPartitionRef	34
4.10 Container OsIfOperatingSystemType	34
4.11 Container OsIfCounterConfig	35
4.12 Reference OsIfCounterEcucPartitionRef	35
4.13 Reference OsIfSystemTimerClockRef	36
4.14 Reference OsIfOsCounterRef	36
4.15 Container CommonPublishedInformation	37
4.16 Parameter ArReleaseMajorVersion	37
4.17 Parameter ArReleaseMinorVersion	38
4.18 Parameter ArReleaseRevisionVersion	38
4.19 Parameter ModuleId	38
4.20 Parameter SwMajorVersion	39

4.21 Parameter SwMinorVersion . . . . .	39
4.22 Parameter SwPatchVersion . . . . .	40
4.23 Parameter VendorApiInfix . . . . .	40
4.24 Parameter VendorId . . . . .	41
<b>5 Module Index</b>	<b>42</b>
5.1 Software Specification . . . . .	42
<b>6 Module Documentation</b>	<b>43</b>
6.1 OsIf . . . . .	43
6.1.1 Detailed Description . . . . .	43
6.1.2 Enum Reference . . . . .	43
6.1.3 Function Reference . . . . .	44
6.2 BASE_COMPONENT . . . . .	47
6.2.1 Detailed Description . . . . .	47
6.2.2 Data Structure Documentation . . . . .	67
6.2.3 Macro Definition Documentation . . . . .	75
6.2.4 Types Reference . . . . .	137
6.2.5 Enum Reference . . . . .	144
6.2.6 Variable Documentation . . . . .	159



## Chapter 1

### Revision History

Revision	Date	Author	Description
1.0	24.02.2022	NXP RTD Team	Prepared for release RTD S32K1 Version 1.0.1

## Chapter 2

### Introduction

- [Supported Derivatives](#)
- [Overview](#)
- [About This Manual](#)
- [Acronyms and Definitions](#)
- [Reference List](#)

This User Manual describes NXP Semiconductor AUTOSAR Base for S32K1. AUTOSAR Base driver configuration parameters and deviations from the specification are described in Driver chapter of this document.

### 2.1 Supported Derivatives

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

- s32k116\_qfn32
- s32k116\_lqfp48
- s32k118\_lqfp48
- s32k118\_lqfp64
- s32k142\_lqfp48
- s32k142\_lqfp64
- s32k142\_lqfp100
- s32k142w\_lqfp48
- s32k142w\_lqfp64
- s32k144\_lqfp48
- s32k144\_lqfp64

- s32k144\_lqfp100
- s32k144\_mapbga100
- s32k144w\_lqfp48
- s32k144w\_lqfp64
- s32k146\_lqfp64
- s32k146\_lqfp100
- s32k146\_mapbga100
- s32k146\_lqfp144
- s32k148\_lqfp100
- s32k148\_mapbga100
- s32k148\_lqfp144
- s32k148\_lqfp176

All of the above microcontroller devices are collectively named as S32K1.

## 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** style: Used for important terms, notes and warnings.
- *Italic* style: 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.

Warning

This is a warning

## 2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CS	Chip Select
CTU	Cross Trigger Unit
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
FIFO	First In First Out
LSB	Least Significant Bit
MCU	Micro Controller Unit
MIDE	Multi Integrated Development Environment
MSB	Most Significant Bit
N/A	Not Applicable
RAM	Random Access Memory
SIU	Systems Integration Unit
SWS	Software Specification
VLE	Variable Length Encoding
XML	Extensible Markup Language

## 2.5 Reference List

#	Title	Version
1	General Specification of Basic Software Modules	AUTOSAR Release 4.4.0
2	Specification of Communication Stack Types	AUTOSAR Release 4.4.0
3	Specification of Compiler Abstraction	AUTOSAR Release 4.4.0
4	Specification of Platform Types	AUTOSAR Release 4.4.0
5	Specification of Standard Types	AUTOSAR Release 4.4.0
6	S32K1xx Series Reference Manual	Rev. 14, 09/2021
7	Errata	S32K116_0N96V Rev. 22/OCT/2021
		S32K118_0N97V Rev. 22/OCT/2021
		S32K142_0N33V Rev. 22/OCT/2021
		S32K144_0N57U Rev. 22/OCT/2021
		S32K144W_0P64A Rev. 22/OCT/2021
		S32K146_0N73V Rev. 22/OCT/2021
		S32K148_0N20V Rev. 22/OCT/2021
8	S32K1xx Data Sheet	Rev. 14, 08/2021



## Chapter 3

### Driver

- [Requirements](#)
- [Driver Design Summary](#)
- [Hardware Resources](#)
- [Deviations from Requirements](#)
- [Driver Limitations](#)
- [Driver usage and configuration tips](#)
- [Runtime errors](#)
- [Symbolic Names Disclaimer](#)

### 3.1 Requirements

BASE is a custom module, so AUTOSAR only specifies some guidelines for the design and configuration. Other details for this module can be found in EB tresos Studio developer's guide. This module contains stubs from several AutoSAR components. The requirements used for the files present in this module are available in the Software Specification documents from [Reference List](#) .

### 3.2 Driver Design Summary

The BASE module contains the common files/definitions needed by the MCAL. This means that it is a dependency for all other MCAL modules. The BASE module consists from a list of C header files that can be split into 3 categories:

- AutoSAR required files (that AutoSAR specifies and must not be modified)
- Stubs - files that are required by AutoSAR but are provided as examples in the NXP Semiconductor S32K1 RTD release. They must be re-written by the integrator.
- Files that are required by the NXP Semiconductor S32K1 RTD MCAL and must not be modified.

Below you can find the descriptions for each file present in the BASE module:

File Name	File Type	Description
<a href="#">Can_GeneralTypes.h</a>	Stub file. Must be replaced by all integrators.	This file is a stub. Its name and content is specified by AutoSAR but in the NXP Semiconductor S32K1 RTD release, it contains only the defines/typedefs that are needed by the RTD MCAL drivers. Note: The following files need to be included prior to include <a href="#">Can_GeneralTypes.h</a> - <a href="#">ComStack_Cfg.h</a> and <a href="#">Can_Cfg.h</a> .
<a href="#">Compiler.h</a>	AutoSAR specified file - must not be modified.	This is a file with content fully defined by the AutoSAR standard. AutoSAR requires that no modification must be done to the contents of this file. During integration this file can be overwritten with another one with the same C content. The NXP Semiconductor S32K1 RTD MCAL release provides this file and can be used as-is.
<a href="#">Compiler_Cfg.h</a>	Stub file. Must be replaced by all integrators.	This file is a stub. Its name and content is specified by AutoSAR but in the NXP Semiconductor S32K1 RTD MCAL release, it contains only the defines that are needed by the RTD MCAL drivers. This file defines the compiler memory and pointer classes to be used for RTD MCAL. The value of the defines must be set by each integrator.
<a href="#">ComStack_Cfg.h</a>	Stub file. Must be replaced by all integrators.	This file is a stub. Its name and content is specified by AutoSAR but in the NXP Semiconductor S32K1 RTD release, it contains only the defines/typedefs that are needed by the RTD MCAL drivers.
<a href="#">ComStackTypes.h</a>	Stub file. Must be replaced by all integrators.	This file is a stub. Its name and content is specified by AutoSAR but in the NXP Semiconductor S32K1 RTD release, it contains only the defines/typedefs that are needed by the RTD MCAL drivers.
<a href="#">ComStack_Types.h</a>	RTD MCAL specific file - to be used as-is. Can be replaced by integrators to ensure compatibility in stacks where the ComStack header file name was not aligned to <a href="#">ComStackTypes.h</a> .	This is a file that is specific to NXP Semiconductor S32K1 RTD MCAL release. It is a wrapper of <a href="#">ComStackTypes.h</a> to ensure compatibility of Autosar header includes.

File Name	File Type	Description
<a href="#">Eth_GeneralTypes.h</a>	Stub file. Must be replaced by all integrators.	This file is a stub. Its name and content is specified by AutoSAR but in the NXP Semiconductor S32K1 RTD release, it contains only the defines/typedefs/constants that are needed by the RTD MCAL drivers.
<a href="#">Fr_GeneralTypes.h</a>	Stub file. Must be replaced by all integrators.	This file is a stub. Its name and content is specified by AutoSAR but in the NXP Semiconductor S32K1 RTD release, it contains only the defines/typedefs/constants that are needed by the RTD MCAL drivers.
<a href="#">Lin_GeneralTypes.h</a>	Stub file. Must be replaced by all integrators.	This file is a stub. Its name and content is specified by AutoSAR but in the NXP Semiconductor S32K1 RTD release, it contains only the defines/typedefs/constants that are needed by the RTD MCAL drivers.
<a href="#">Mcal.h</a>	RTD MCAL specific file	This is a file that is specific to RTD MCAL release. It contains defines and macros needed by RTD MCAL driver. It contains several macros defined for every compiler supported by RTD MCAL (but not all compilers are available for all releases - for a list of compilers supported by this release, please check the release note document).
<a href="#">PlatformTypes.h</a>	AutoSAR specified file - must not be modified.	This is a file with content fully defined by the AutoSAR standard. AutoSAR requires that no modification must be done to the contents of this file. During integration this file can be overwritten with another one with the same C content. The NXP Semiconductor S32K1 RTD MCAL release provides this file and can be used as-is.
<a href="#">Platform_Types.h</a>	RTD MCAL specific file - to be used as-is. Can be replaced by integrators to ensure compatibility in stacks where the PlatformTypes header file name was not aligned to <a href="#">PlatformTypes.h</a> .	This is a file that is specific to NXP Semiconductor S32K1 RTD MCAL release. It is a wrapper of <a href="#">PlatformTypes.h</a> to ensure compatibility of Autosar header includes.
<a href="#">RegLockMacros.h</a>	RTD MCAL specific file - to be used as-is.	This is a file that is specific to S32K1 RTD MCAL release. It contains defines needed by RTD MCAL drivers.

File Name	File Type	Description
<a href="#">Reg_eSys.h</a>	RTD MCAL specific file - to be used as-is.	This is a file that is specific to S32K1 RTD MCAL release. It contains defines needed by RTD MCAL drivers.
Soc_Ips.h	RTD MCAL specific file - to be used as-is.	This is a file that is specific to S32K1 RTD MCAL release. It contains defines needed by RTD MCAL drivers.
<a href="#">StandardTypes.h</a>	AutoSAR specified file - must not be modified.	This is a file with content fully defined by the AutoSAR standard. AutoSAR requires that no modification must be done to the contents of this file. During integration this file can be overwritten with another one with the same C content. The NXP Semiconductor S32K1 RTD MCAL release provides this file and can be used as-is.
<a href="#">Std_Types.h</a>	RTD MCAL specific file - to be used as-is. Can be replaced by integrators to ensure compatibility in stacks where the StandardTypes header file name was not aligned to <a href="#">StandardTypes.h</a> .	This is a file that is specific to NXP Semiconductor S32K1 RTD MCAL release. It is a wrapper of <a href="#">StandardTypes.h</a> to ensure compatibility of Autosar header includes.
modules.h	RTD MCAL specific file - to be used as-is.	This is a file that is generated by Base plugin and contains defines needed by RTD MCAL drivers.

### 3.3 Hardware Resources

In baremetal or FreeRTOS mode, OsIf module will use the Cortex M SysTick counter.

### 3.4 Deviations from Requirements

Since this is a custom module, it contains files from several AutoSAR components. The AUTOSAR provides some guidelines for design and configuration the BASE Module. The BASE module deviates from the AUTOSAR software specification documents from [Reference List](#) mainly for the files provided as stubs in the current release. There are also some additional requirements (on top of requirements detailed in AUTOSAR software specification documents from [Reference List](#) which need to be satisfied for correct operation.

Term	Definition
N/S	Out of scope
N/I	Not implemented
N/F	Not fully implemented

Below table identifies the AUTOSAR requirements that are not fully implemented, implemented differently, not available, not testable or out of scope for the driver.

Requirement	Status	Description	Notes
SWS_COMPILER_00041	N/S	Each AUTOSAR software module and application software component shall wrap declaration and definition of code, variables, constants and pointer types using the following keyword macros.	Not applicable anymore for RTD package. Modern CPU architectures do not require special keywords.

Requirement	Status	Description	Notes
SWS_COMPILER_00999	N/S	These requirements are not applicable to this specification.	SRS_BSW_00300, SRS_BSW_00301, SRS_BSW_00302, SRS_BSW_00305, SRS_BSW_00307, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00310, SRS_BSW_00312, SRS_BSW_00314, SRS_BSW_00323, SRS_BSW_00325, SRS_BSW_00327, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00333, SRS_BSW_00334, SRS_BSW_00335, SRS_BSW_00336, SRS_BSW_00339, SRS_BSW_00341, SRS_BSW_00342, SRS_BSW_00343, SRS_BSW_00344, SRS_BSW_00346, SRS_BSW_00350, SRS_BSW_00353, SRS_BSW_00357, SRS_BSW_00358, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00369, SRS_BSW_00371, SRS_BSW_00373, SRS_BSW_00375, SRS_BSW_00377, SRS_BSW_00378, SRS_BSW_00380, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00394, SRS_BSW_00395, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00004, SRS_BSW_00400, SRS_BSW_00401, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00406, SRS_BSW_00407, SRS_BSW_00408, SRS_BSW_00409, SRS_BSW_00410, SRS_BSW_00411, SRS_BSW_00413, SRS_BSW_00414, SRS_BSW_00415, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00005, SRS_BSW_00007, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00158, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00164, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00171, SRS_BSW_00172.
		<b>S32K1 BASE Driver</b>	Not a requirement.

Requirement	Status	Description	Notes
SWS_Platform_00063	N/S	These requirements are not applicable to this specification.	Not a requirement
SWS_Comtype_NA_0	N/S	This specification item references requirements that are not applicable, because ComStack_↔Types neither has configurable parameters nor has reference to configuration parameters from other modules.	Not a requirement.
SWS_Comtype_NA_1	N/S	This specification item references requirements that are not applicable, because ComStack_↔Types has no interdependencies to SW Components.	Not a requirement.
SWS_Comtype_NA_2	N/S	This specification item references requirements that are not applicable, because ComStack_↔Types does not implement any interrupts, is not a driver or MCAL abstraction layer or has any direct access to OS.	Not a requirement.
SWS_Comtype_NA_3	N/S	This specification item references requirements that are not applicable, because ComStack_↔Types does not implement any version check information, main function, APIs, standard types.	Not a requirement.
SWS_Comtype_NA_4	N/S	This specification item references requirements that are not applicable, because ComStack_↔Types does not have any shutdown functionality.	Not a requirement.
SWS_Comtype_NA_5	N/S	This specification item references requirements that are not applicable, because ComStack_↔Types does not implement development errors and production errors.	Not a requirement.
SWS_MemMap_00999	N/S	These requirements are not applicable to this specification.	Not a requirement.
ECUC_MemMap_00001	N/S	Module Name - MemMap - Module Description - Configuration of the Memory Mapping and Compiler Abstraction module. - Post-Build Variant Support - false - Supported Config Variants - VA↔RIANT-PRE-COMPILE -	MemMap section is a stub, this requirement is not implemented.

Requirement	Status	Description	Notes
ECUC_MemMap_00003	N/S	Container Name - MemMap↔ AddressingMode - Description - Defines a addressing mode with a set of #pragma statements implementing the start and the stop of a section. - Configuration Parameters -	MemMap section is a stub, this requirement is not implemented.
ECUC_MemMap_00004	N/S	Name - MemMapAddressing↔ ModeStart - Parent Container - MemMap↔ AddressingMode - Description - Defines a set of #pragma statements implementing the start of a section. - Multiplicity - 1 - Type - EcucMultilineString↔ ParamDef - Default value - - - maxLength - - - minLength - - - regularExpression - - - Post-Build Variant Value - false - Value Configuration Class - Pre-compile time - X - All Variants - Link time - - - - Post-build time - - - - Scope / Dependency - scope: local -	MemMap section is a stub, this requirement is not implemented.
ECUC_MemMap_00005	N/S	Name - MemMapAddressing↔ ModeStop - Parent Container - MemMap↔ AddressingMode - Description - Defines a set of #pragma statements implementing the start of a section. - Multiplicity - 1 - Type - EcucMultilineString↔ ParamDef - Default value - - - maxLength - - - minLength - - - regularExpression - - - Post-Build Variant Value - false - Value Configuration Class - Pre-compile time - X - All Variants - Link time - - - - Post-build time - - - - Scope / Dependency - scope: local -	MemMap section is a stub, this requirement is not implemented.



Requirement	Status	Description	Notes
ECUC_MemMap_00006	N/S	<p>Name - MemMapAlignment↔  Selector -  Parent Container - MemMap↔  AddressingMode -  Description - Defines a the alignments for which the Mem↔MapAddressingMode applies.  The to be used alignment is defined in the alignment attribute of the MemorySection. If the MemMapAlignmentSelector fits to alignment attribute of the MemorySection the set of #pragmas of the related Mem↔MapAddressingMode shall be used to implement the start and the stop of a section. Please note that  the same MemMapAddressing↔Mode can be applicable for several alignments,  e.g. "8" bit and "UNSPECIFIED".  -  Multiplicity - 1..* -  Type - EcucStringParamDef -  Default value - - -  maxLength - - -  minLength - - -  regularExpression - [1-9][0-9]*[0x[0-9a-f]*[0-7]*[0b[0-1]* U↔NSPECIFIED UNKNOWN BO↔OLEAN PTR -  Post-Build Variant Multiplicity - false -  Post-Build Variant Value - false -  Multiplicity Configuration Class -  Pre-compile time - X - All Variants -  -  Link time - - - -  Post-build time - - - -  Value Configuration Class - Pre-compile time - X - All Variants -  Link time - - - -  Post-build time - - - -  Scope / Dependency - scope: local -</p>	MemMap section is a stub, this requirement is not implemented.
ECUC_MemMap_00002	N/S	<p>Container Name - MemMap↔  AddressingModeSet -  Description - Defines a set of addressing modes which might apply to a SwAddrMethod. -  Configuration Parameters -</p>	MemMap section is a stub, this requirement is not implemented.

Requirement	Status	Description	Notes
ECUC_MemMap_00018	N/S	Name - MemMapCompilerMem↔ ClassSymbolImpl - Parent Container - MemMap↔ AddressingModeSet - Description - Defines the implementation behind a MemClass↔ Symbol and configures the Compiler Abstraction. - Multiplicity - 1 - Type - EcucStringParamDef - Default value - - - maxLength - - - minLength - - - regularExpression - - - Post-Build Variant Value - false - Value Configuration Class - Pre-compile time - X - All Variants - Link time - - - - Post-build time - - - - Scope / Dependency - scope: ECU -	MemMap section is a stub, this requirement is not implemented.

Requirement	Status	Description	Notes
ECUC_MemMap_00009	N/S	<p>Name - MemMapSupported↵  AddressingMethodOption -  Parent Container - MemMap↵  AddressingModeSet -  Description - This constrains the usage of this addressing mode set for Generic Mappings to swAddrMethods.The attribute option of a swAddrMethod mapped via MemMapGenericMapping to this MemMapAddressing↵  ModeSet shall be equal to one of the configured  MemMapSupportedAddress↵  MethodOption's -  Multiplicity - 0..* -  Type - EcucStringParamDef -  Default value - - -  maxLength - - -  minLength - - -  regularExpression - [a-zA-Z]([a-z↵  A-Z0-9] _[a-zA-Z0-9])*_? -  Post-Build Variant Multiplicity - false -  Post-Build Variant Value - false -  Multiplicity Configuration Class -  Pre-compile time - X - All Variants -  Link time - - - -  Post-build time - - - -  Value Configuration Class - Pre-compile time - X - All Variants -  Link time - - - -  Post-build time - - - -  Scope / Dependency - scope: ECU -</p>	MemMap section is a stub, this requirement is not implemented.

Requirement	Status	Description	Notes
ECUC_MemMap_00017	N/S	<p>Name - MemMapSupported↵ MemoryAllocationKeywordPolicy - Parent Container - MemMap↵ AddressingModeSet - Description - This constrains the usage of this addressing mode set for Generic Mappings to swAddrMethods.The attribute MemoryAllocationKeywordPolicy of a swAddrMethod mapped via MemMapGeneric↵ Mapping to this MemMap↵ AddressingModeSet shall be equal to one of the configured MemMapSupported↵ MemoryAllocationKeyword↵ Policy's - Multiplicity - 0..* - Type - EcucEnumerationParam↵ Def - Range - MEMMAP_ALLOCAT↵ ION_KEYWORD_POLICY_↵ ADDR_METHOD_SHORT_N↵ AME - The Memory Allocation Keyword is build with the short name of the SwAddrMethod. This is the default value if the attribute does not exist in the SwAddrMethod. - MEMMAP_ALLOCATION_K↵ EYWORD_POLICY_ADDR_↵ METHOD_SHORT_NAME_A↵ ND_ALIGNMENT - The Memory Allocation Keyword is build with the the short name of the SwAddrMethod and the alignment attribute of the MemorySection. This requests a separation of objects in memory dependent from the alignment and is not applicable for RunnableEntitys and BswSchedulableEntitys. - Post-Build Variant Multiplicity - false - Post-Build Variant Value - false - Multiplicity Configuration Class - Pre-compile time - X - All Variants - Link time - - - - Post-build time - - - - Value Configuration Class - Pre-compile time - X - All Variants - Link time - - - -</p>	MemMap section is a stub, this requirement is not implemented.
18		<p>Post-build time - - - - Scope / Dependency - scope: ECU</p>	NXP Semiconductors

Requirement	Status	Description	Notes
-------------	--------	-------------	-------

Requirement	Status	Description	Notes
ECUC_MemMap_00008	N/S	<p>Name - MemMapSupported↵  SectionInitializationPolicy -  Parent Container - MemMap↵  AddressingModeSet -  Description - This constrains the usage of this addressing mode set for Generic Mappings to swAddr↵  Methods.  The sectionIntializationPolicy attribute value of a swAddrMethod mapped via MemMapGeneric↵  Mapping to this MemMap↵  AddressingModeSet shall be equal to one of  the configured MemMap↵  SupportedSectionIntialization↵  Policy's.  Please note that Section↵  InitializationPolicyType describes the  intended initialization of  MemorySections. The follow-  ing values are standardized in  AUTOSAR Methodology:NO-↵  INIT: No initialization and no  clearing is performed.  Such data elements must not be  read before one has written a value  into it.  INIT: To be used for data that are  initialized by every reset to the  specified  value (initValue).POWER-ON-I↵  NIT: To be used for data that are  initialized by "Power On"  to the specified value (initValue).  Note: there might be several resets  between power on resets.  CLEARED: To be used for data  that are initialized by every reset  to zero.  POWER-ON-CLEARED: To be  used for data that are initialized  by "Power On" to zero.  Note: there might be several resets  between power on resets.  Please note that the values are de-  fined similar to the representation  of enumeration types in the XML  schema to ensure backward com-  patibility. -  Multiplicity - 0..* -  Type - EcucStringParamDef -  Default value - - -  maxLength - - -  minLength - - -</p>	MemMap section is a stub, this re- quirement is not implemented.
20		<p><b>S32K1 BASE Driver</b></p> <p>regularExpression - - -  Post_Build_Variant_Multiplicity -</p>	NXP Semiconductors

Requirement	Status	Description	Notes
-------------	--------	-------------	-------

Requirement	Status	Description	Notes
ECUC_MemMap_00007	N/S	<p>Name - MemMapSupported↵  SectionType -  Parent Container - MemMap↵  AddressingModeSet -  Description - This constrains the usage of this addressing mode set for Generic Mappings to swAddr↵  Methods.  The attribute sectionType of a swAddrMethod mapped via MemMapGenericMapping or MemMapSectionSpecificMapping to this MemMapAddressing↵  ModeSet shall be equal to one of the configured Mem↵  MapSupportedSectionType's. -  Multiplicity - 0..* -  Type - EcucEnumerationParam↵  Def -  Range - MEMMAP_SECTIO↵  N_TYPE_CAL_PRM - To be used for calibratable constants of ECU-functions. -  MEMMAP_SECTION_TYPE↵  _CODE - To be used for mapping code to application block, boot block, external flash etc. -  MEMMAP_SECTION_TYP↵  E_CONFIG_DATA - Constants with attributes that show that they reside in one segment for module configuration. -  MEMMAP_SECTION_TYPE↵  _CONST - To be used for global or static constants. -  MEMMAP_SECTION_TYPE↵  _EXCLUDE_FROM_FLASH - Values existing in the ECU but not dropped down in the binary file. No upload should be needed to obtain access to the ECU data. The ECU will never be touched by the instrumentation tool, with the exception of upload. These are memory areas which are not overwritten by downloading the executable. -  MEMMAP_SECTION_TYP↵  E_VAR - To be used for global or static variables. The expected initialization is specified with the attribute sectionInitialization↵  Policy. -  Post-Build Variant Multiplicity - false -</p> <p><b>S32K1 BASE Driver</b></p>	MemMap section is a stub, this requirement is not implemented.
22		<p>Post-Build Variant Value - false -  Multiplicity Configuration Class -</p>	NXP Semiconductors



Requirement	Status	Description	Notes
ECUC_MemMap_00010	N/S	<p>Container Name - MemMap↔Allocation -</p> <p>Description - Defines which MemorySection of a BSW Module or a Software Component is implemented with which Mem↔MapAddressingModeSet. This can either be specified for a set of MemorySections which refer to an identical SwAddrMethod (Mem↔MapGenericMapping) or for individual MemorySections (MemMapSectionSpecific↔Mapping). If both are defined for the same MemorySection the MemMapSectionSpecific↔Mapping overrules the MemMap↔GenericMapping. -</p> <p>Configuration Parameters -</p>	MemMap section is a stub, this requirement is not implemented.
ECUC_MemMap_00019	N/S	<p>Container Name - MemMap↔GenericCompilerMemClass -</p> <p>Description - The shortName of the container defines the name of the generic Compiler memclass which is global for all using modules, e.g. REGS↔PACE.</p> <p>The configures the Compiler Abstraction. -</p> <p>Configuration Parameters -</p>	MemMap section is a stub, this requirement is not implemented.
ECUC_MemMap_00020	N/S	<p>Name - MemMapGeneric↔CompilerMemClassSymbolImpl -</p> <p>Parent Container - MemMap↔GenericCompilerMemClass -</p> <p>Description - Defines the implementation behind the generic MemClassSymbol and configures the Compiler Abstraction. -</p> <p>Multiplicity - 1 -</p> <p>Type - EcucStringParamDef -</p> <p>Default value - - -</p> <p>maxLength - - -</p> <p>minLength - - -</p> <p>regularExpression - - -</p> <p>Post-Build Variant Value - false -</p> <p>Value Configuration Class - Pre-compile time - X - All Variants -</p> <p>Link time - - - -</p> <p>Post-build time - - - -</p> <p>Scope / Dependency - scope: ECU -</p>	MemMap section is a stub, this requirement is not implemented.

Requirement	Status	Description	Notes
ECUC_MemMap_00011	N/S	<p>Container Name - MemMap↔GenericMapping -</p> <p>Description - Defines which Sw↔AddrMethod is implemented with which MemMapAddressingMode↔Set.</p> <p>The pragmas for the implementation of the MemorySelector↔Keywords are taken from the MemMapAddressingModeStart and MemMapAddressingMode↔Stop parameters of the Mem↔MapAddressingModeSet for the individual alignments.↔</p> <p>That this mapping becomes valid requires matching MemMapSupportedSectionType's, MemMapSupportedSection↔InitializationPolicy's and MemMapSupportedAddressing↔MethodOption's. The MemMap↔GenericMapping applies only if it is not overruled by an MemMap↔SectionSpecificMapping - Configuration Parameters -</p>	MemMap section is a stub, this requirement is not implemented.
ECUC_MemMap_00012	N/S	<p>Name - MemMapAddressing↔ModeSetRef -</p> <p>Parent Container - MemMap↔GenericMapping -</p> <p>Description - Reference to the MemMapAddressingModeSet which applies to the MemMap↔GenericMapping. -</p> <p>Multiplicity - 1 -</p> <p>Type - Reference to [ MemMap↔AddressingModeSet ] -</p> <p>Post-Build Variant Value - false -</p> <p>Value Configuration Class - Pre-compile time - X - All Variants -</p> <p>Link time - - - -</p> <p>Post-build time - - - -</p> <p>Scope / Dependency - scope: ECU -</p>	MemMap section is a stub, this requirement is not implemented.

Requirement	Status	Description	Notes
ECUC_MemMap_00013	N/S	<p>Name - MemMapSwAddress↔ MethodRef - Parent Container - MemMap↔ GenericMapping -</p> <p>Multiplicity - 1 - Type - Foreign reference to [ SW-ADDR-METHOD ] - Post-Build Variant Value - false - Value Configuration Class - Pre-compile time - X - All Variants - Link time - - - -</p> <p>Scope / Dependency - scope: ECU -</p>	MemMap section is a stub, this requirement is not implemented.
ECUC_MemMap_00014	N/S	<p>Container Name - MemMap↔ SectionSpecificMapping - Description - Defines which MemorySection of a BSW Module or a Software Component is implemented with which Mem↔ MapAddressingModeSet. The pragmas for the implementation of the MemorySelectorKeywords are taken from the MemMap↔ AddressingModeStart and MemMapAddressingMode↔ Stop parameters of the Mem↔ MapAddressingModeSet for the specific alignment of the Memory↔ Section. The MemMapSection↔ SpecificMapping precedes a mapping defined by MemMapGenericMapping. - Configuration Parameters -</p>	MemMap section is a stub, this requirement is not implemented.

Requirement	Status	Description	Notes
ECUC_MemMap_00015	N/S	Name - MemMapAddressing↔ ModeSetRef - Parent Container - MemMap↔ SectionSpecificMapping - Description - Reference to the MemMapAddressingModeSet which applies to the MemMap↔ ModuleSectionSpecificMapping. - Multiplicity - 1 - Type - Reference to [ MemMap↔ AddressingModeSet ] - Post-Build Variant Value - false - Value Configuration Class - Pre- compile time - X - All Variants - Link time - - - - Post-build time - - - - Scope / Dependency - scope: ECU -	MemMap section is a stub, this re- quirement is not implemented.
ECUC_MemMap_00016	N/S	Name - MemMapMemorySection↔ Ref - Parent Container - MemMap↔ SectionSpecificMapping - Description - Reference to the MemorySection which applies to the MemMapSectionSpecific↔ Mapping. - Multiplicity - 1 - Type - Foreign reference to [ MEMORY-SECTION ] - Post-Build Variant Value - false - Value Configuration Class - Pre- compile time - X - All Variants - Link time - - - -  Scope / Dependency - scope: ECU -	MemMap section is a stub, this re- quirement is not implemented.
SWS_Std_00999	N/S	These requirements are not appli- cable to this specification.	These requirements are not appli- cable to this specification.
SWS_MemMap_00040	N/S	When a BSW module or Software Component is split into allocatable memory parts the <PREFIX> as described in S↔ WS_MemMap_00022 shall be sub-structured in the following way:<PREFIX> = <snp>[_<vi>_<ai>]_<feature>	Will be analyzed after Autosar clarifications (ARTD-8691).

Requirement	Status	Description	Notes
SWS_MemMap_00041	N/S	When a BSW module or Software Component is split into allocatable memory parts the resulting <PREFIX> as specified in SWS_MemMap_00040 (inclusive [_<vi>_<ai>]) shall be described as a SectionNamePrefix and all belonging MemorySections.MemorySection. prefix needs to reference the SectionNamePrefix.	Will be analyzed after Autosar clarifications (ARTD-8691).
ECUC_MemMap_00023	N/S	Name - MemMapMappingSelectorRef - Parent Container - MemMapGenericMapping - Description - Reference to a MemMapPrefixSelector. The owning MemMapGenericMapping is only effective for those memories where the MemMapMappingSelector matches. - Multiplicity - 0..1 - Type - Reference to [ MemMapMappingSelector ] - Post-Build Variant Value - false - Value Configuration Class - Pre-compile time - X - All Variants - Link time - - - - Post-build time - - - - Scope / Dependency - scope: ECU -	Does not impact to Base driver.
ECUC_MemMap_00021	N/S	Container Name - MemMapMappingSelector - Description - The container holds a section criteria reusable for MemMapGenericMappings. - Configuration Parameters -	Does not impact to Base driver.

Requirement	Status	Description	Notes
ECUC_MemMap_00022	N/S	<p>Name - MemMapPrefixSelector -  Parent Container - MemMap↔  MappingSelector -  Description - The parameter MemMapPrefixSelector defines a regular expression which shall be applied to the part of the memory allocation keywords.  The mapping using this selector is only effective for those memories where the part of the memory allocation keyword matches the regular expression.  Note: This is in particular intended the restrict the usage of of a MemMapAddressingModeSet for a sub set of BSW Modules or Software Components or a subset of allocatable memory parts inside BSW Modules or Software Components. -  Multiplicity - 0..1 -  Type - EcucStringParamDef -  Default value - - -  maxLength - - -  minLength - - -</p> <p>Post-Build Variant Value - false -  Value Configuration Class - Pre-compile time - X - All Variants -  Link time - - - -  Post-build time - - - -  Scope / Dependency - scope: ECU -</p>	Does not impact to Base driver.

## 3.5 Driver Limitations

None.

## 3.6 Driver usage and configuration tips

**3.6.1 NO\_STDINT\_H compiler symbol** By default, [PlatformTypes.h](#) defines its types (e.g. uint8) based on `stdint.h` (e.g. `uint8_t`). AUTOSAR integrators can replace this implementation with their own. The platform header file needs `stdint`-like types for its definition, so a 'glue layer' header is defined in [BasicTypes.h](#).

This either includes `stdint.h` or, if the compiler symbol `NO_STDINT_H` is provided, `stdint.h` is not included, and `stdint`-like types (`uint8_t`) are defined over [PlatformTypes.h](#) (`uint8`).

The behavior in [BasicTypes.h](#) is as follows:

- if `stdint.h` is included in [PlatformTypes.h](#), [BasicTypes.h](#) has no effect
- if `stdint.h` is not included in [PlatformTypes.h](#) and `NO_STDINT_H` is NOT defined, [BasicTypes.h](#) includes `stdint.h`
- if `stdint.h` is not included in [PlatformTypes.h](#) and `NO_STDINT_H` IS defined, [BasicTypes.h](#) defines `stdint`-like types

Care must be taken when `NO_STDINT_H` is used, if other software libraries include `stdint.h` or define similar types, as this can cause double definitions of types and compiler warnings.

### 3.7 Runtime errors

The module does not generate any DEM errors at runtime.

Function	Error Code	Condition triggering the error
N/A	N/A	N/A

### 3.8 Symbolic Names Disclaimer

All containers having `symbolicNameValue` set to `TRUE` in the AUTOSAR schema will generate defines like:

```
#define <Mip>Conf_<Container_ShortName>_<Container_ID>
```

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

## Chapter 4

### Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the driver. All the parameters are described below.

- Module [Base](#)
  - Container [OsIfGeneral](#)
    - \* Parameter [OsIfMulticoreSupport](#)
    - \* Parameter [OsIfEnableUserModeSupport](#)
    - \* Parameter [OsIfDevErrorDetect](#)
    - \* Parameter [OsIfUseSystemTimer](#)
    - \* Parameter [OsIfUseCustomTimer](#)
    - \* Parameter [OsIfInstanceId](#)
    - \* Reference [OsIfEcucPartitionRef](#)
    - \* Container [OsIfOperatingSystemType](#)
    - \* Container [OsIfCounterConfig](#)
      - Reference [OsIfCounterEcucPartitionRef](#)
      - Reference [OsIfSystemTimerClockRef](#)
      - Reference [OsIfOsCounterRef](#)
  - Container [CommonPublishedInformation](#)
    - \* Parameter [ArReleaseMajorVersion](#)
    - \* Parameter [ArReleaseMinorVersion](#)
    - \* Parameter [ArReleaseRevisionVersion](#)
    - \* Parameter [ModuleId](#)
    - \* Parameter [SwMajorVersion](#)
    - \* Parameter [SwMinorVersion](#)
    - \* Parameter [SwPatchVersion](#)
    - \* Parameter [VendorApiInfix](#)
    - \* Parameter [VendorId](#)

#### 4.1 Module Base

Configuration of Base module.

Included containers:

- [OsIfGeneral](#)
- [CommonPublishedInformation](#)



Property	Value
type	ECUC-MODULE-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantSupport	false
supportedConfigVariants	VARIANT-PRE-COMPILE

## 4.2 Container OsIfGeneral

This container contains the configuration parameters for the OS Interface.

Included subcontainers:

- [OsIfOperatingSystemType](#)
- [OsIfCounterConfig](#)

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

## 4.3 Parameter OsIfMulticoreSupport

Switches global multicore support on or off:

False: For all variants, no EcucPartition shall be referenced in OsIfEcucPartitionRef.

True: For all variants, at least one EcucPartition needs to be referenced in OsIfEcucPartitionRef.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

## 4.4 Parameter OsIfEnableUserModeSupport

When this parameter is enabled, the OsIf module will adapt to run from User Mode, with the following measures:

(if applicable) a) configuring REG\_PROT for the Eth Controllers so that the registers under protection eth be accessed from user mode by setting UAA bit in REG\_PROT\_GCR to 1

(if applicable) b) using 'call trusted function' stubs for all internal function calls that access registers requiring supervisor mode.

(if applicable) c) other module specific measures for more information, please see chapter 5.7 User Mode Support in IM

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

## 4.5 Parameter OsIfDevErrorDetect

Switches the development error detection and notification on or off.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

## 4.6 Parameter OsIfUseSystemTimer

Switches the system timer usage on or off. The system timer is architecture-specific and may not exist.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

## 4.7 Parameter OsIfUseCustomTimer

Switches the custom timer usage on or off.  
will have to provide the functions expected by the OSIF API.

When this feature is enabled, the application

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

## 4.8 Parameter OsIfInstanceId

Instance ID of the OsIf driver. This ID is used to discern several OsIf drivers in case more than one driver is used in the same ECU.

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	255
min	0

## 4.9 Reference OsIfEcucPartitionRef

Maps the OsIf module to zero or multiple ECUC partitions to make the module's API

available in this partition. The OsIf module will operate as an independent

instance in each of the partitions.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

## 4.10 Container OsIfOperatingSystemType

This container contains the configuration parameters for the OS Interface.

Included choices:

- OsIfAutosarOsType

- OsIfFreeRtosType
- OsIfZephyrOsType
- OsIfBaremetalType

Property	Value
type	ECUC-CHOICE-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

## 4.11 Container OsIfCounterConfig

Configures counters used by OsIf.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

## 4.12 Reference OsIfCounterEcucPartitionRef

Maps the OsIf Counter to zero or one ECUC partition.

The ECUC partition referenced is a subset of the ECUC partitions where the OsIf module is mapped to.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

Property	Value
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcuDefs/EcuC/EcuPartitionCollection/EcuPartition

## 4.13 Reference OsIfSystemTimerClockRef

Reference to the system timer clock source configuration, which is set in the MCU module configuration.

Cortex-M: The clock source of the system timer is usually the clock source of the CPU on which the application will run.

Cortex-A/R: The clock source of the system timer is usually a divided clock of FXOSC\_CLK. The divide value is system specific, usually residing in a register of the GPR hardware module.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcuDefs/Mcu/McuModuleConfiguration/McuClockSetting↔ Config/McuClockReferencePoint

## 4.14 Reference OsIfOsCounterRef

A reference to an OS Counter.

Parameter OsSecondsPerTick of the referenced OS Counter must have multiplicity = 1.

Limitation: The referenced OS Counter will be used by all drivers assigned to the same partition referenced by OsIfCounterEcuPartitionRef.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcuDefs/Os/OsCounter

## 4.15 Container CommonPublishedInformation

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

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

## 4.16 Parameter ArReleaseMajorVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	4
max	4
min	4

## 4.17 Parameter ArReleaseMinorVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	4
max	4
min	4

## 4.18 Parameter ArReleaseRevisionVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	0
min	0

## 4.19 Parameter ModuleId

Module ID of this module from Module List.



Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	0
max	0
min	0

## 4.20 Parameter SwMajorVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	1
max	1
min	1

## 4.21 Parameter SwMinorVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	0
min	0

## 4.22 Parameter SwPatchVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	1
min	1

## 4.23 Parameter VendorApiInfix

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>.

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.

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	

## 4.24 Parameter VendorId

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	43
max	43
min	43



# Chapter 5

## Module Index

### 5.1 Software Specification

Here is a list of all modules:

OsIf . . . . .	43
BASE_COMPONENT . . . . .	47

## Chapter 6

### Module Documentation

#### 6.1 OsIf

##### 6.1.1 Detailed Description OsIf module (Os Interface)

This module provides basic timing/Os services for drivers, allowing for Os independent implementations.

##### Enum Reference

- enum [OsIf\\_CounterType](#)  
*OsIf Counter type.*

##### Function Reference

- void [OsIf\\_Init](#) (const void \*Config)  
*Initialize OsIf.*
- uint32 [OsIf\\_GetCounter](#) ([OsIf\\_CounterType](#) SelectedCounter)  
*Get the current value counter.*
- uint32 [OsIf\\_GetElapsed](#) (uint32 \*const CurrentRef, [OsIf\\_CounterType](#) SelectedCounter)  
*Get the elapsed value from a reference point.*
- void [OsIf\\_SetTimerFrequency](#) (uint32 Freq, [OsIf\\_CounterType](#) SelectedCounter)  
*Set a new frequency used for time conversion (microseconds to ticks)*
- uint32 [OsIf\\_MicrosToTicks](#) (uint32 Micros, [OsIf\\_CounterType](#) SelectedCounter)  
*Convert microseconds to ticks.*

##### 6.1.2 Enum Reference

###### 6.1.2.1 OsIf\_CounterType

enum [OsIf\\_CounterType](#)

OsIf Counter type.

Counter type.

Enumerator

OSIF_COUNTER_DUMMY	dummy counter
--------------------	---------------

Definition at line 111 of file OsIf.h.

### 6.1.3 Function Reference

#### 6.1.3.1 OsIf\_Init()

```
void OsIf_Init (
    const void * Config )
```

Initialize OsIf.

This function initializes the OsIf module and should be called during startup, before every other initialization other than Mcu.

#### 6.1.3.2 OsIf\_GetCounter()

```
uint32 OsIf_GetCounter (
    OsIf_CounterType SelectedCounter )
```

Get the current value counter.

This function returns the current value of the counter.

Parameters

in	<i>SelectedCounter</i>	the type of counter to use
----	------------------------	----------------------------

Returns

the current value of the counter

#### 6.1.3.3 OsIf\_GetElapsed()

```
uint32 OsIf_GetElapsed (
    uint32 *const CurrentRef,
    OsIf_CounterType SelectedCounter )
```

Get the elapsed value from a reference point.

This function returns the delta time in ticks compared to a reference, and updates the reference.

Parameters

in, out	<i>CurrentRef</i>	reference counter value, updated to current counter value
in	<i>SelectedCounter</i>	the type of counter to use

Returns

the elapsed time

#### 6.1.3.4 OsIf\_SetTimerFrequency()

```
void OsIf_SetTimerFrequency (
    uint32 Freq,
    OsIf_CounterType SelectedCounter )
```

Set a new frequency used for time conversion (microseconds to ticks)

This function stores a new timer frequency used for time conversion computations

Parameters

in	<i>Freq</i>	the new frequency
in	<i>SelectedCounter</i>	the type of counter to use

#### 6.1.3.5 OsIf\_MicrosToTicks()

```
uint32 OsIf_MicrosToTicks (
    uint32 Micros,
    OsIf_CounterType SelectedCounter )
```

Convert microseconds to ticks.

This function converts a value from microsecond units to ticks units.

Parameters

in	<i>Micros</i>	microseconds value
in	<i>SelectedCounter</i>	the type of counter to use

## Module Documentation

### Returns

the equivalent value in ticks



## 6.2 BASE\_COMPONENT

### 6.2.1 Detailed Description

#### Data Structures

- struct [Can\\_PduType](#)  
*[Can\\_PduType](#). [More...](#)*
- struct [Can\\_HwType](#)  
*[Can\\_HwType](#). [More...](#)*
- struct [PduInfoType](#)  
*Variables of this type are used to store the basic information about a PDU of any type, namely a pointer variable pointing to it's SDU (payload), and the corresponding length of the SDU in bytes. [More...](#)*
- struct [RetryInfoType](#)  
*Variables of this type shall be used to store the information about Tp buffer handling. [More...](#)*
- struct [Eth\\_TimeStampType](#)  
*Type used to express the timestamp value. [More...](#)*
- struct [Eth\\_TimeIntDiffType](#)  
*Type used to express the diff between timestamp values. [More...](#)*
- struct [Eth\\_RateRatioType](#)  
*Type used to express frequency ratio. [More...](#)*
- struct [Eth\\_CounterType](#)  
*Type used to statistic counter for diagnostics. [More...](#)*
- struct [Eth\\_RxStatsType](#)  
*Type used to statistic counter for diagnostics. [More...](#)*
- struct [Eth\\_TxStatsType](#)  
*Type used to statistic counter for diagnostics. [More...](#)*
- struct [Eth\\_TxErrorCounterValuesType](#)  
*Type used to statistic counter for diagnostics. [More...](#)*
- struct [Eth\\_MacVlanType](#)  
*Type used for VLAN management in EthSwt. [More...](#)*
- struct [EthSwt\\_MgmtInfoType](#)  
*Type for holding the management information received/transmitted on Switches (ports). [More...](#)*
- struct [EthSwt\\_PortMirrorCfgType](#)  
*The [EthSwt\\_PortMirrorCfgType](#) specify the port mirror configuration which is set up per Ethernet switch. The configuration is written to the Ethernet switch driver by calling [EthSwt\\_WritePortMirrorConfiguration](#). One port mirror configuration is maintained per Ethernet Switch. [More...](#)*
- struct [EthSwt\\_MgmtObjectValidType](#)  
*Will be set from EthSwt and marks [EthSwt\\_MgmtObject](#) as valid or not. So the upper layer will be able to detect inconsistencies. [More...](#)*
- struct [EthSwt\\_MgmtObjectType](#)  
*Provides information about all struct member elements. The ownership gives information whether EthSwt has finished its activities in providing all struct member elements. [More...](#)*
- struct [Fr\\_POCTestStatusType](#)  
*Variables of this type are used to query the flexRay controller status. [More...](#)*
- struct [Fr\\_SlotAssignmentType](#)

Variables of this type are used to store information of frame indentified by `Fr_LPduIdx`. [More...](#)

- struct [Lin\\_PduType](#)

The LIN identifier (0..0x3F) with its parity bits. [More...](#)

- struct [Mcal\\_DemErrorType](#)

Typedef for DEM error management implemented by MCAL drivers. [More...](#)

- struct [Std\\_VersionInfoType](#)

This type shall be used to request the version of a BSW module using the "ModuleName"\_GetVersionInfo() function. [More...](#)

## Macros

- `#define` [MEMMAP\\_ERROR](#)

Symbol used for checking correctness of the includes.

- `#define` [MEMMAP\\_ERROR](#)

Symbol used for checking correctness of the includes.

- `#define` [MEMMAP\\_ERROR](#)

Symbol used for checking correctness of the includes.

- `#define` [MEMMAP\\_ERROR](#)

Symbol used for checking correctness of the includes.

- `#define` [CAN\\_BUSY](#)

Transmit request could not be processed because no transmit object was available.

- `#define` [MEMMAP\\_ERROR](#)

Symbol used for checking correctness of the includes.

- `#define` [MEMMAP\\_ERROR](#)

Symbol used for checking correctness of the includes.

- `#define` [AUTOMATIC](#)

The memory class `AUTOMATIC` shall be provided as empty definition, used for the declaration of local pointers.

- `#define` [TYPEDEF](#)

The memory class `TYPEDEF` shall be provided as empty definition. This memory class shall be used within type definitions, where no memory qualifier can be specified. This can be necessary for defining pointer types, with e.g. `P2VAR`, where the macros require two parameters. First parameter can be specified in the type definition (distance to the memory location referenced by the pointer), but the second one (memory allocation of the pointer itself) cannot be defined at this time. Hence memory class `TYPEDEF` shall be applied.

- `#define` [NULL\\_PTR](#)

The compiler abstraction shall provide the `NULL_PTR` define with a void pointer to zero definition.

- `#define` [FUNC](#)(rettype, memclass)

The compiler abstraction shall define the `FUNC` macro for the declaration and definition of functions, that ensures correct syntax of function declarations as required by a specific compiler.

- `#define` [P2VAR](#)(ptrtype, memclass, ptrclass)

The compiler abstraction shall define the `P2VAR` macro for the declaration and definition of pointers in RAM, pointing to variables.

- `#define` [P2CONST](#)(ptrtype, memclass, ptrclass)

The compiler abstraction shall define the `P2CONST` macro for the declaration and definition of pointers in RAM pointing to constants.

- `#define` [CONSTP2VAR](#)(ptrtype, memclass, ptrclass)

The compiler abstraction shall define the `CONSTP2VAR` macro for the declaration and definition of constant pointers accessing variables.

- `#define` [CONSTP2CONST](#)(ptrtype, memclass, ptrclass)

The compiler abstraction shall define the `CONSTP2CONST` macro for the declaration and definition of constant pointers accessing constants.

- #define `P2FUNC`(rettype, ptrclass, fctname)

The compiler abstraction shall define the `P2FUNC` macro for the type definition of pointers to functions.

- #define `CONST`(consttype, memclass)

The compiler abstraction shall define the `CONST` macro for the declaration and definition of constants.

- #define `VAR`(vartype, memclass)

The compiler abstraction shall define the `VAR` macro for the declaration and definition of variables.

- #define `CONSTP2FUNC`(rettype, ptrclass, fctname)

The compiler abstraction for const pointer to function.

- #define `FUNC_P2CONST`(rettype, ptrclass, memclass)

The compiler abstraction shall define the `FUNC_P2CONST` macro for the declaration and definition of functions returning a pointer to a constant.

- #define `FUNC_P2VAR`(rettype, ptrclass, memclass)

The compiler abstraction shall define the `FUNC_P2VAR` macro for the declaration and definition of functions returning a pointer to a variable.

- #define `AUTOSAR_COMSTACKDATA`

Define for ComStack Data.

- #define `COMTYPE_VENDOR_ID`

Parameters that shall be published within the standard types header file and also in the module's description file.

- #define `NTFRSLT_OK`

Action has been successfully finished.

- #define `NTFRSLT_E_NOT_OK`

Message not successfully received or sent out.

- #define `NTFRSLT_E_TIMEOUT_A`

Timer `N_Ar/N_As` has passed its time-out value `N_Asmax/N_Armax`.

- #define `NTFRSLT_E_TIMEOUT_BS`

Timer `N_Bs` has passed its time-out value `N_Bsmax`.

- #define `NTFRSLT_E_TIMEOUT_CR`

Timer `N_Cr` has passed its time-out value `N_Crmax`.

- #define `NTFRSLT_E_WRONG_SN`

Unexpected sequence number (PCI.SN) value received.

- #define `NTFRSLT_E_INVALID_FS`

Invalid or unknown FlowStatus value has been received.

- #define `NTFRSLT_E_UNEXP_PDU`

Unexpected protocol data unit received.

- #define `NTFRSLT_E_WFT_OVRN`

Flow control WAIT frame that exceeds the maximum counter `N_WFTmax` received.

- #define `NTFRSLT_E_ABORT`

Flow control (FC) `N_PDU` with FlowStatus = `OVFLW` received.

- #define `NTFRSLT_E_NO_BUFFER`

Indicates an abort of a transmission.

- #define `NTFRSLT_E_CANCELTION_OK`

Requested cancellation has been executed.

- #define `NTFRSLT_E_CANCELTION_NOT_OK`

Request cancellation has not been executed Due to an internal error the requested cancelation has not been executed. This will happen e.g. if the to be canceled transmission has been executed already.

- #define `NTFRSLT_PARAMETER_OK`



- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MCAL_DATA_SYNC_BARRIER()`  
*Data Synchronization Barrier (DSB) completes when all instructions before this instruction complete.*
- `#define MCAL_INSTRUCTION_SYNC_BARRIER()`  
*flushes the pipeline in the processor, so that all instructions following the ISB are fetched from cache or memory, after the ISB has been completed.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define PLATFORM_TYPES_VENDOR_ID`
- `#define CPU_TYPE_8`  
*8bit Type Processor*
- `#define CPU_TYPE_16`  
*16bit Type Processor*
- `#define CPU_TYPE_32`  
*32bit Type Processor*
- `#define CPU_TYPE_64`  
*64bit Type Processor*
- `#define MSB_FIRST`  
*MSB First Processor.*
- `#define LSB_FIRST`  
*LSB First Processor.*
- `#define HIGH_BYTE_FIRST`  
*HIGH\_BYTE\_FIRST Processor.*
- `#define LOW_BYTE_FIRST`  
*LOW\_BYTE\_FIRST Processor.*
- `#define CPU_TYPE`  
*Processor type.*
- `#define CPU_BIT_ORDER`  
*Bit order on register level.*
- `#define CPU_BYTE_ORDER`

The byte order on memory level shall be indicated in the platform types header file using the symbol `CPU_BYTE_ORDER`.

- `#define TRUE`  
Boolean true value.
- `#define FALSE`  
Boolean false value.
- `#define MEMMAP_ERROR`  
Symbol used for checking correctness of the includes.
- `#define MEMMAP_ERROR`  
Symbol used for checking correctness of the includes.
- `#define MEMMAP_ERROR`  
Symbol used for checking correctness of the includes.
- `#define MEMMAP_ERROR`  
Symbol used for checking correctness of the includes.
- `#define MCAL_AXBS_REG_PROT_AVAILABLE`  
Macros defined for the IPVs that are protected.
- `#define AXBS_PROT_MEM_U32`  
Macros defined for the protection size.
- `#define RLM_REG_WRITE8(address, value)`  
8 bits memory write macro
- `#define RLM_REG_WRITE16(address, value)`  
16 bits memory write macro.
- `#define RLM_REG_WRITE32(address, value)`  
32 bits memory write macro.
- `#define RLM_REG_READ8(address)`  
8 bits memory read macro.
- `#define RLM_REG_READ16(address)`  
16 bits memory read macro.
- `#define RLM_REG_READ32(address)`  
32 bits memory read macro.
- `#define RLM_REG_BIT_CLEAR8(address, mask)`  
8 bits bits clearing macro.
- `#define RLM_REG_BIT_CLEAR16(address, mask)`  
16 bits bits clearing macro.
- `#define RLM_REG_BIT_CLEAR32(address, mask)`  
32 bits bits clearing macro.
- `#define RLM_REG_BIT_GET8(address, mask)`  
8 bits bits getting macro.
- `#define RLM_REG_BIT_GET16(address, mask)`  
16 bits bits getting macro.
- `#define RLM_REG_BIT_GET32(address, mask)`  
32 bits bits getting macro.
- `#define RLM_REG_BIT_SET8(address, mask)`  
8 bits bits setting macro.
- `#define RLM_REG_BIT_SET16(address, mask)`  
16 bits bits setting macro.
- `#define RLM_REG_BIT_SET32(address, mask)`

- 32 bits bits setting macro.
- #define [RLM\\_REG\\_RMW8](#)(address, mask, value)  
8 bit clear bits and set with new value
- #define [RLM\\_REG\\_RMW16](#)(address, mask, value)  
16 bit clear bits and set with new value
- #define [RLM\\_REG\\_RMW32](#)(address, mask, value)  
32 bit clear bits and set with new value
- #define [SLBR\\_SET\\_BIT\\_8BIT\\_REG\\_MASK\\_U8](#)  
Mask for setting SLB bit(s) in a SLBR register (for 8/16/32bit registers)
- #define [SLBR\\_CLR\\_BIT\\_8BIT\\_REG\\_MASK\\_U8](#)  
Mask for clearing WE bit(s) in a SLBR register (for 8/16/32bit registers)
- #define [SLBR\\_GET\\_BIT\\_8BIT\\_REG\\_MASK\\_U8](#)  
Mask for getting SLB bit(s) in a SLBR register (for 8/16/32bit registers)
- #define [SLBR\\_XOR\\_8BIT\\_REG\\_MASK\\_U8](#)  
Masks for inverting bit positions in a SLBR register.
- #define [MODULO\\_4\\_BIT\\_MASK\\_U32](#)  
Mask used for getting the alignment of an address inside a 32 bit word.
- #define [MIRRORED\\_ADDR\\_OFFSET\\_U32](#)  
Offset to REG\_PROT mirrored registers area of an IP module.
- #define [SLBR\\_ADDR\\_OFFSET\\_U32](#)  
Offset to baseAddress of the SLBR registers area of an IP module.
- #define [SLBR\\_ADDR32](#)(baseAddr, regAddr, prot\_mem)  
Macro for getting the address of a lockable register's corresponding SLBR register.
- #define [GCR\\_OFFSET\\_U32](#)  
Offset to baseAddress of the REG\_PROT GCR register of an IP module.
- #define [REGPROT\\_GCR\\_HLB\\_MASK\\_U32](#)  
REG\_PROT GCR bit masks.
- #define [REGPROT\\_GCR\\_HLB\\_POS\\_U32](#)  
REG\_PROT GCR bit positions.
- #define [REG\\_SET\\_SOFT\\_LOCK8](#)(baseAddr, regAddr, prot\_mem)  
Soft locks a register by setting it's corresponding soft lock bit.
- #define [REG\\_CLR\\_SOFT\\_LOCK8](#)(baseAddr, regAddr, prot\_mem)  
Soft unlocks a register by clearing it's corresponding soft lock bit.
- #define [REG\\_GET\\_SOFT\\_LOCK8](#)(baseAddr, regAddr, prot\_mem)  
Reads the status of the soft lock bit of a register.
- #define [REG\\_BIT\\_SET\\_LOCK8](#)(baseAddr, regAddr, prot\_mem, mask)  
Sets one bit in a 8 bit register and locks the register automatically.
- #define [REG\\_BIT\\_SET\\_LOCK16](#)(baseAddr, regAddr, prot\_mem, mask)  
Sets one bit in a 16 bit register and locks the register automatically.
- #define [REG\\_BIT\\_SET\\_LOCK32](#)(baseAddr, regAddr, prot\_mem, mask)  
Sets one bit in a 32 bit register and locks the register automatically.
- #define [REG\\_BIT\\_CLEAR\\_LOCK8](#)(baseAddr, regAddr, prot\_mem, mask)  
Clears one bit in a 8 bit register and locks the register automatically.
- #define [REG\\_BIT\\_CLEAR\\_LOCK16](#)(baseAddr, regAddr, prot\_mem, mask)  
Clears one bit in a 16 bit register and locks the register automatically.
- #define [REG\\_BIT\\_CLEAR\\_LOCK32](#)(baseAddr, regAddr, prot\_mem, mask)  
Clears one bit in a 32 bit register and locks the register automatically.

- `#define REG_WRITE_LOCK8(baseAddr, regAddr, prot_mem, value)`  
*Writes the content of a 8 bit register and locks it automatically.*
- `#define REG_WRITE_LOCK16(baseAddr, regAddr, prot_mem, value)`  
*Writes the content of a 16 bit register and locks it automatically.*
- `#define REG_WRITE_LOCK32(baseAddr, regAddr, prot_mem, value)`  
*Writes the content of a 32 bit register and locks it automatically.*
- `#define REG_RMW_LOCK8(baseAddr, regAddr, prot_mem, mask, value)`  
*Clears the content of a 8 bit register, writes it with the value in 'value' parameter masked with the one in 'mask' parameter and locks it automatically.*
- `#define REG_RMW_LOCK16(baseAddr, regAddr, prot_mem, mask, value)`  
*Clears the content of a 16 bit register, writes it with the value in 'value' parameter masked with the one in 'mask' parameter and locks it automatically.*
- `#define REG_RMW_LOCK32(baseAddr, regAddr, prot_mem, mask, value)`  
*Clears the content of a 32 bit register, writes it with the value in 'value' parameter masked with the one in 'mask' parameter and locks it automatically.*
- `#define SET_HARD_LOCK(baseAddr, prot_mem)`  
*Sets the hardlock bit of an IP module.*
- `#define GET_HARD_LOCK(baseAddr, prot_mem)`  
*Reads the Hard Lock bit of an IP module.*
- `#define SET_USER_ACCESS_ALLOWED(baseAddr, prot_mem)`  
*Sets the User Access Allowed bit of an IP module.*
- `#define CLR_USER_ACCESS_ALLOWED(baseAddr, prot_mem)`  
*Clears the User Access Allowed bit of an IP module.*
- `#define GET_USER_ACCESS_ALLOWED(baseAddr, prot_mem)`  
*Reads the User Access Allowed bit of an IP module.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define STD_VENDOR_ID`  
*Include compiler abstraction.*
- `#define STD_HIGH`  
*Physical state 5V or 3.3V.*
- `#define STD_LOW`  
*Physical state 0V.*
- `#define STD_ACTIVE`  
*Logical state active.*
- `#define STD_IDLE`  
*Logical state idle.*
- `#define STD_ON`  
*ON State.*



- `#define STD_OFF`  
*OFF state.*
- `#define E_NOT_OK`  
*Return code for failure/error.*
- `#define STATUSTYPEDEFINED`  
*Because E\_OK is already defined within OSEK, the symbol E\_OK has to be shared. To avoid name clashes and redefinition problems, the symbols have to be defined in the following way (approved within implementation).*
- `#define E_OK`  
*Success return code.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*
- `#define MEMMAP_ERROR`  
*Symbol used for checking correctness of the includes.*

## Types Reference

- `typedef uint16 PduIdType`  
*This type serve as a unique identifier of a PDU within a software module. Allowed ranges: uint8 .. uint16.*
- `typedef uint32 PduLengthType`  
*This type serve as length information of a PDU in bytes. Allowed ranges: uint8 .. uint32.*
- `typedef uint8 NotifResultType`  
*Variables of this type are used to store the result status of a notification (confirmation or indication).*
- `typedef uint8 NetworkHandleType`  
*Variables of the type NetworkHandleType are used to store the identifier of a communication channel.*
- `typedef uint8 PNCHandleType`  
*Variables of the type PNCHandleType used to store the identifier of a partial network cluster.*
- `typedef uint8 IcomConfigIdType`  
*Variables of the type IcomConfigIdType defines the configuration ID. An ID of 0 is the default configuration. An ID greater than 0 shall identify a configuration for Pretended Networking.*
- `typedef bool boolean`  
*The standard AUTOSAR type boolean shall be implemented on basis of an eight bits long unsigned integer.*
- `typedef uint8_t uint8`  
*Unsigned 8 bit integer with range of 0 ..+255 (0x00..0xFF) - 8 bit.*
- `typedef uint16_t uint16`  
*Unsigned 16 bit integer with range of 0 ..+65535 (0x0000..0xFFFF) - 16 bit.*
- `typedef uint32_t uint32`  
*Unsigned 32 bit integer with range of 0 ..+4294967295 (0x00000000..0xFFFFFFFF) - 32 bit.*
- `typedef uint64_t uint64`  
*Unsigned 64 bit integer with range of 0..18446744073709551615 (0x0000000000000000..0xFFFFFFFFFFFFFFFF)- 64 bit.*
- `typedef int8_t sint8`  
*Signed 8 bit integer with range of -128 ..+127 (0x80..0x7F) - 7 bit + 1 sign bit.*
- `typedef int16_t sint16`  
*Signed 16 bit integer with range of -32768 ..+32767 (0x8000..0x7FFF) - 15 bit + 1 sign bit.*

- typedef int32\_t [sint32](#)  
*Signed 32 bit integer with range of -2147483648.. +2147483647 (0x80000000..0x7FFFFFFF) - 31 bit + 1 sign bit.*
- typedef int64\_t [sint64](#)  
*Signed 64 bit integer with range of -9223372036854775808..9223372036854775807 (0x8000000000000000..0x7FFFFFFFFFFFFFFF) - 63 bit + 1 sign bit.*
- typedef uint\_least8\_t [uint8\\_least](#)  
*Unsigned integer at least 8 bit long. Range of at least 0 ..+255 (0x00..0xFF) - 8 bit.*
- typedef uint\_least16\_t [uint16\\_least](#)  
*Unsigned integer at least 16 bit long. Range of at least 0 ..+65535 (0x0000..0xFFFF) - 16 bit.*
- typedef uint\_least32\_t [uint32\\_least](#)  
*Unsigned integer at least 32 bit long. Range of at least 0 ..+4294967295 (0x00000000..0xFFFFFFFF) - 32 bit.*
- typedef int\_least8\_t [sint8\\_least](#)  
*Signed integer at least 8 bit long. Range - at least -128 ..+127. At least 7 bit + 1 bit sign.*
- typedef int\_least16\_t [sint16\\_least](#)  
*Signed integer at least 16 bit long. Range - at least -32768 ..+32767. At least 15 bit + 1 bit sign.*
- typedef int\_least32\_t [sint32\\_least](#)  
*Signed integer at least 32 bit long. Range - at least -2147483648.. +2147483647. At least 31 bit + 1 bit sign.*
- typedef float [float32](#)  
*32bit long floating point data type*
- typedef double [float64](#)  
*64bit long floating point data type*
- typedef uint8 [StatusType](#)  
*This type is defined for OSEK compliance.*
- typedef uint8 [Std\\_ReturnType](#)  
*This type can be used as standard API return type which is shared between the RTE and the BSW modules.*

## Enum Reference

- enum [Can\\_ControllerStateType](#)  
*CAN Controller State Modes of operation.*
- enum [Can\\_ErrorStateType](#)  
*CAN Controller State Modes of operation.*
- enum [CanTrcv\\_TrsvModeType](#)  
*CAN Transceiver modes.*
- enum [CanTrcv\\_TrsvWakeupModeType](#)
- enum [CanTrcv\\_TrsvWakeupReasonType](#)
- enum [BufReq\\_ReturnType](#)  
*Variables of this type are used to store the result of a buffer request.*
- enum [TpDataStateType](#)  
*Variables of this type shall be used to store the state of TP buffer.*
- enum [TPParameterType](#)  
*Specify the parameter to which the value has to be changed (BS or STmin)*
- enum [IcomSwitch\\_ErrorType](#)  
*IcomSwitch\_ErrorType defines the errors which can occur when activating or deactivating Pretended Networking.*

## Variables

- [Can\\_IdType id](#)  
*CAN L-PDU = Data Link Layer Protocol Data Unit. Consists of Identifier, DLC and Data(SDU) It is uint32 for CAN\_EXTENDEDEDID=STD\_ON, else is uint16.*
- [PduIdType swPduHandle](#)  
*The L-PDU Handle = defined and placed inside the CanIf module layer. Each handle represents an L-PDU, which is a constant structure with information for Tx/Rx processing.*
- [uint8 length](#)  
*DLC = Data Length Code (part of L-PDU that describes the SDU length).*
- [uint8 \\* sdu](#)  
*CAN L-SDU = Link Layer Service Data Unit. Data that is transported inside the L-PDU.*
- [Can\\_IdType CanId](#)  
*Standard/Extended CAN ID of CAN L-PDU.*
- [Can\\_HwHandleType Hoh](#)  
*ID of the corresponding Hardware Object Range.*
- [uint8 ControllerId](#)  
*ControllerId provided by CanIf clearly identify the corresponding controller.*
- [uint32 nanoseconds](#)  
*Nanoseconds part of the time.*
- [uint32 seconds](#)  
*32 bit LSB of the 48 bits Seconds part of the time*
- [uint16 secondsHi](#)  
*16 bit MSB of the 48 bits Seconds part of the time*
- [Eth\\_TimeStampType diff](#)  
*diff time difference*
- [boolean sign](#)  
*Positive (True) Or negative (False) time.*
- [Eth\\_TimeIntDiffType IngressTimeStampDelta](#)  
*IngressTimeStampSync2 -IngressTimeStampSync1.*
- [Eth\\_TimeIntDiffType OriginTimeStampDelta](#)  
*OriginTimeStampSync2[FUP2]-OriginTimeStampSync1[FUP1].*
- [uint8 SwitchIdx](#)  
*Switch index.*
- [uint8 SwitchPortIdx](#)  
*Port index of the switch.*
- [uint8 srcMacAddrFilter \[6U\]](#)  
*Specifies the source MAC address [0..255,0..255,0..255,0..255,0..255,0..255] that should be mirrored. If set to 0,0,0,0,0,0, no source MAC address filtering shall take place.*
- [uint8 dstMacAddrFilter \[6U\]](#)  
*Specifies the destination MAC address [0..255,0..255,0..255,0..255,0..255,0..255] that should be mirrored. If set to 0,0,0,0,0,0, no destination MAC address filtering shall take place.*
- [uint16 VlanIdFilter](#)  
*Specifies the VLAN address 0..4094 that should be mirrored. If set to 65535, no VLAN filtering shall take place.*
- [uint8 MirroringPacketDivider](#)  
*Divider if only a subset of received frames should be mirrored. E.g. MirroringPacketDivider = 2 means every second frames is mirrored.*
- [uint8 MirroringMode](#)

specifies the mode how the mirrored traffic should be tagged : 0x00 == No VLAN retagging; 0x01 == VLAN retagging; 0x03 == VLAN Double tagging

- [uint32 TrafficDirectionIngressBitMask](#)

Specifies the bit mask of Ethernet switch ingress port traffic direction to be mirrored. The bit mask is calculated depending of the values of EthSwtPortIdx. (e.g. set EthSwtPortIdx == 2 => TrafficDirectionIngressBitMask = 0b0000 0000 0000 0000 0000 0000 0100). 0b0 == enable ingress port mirroring 0b1 == disable ingress port mirroring Example: TrafficDirectionIngressBitMask = 0b0000 0000 0000 0000 0000 0000 0100 => Ingress traffic mirroring is enabled of Ethernet switch port with EthSwtPortIdx=2.

- [uint32 TrafficDirectionEgressBitMask](#)

Specifies the bit mask of Ethernet switch egress port traffic direction to be mirrored. The bit mask is calculated depending of the values of EthSwtPortIdx. (e.g. set EthSwtPortIdx == 2 => TrafficDirectionEgressBitMask = 0b0000 0000 0000 0000 0000 0000 0100). 0b0 == enable egress port mirroring 0b1 == disable egress port mirroring Example: TrafficDirectionEgressBitMask = 0b0000 0000 0000 0000 0000 0000 0001 => Egress traffic mirroring is enabled of Ethernet switch port with EthSwtPortIdx=0.

- [uint8 CapturePortIdx](#)

Specifies the Ethernet switch port which capture the mirrored traffic.

- [uint16 ReTaggingVlanId](#)

Specifies the VLAN address 0..4094 which shall be used for re-tagging if MirroringMode is set to 0x01 (VLAN re-tagging). If the value is set to 65535, the value shall be ignored, because the VLAN address for re-tagging is provided by the Ethernet switch configuration.

- [uint16 DoubleTaggingVlanId](#)

Specifies the VLAN address 0..4094 which shall be used for double-tagging if MirroringMode is set to 0x02 (VLAN double tagging). If the value is set to 65535, the value shall be ignored, because the VLAN address for double tagging is provided by the Ethernet switch configuration.

- [Std\\_ReturnType IngressTimestampValid](#)

IngressTimestampValid shall be set to E\_NOT\_OK if ingress timestamp is not available.

- [Std\\_ReturnType EgressTimestampValid](#)

EgressTimestampValid shall be set to E\_NOT\_OK if ingress timestamp is not available.

- [Std\\_ReturnType MgmtInfoValid](#)

MgmtInfoValid shall be set to E\_NOT\_OK if ingress timestamp is not available(e.g. timeout).

- [EthSwt\\_MgmtObjectValidType Validation](#)

The validation information for the mgmt\_obj.

- [Eth\\_TimeStampType IngressTimestamp](#)

The ingress timestamp value out of the switch.

- [Eth\\_TimeStampType EgressTimestamp](#)

The egress timestamp value out of the switch.

- [EthSwt\\_MgmtInfoType MgmtInfo](#)

Received/Transmitted Management information of the switches.

- [EthSwt\\_MgmtOwner Ownership](#)

The ownership of MgmtObj.

- [Lin\\_FramePidType Pid](#)

LIN frame identifier.

- [Lin\\_FrameCsModelType Cs](#)

Checksum model type.

- [Lin\\_FrameResponseType Drc](#)

Response type.

- [Lin\\_FrameDlType Dl](#)

Data length.

- [uint8 \\* SduPtr](#)

*Pointer to Sdu.*

- `#define ADC_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define BASE_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define CAN_43_FLEXCAN_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define CAN_43_LLCE_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define CAN_GENERALTYPES_AR_RELEASE_MAJOR_VERSION`  
*Parameters that shall be published within the modules header file. The integration of incompatible files shall be avoided.*
- `typedef uint32 Can_IdType`  
*Can\_IdType.*
- `typedef uint16 Can_HwHandleType`  
*Can\_HwHandleType.*
- `#define COMPILERDEFINITION_VENDOR_ID`  
*Parameters that shall be published within the compiler abstraction header file and also in the module's description file.*
- `#define CAN_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define ADC_CODE`  
*ADC memory and pointer classes.*
- `#define CAN_CODE`  
*CAN memory and pointer classes.*
- `#define CAN_43_LLCE_CODE`  
*CAN\_43\_LLCE memory and pointer classes.*
- `#define CANIF_CODE`  
*CANIF memory and pointer classes.*

- `#define CRCU_CODE`  
*CRCU memory and pointer classes.*
- `#define CSEC_CODE`  
*CSEC memory and pointer classes.*
- `#define DEM_CODE`  
*DEM memory and pointer classes.*
- `#define DET_CODE`  
*DET memory and pointer classes.*
- `#define DIO_CODE`  
*DIO memory and pointer classes.*
- `#define EEP_CODE`  
*EEP memory and pointer classes.*
- `#define ETH_CODE`  
*ETH memory and pointer classes.*
- `#define ETHIF_CODE`  
*ETH memory and pointer classes.*
- `#define ETHTRCV_CODE`  
*ETH memory and pointer classes.*
- `#define FEE_CODE`  
*FEE memory and pointer classes.*
- `#define FLS_CODE`  
*FLS memory and pointer classes.*
- `#define FR_CODE`  
*FlexRay memory and pointer classes.*
- `#define GPT_CODE`  
*GPT memory and pointer classes.*
- `#define ICU_CODE`  
*ICU memory and pointer classes.*

- `#define I2C_CODE`  
*I2C memory and pointer classes.*
- `#define LIN_CODE`  
*LIN memory and pointer classes.*
- `#define LIN_43_LLCE_CODE`  
*LIN\_43\_LLCE memory and pointer classes.*
- `#define LINIF_CODE`  
*LIN memory and pointer classes.*
- `#define MCEM_CODE`  
*MCEM memory and pointer classes.*
- `#define MCL_CODE`  
*MCL memory and pointer classes.*
- `#define COMPILER_VENDOR_ID`  
*Parameters that shall be published within the compiler abstraction header file and also in the module's description file.*
- `#define MCU_CODE`  
*MCU memory and pointer classes.*
- `#define PMIC_CODE`  
*PMIC memory and pointer classes.*
- `#define PORT_CODE`  
*PORT memory and pointer classes.*
- `#define PWM_CODE`  
*PWM memory and pointer classes.*
- `#define RAMTST_CODE`  
*RamTST memory and pointer classes.*
- `#define SENT_CODE`  
*SENT memory and pointer classes.*
- `#define SCHM_CODE`

*SchM memory and pointer classes.*

- `#define SPI_CODE`  
*SPI memory and pointer classes.*
- `#define TM_CODE`  
*TM memory and pointer classes.*
- `#define WDG_CODE`  
*WDG memory and pointer classes.*
- `#define WDGIF_CODE`  
*WDGIF memory and pointer classes.*
- `#define CRC_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define CSEC_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define CRYPTO_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define CANTRCV_43_TJA1145A_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define DEM_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define DET_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define DIO_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define ECUM_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define EEP_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*



- `#define ETH_43_PFE_MEMMAP_VENDOR_ID`  
Parameters that shall be published within the memory map header file and also in the module's description file.
- `#define ETH_MEMMAP_VENDOR_ID`  
Parameters that shall be published within the memory map header file and also in the module's description file.
- `#define ETHTRCV_43_PHY_MEMMAP_VENDOR_ID`  
Parameters that shall be published within the memory map header file and also in the module's description file.
- `#define ETHSWT_43_SJA11XX_MEMMAP_VENDOR_ID`  
Parameters that shall be published within the memory map header file and also in the module's description file.
- `#define ETHTRCV_43_TJA110X_MEMMAP_VENDOR_ID`  
Parameters that shall be published within the memory map header file and also in the module's description file.
- `#define ETHSWITCH_43_SJA1105P_MEMMAP_VENDOR_ID`  
Parameters that shall be published within the memory map header file and also in the module's description file.
- `enum Fr_TxLPduStatusType`  
Transmit resource status is stored to variable of this type.
- `enum Fr_RxLPduStatusType`  
Transmit resource status is stored to variable of this type.
- `enum Fr_POCTestType`
- `enum Fr_SlotModeType`  
This type is used to store the slot mode of the controller.
- `enum Fr_ErrorModeType`  
Variables of this type are used for storage of FlexRay controller error mode.
- `enum Fr_WakeupStatusType`
- `enum Fr_StartupStateType`
- `enum Fr_ChannelType`
- `#define FR_GENERALTYPES_AR_RELEASE_MAJOR_VERSION`  
Parameters that shall be published within the modules header file. The integration of incompatible files shall be avoided.
- `#define FR_CIDX_GDCYCLE`  
Macros which can be passed into `Fr_ReadCCConfig` as parameter `Fr_ConfigParamIdx`.
- `#define FR_SLOTMODE_SINGLE`  
This macro is used for backward compatibility with Autosar 3.0 definition of `Fr_SlotModeType`.
- `#define FEE_MEMMAP_VENDOR_ID`  
Parameters that shall be published within the memory map header file and also in the module's description file.
- `#define FR_43_LLCE_MEMMAP_VENDOR_ID`  
Parameters that shall be published within the memory map header file and also in the module's description file.

- `#define GPT_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define FLS_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define I2S_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define FR_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define ICU_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define I2C_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `enum Lin_FrameCsModelType`  
*Checksum models for the LIN Frame.*
- `enum Lin_FrameResponseType`  
*Frame response types.*
- `enum Lin_StatusType`  
*LIN Frame and Channel states operation.*
- `enum Lin_SlaveErrorType`  
*LIN Slave error type.*
- `typedef uint8 Lin_FrameDlType`  
*Data length of a LIN Frame.*
- `typedef uint8 Lin_FramePidType`  
*The LIN identifier (0..0x3F) with its parity bits.*
- `#define LIN_GENERALTYPES_AR_RELEASE_MAJOR_VERSION`  
*Parameters that shall be published within the modules header file. The integration of incompatible files shall be avoided.*
- `#define LIN_43_LLCE_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define MCEM_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define ISELED_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*

- `#define MCU_MEMMAP_VENDOR_ID`  
Parameters that shall be published within the memory map header file and also in the module's description file.
- `#define MCL_MEMMAP_VENDOR_ID`  
Parameters that shall be published within the memory map header file and also in the module's description file.
- `#define LIN_MEMMAP_VENDOR_ID`  
Parameters that shall be published within the memory map header file and also in the module's description file.
- `#define OCU_MEMMAP_VENDOR_ID`  
Parameters that shall be published within the memory map header file and also in the module's description file.
- `enum Eth_StateType`  
The Ethernet driver state.
- `enum Eth_ModeType`  
The Ethernet controller mode.
- `enum Eth_RxStatusType`  
The Ethernet reception status.
- `enum Eth_FilterActionType`  
Action type for PHY address filtering.
- `enum Eth_TimeStampQualType`  
The Ethernet quality of timestamp type.
- `enum EthTrcv_ModeType`  
This type defines the transceiver modes.
- `enum EthTrcv_LinkStateType`  
This type defines the Ethernet link state. The link state changes after an Ethernet cable gets plugged in and the transceivers on both ends negotiated the transmission parameters (i.e. baud rate and duplex mode)
- `enum EthTrcv_StateType`  
This type defines the Ethernet link state. The link state changes after an Ethernet cable gets plugged in and the transceivers on both ends negotiated the transmission parameters (i.e. baud rate and duplex mode)
- `enum EthTrcv_BaudRateType`  
This type defines the Ethernet baud rate. The baud rate gets either negotiated between the connected transceivers or has to be configured.
- `enum EthTrcv_DuplexModeType`  
This type defines the Ethernet duplex mode. The duplex mode gets either negotiated between the connected transceivers or has to be configured.
- `enum EthTrcv_WakeupModeType`  
This type controls the transceiver wake up modes and/or clears the wake-up reason.
- `enum EthTrcv_WakeupReasonType`  
This type defines the transceiver wake up reasons.
- `enum EthTrcv_PhyTestModeType`  
Describes the possible PHY test modes.
- `enum EthTrcv_PhyLoopbackModeType`  
Describes the possible PHY loopback modes.
- `enum EthTrcv_PhyTxModeType`  
Describes the possible PHY transmit modes.

- enum [EthTrcv\\_CableDiagResultType](#)  
*Describes the results of the cable diagnostics.*
- enum [EthSwt\\_StateType](#)  
*Status supervision used for Development Error Detection. The state shall be available for debugging.*
- enum [EthSwt\\_MacLearningType](#)  
*MAC learning type enumeration.*
- enum [EthSwt\\_PortMirrorStateType](#)  
*Type to request or obtain the port mirroring state (enable/disable) for a particular port mirror configuration per Ethernet switch.*
- enum [EthSwt\\_MgmtOwner](#)  
*Holds information if upper layer or EthSwt is owner of mgmt\_obj.*
- typedef [uint16 Eth\\_FrameType](#)  
*Frame type.*
- typedef [uint8 Eth\\_DataType](#)  
*Type used to pass transmit or receive data to or from the driver.*
- typedef [uint32 Eth\\_BufIdxType](#)  
*Type used to identify the ethernet buffer.*
- #define [ETH\\_GENERALTYPES\\_AR\\_RELEASE\\_MAJOR\\_VERSION](#)  
*Parameters that shall be published within the modules header file. The integration of incompatible files shall be avoided.*
- #define [PWM\\_MEMMAP\\_VENDOR\\_ID](#)  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- #define [PORT\\_MEMMAP\\_VENDOR\\_ID](#)  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- #define [PMIC\\_MEMMAP\\_VENDOR\\_ID](#)  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- #define [PLATFORM\\_MEMMAP\\_VENDOR\\_ID](#)  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- #define [QDEC\\_MEMMAP\\_VENDOR\\_ID](#)  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- #define [RM\\_MEMMAP\\_VENDOR\\_ID](#)  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- #define [RTE\\_MEMMAP\\_VENDOR\\_ID](#)  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- #define [UART\\_MEMMAP\\_VENDOR\\_ID](#)  
*Parameters that shall be published within the memory map header file and also in the module's description file.*

- `#define SBC_FS26_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define SENT_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define SPI_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define WDG_43_FS26_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*
- `#define WDG_MEMMAP_VENDOR_ID`  
*Parameters that shall be published within the memory map header file and also in the module's description file.*

## 6.2.2 Data Structure Documentation

### 6.2.2.1 struct Can\_PduType

`Can_PduType`.

Type used to provide ID, DLC, SDU from CAN interface to CAN driver. HTH or HRH = ID+DLC+SDU.

Definition at line 175 of file `Can_GeneralTypes.h`.

#### Data Fields

- `Can_IdType id`  
*CAN L-PDU = Data Link Layer Protocol Data Unit. Consists of Identifier, DLC and Data(SDU) It is uint32 for CAN\_EXTENDEDID=STD\_ON, else is uint16.*
- `PduIdType swPduHandle`  
*The L-PDU Handle = defined and placed inside the CanIf module layer. Each handle represents an L-PDU, which is a constant structure with information for Tx/Rx processing.*
- `uint8 length`  
*DLC = Data Length Code (part of L-PDU that describes the SDU length).*
- `uint8 * sdu`  
*CAN L-SDU = Link Layer Service Data Unit. Data that is transported inside the L-PDU.*

### 6.2.2.2 struct Can\_HwType

`Can_HwType`.

This type defines a data structure which clearly provides an Hardware Object Handle including its corresponding CAN Controller and therefore CanDrv as well as the specific CanId.

Definition at line 214 of file `Can_GeneralTypes.h`.

### Data Fields

- [Can\\_IdType](#) CanId  
*Standard/Extended CAN ID of CAN L-PDU.*
- [Can\\_HwHandleType](#) Hoh  
*ID of the corresponding Hardware Object Range.*
- [uint8](#) ControllerId  
*ControllerId provided by CanIf clearly identify the corresponding controller.*

#### 6.2.2.3 struct PduInfoType

Variables of this type are used to store the basic information about a PDU of any type, namely a pointer variable pointing to it's SDU (payload), and the corresponding length of the SDU in bytes.

Definition at line 266 of file ComStack\_Types.h.

### Data Fields

- [uint8 \\*](#) SduDataPtr
- [PduLengthType](#) SduLength

#### 6.2.2.3.1 Field Documentation

##### 6.2.2.3.1.1 SduDataPtr [uint8\\*](#) SduDataPtr

pointer to the SDU (i.e. payload data) of the PDU

Definition at line 268 of file ComStack\_Types.h.

##### 6.2.2.3.1.2 SduLength [PduLengthType](#) SduLength

length of the SDU in bytes

Definition at line 270 of file ComStack\_Types.h.

#### 6.2.2.4 struct RetryInfoType

Variables of this type shall be used to store the information about Tp buffer handling.

Definition at line 277 of file ComStack\_Types.h.

## Data Fields

- [TpDataStateType](#) [TpDataState](#)
- [PduLengthType](#) [TxTpDataCnt](#)

### 6.2.2.4.1 Field Documentation

#### 6.2.2.4.1.1 [TpDataState](#) [TpDataStateType](#) [TpDataState](#)

The enum type to be used to store the state of Tp buffer

Definition at line 279 of file `ComStack_Types.h`.

#### 6.2.2.4.1.2 [TxTpDataCnt](#) [PduLengthType](#) [TxTpDataCnt](#)

length of the SDU in bytes

Definition at line 280 of file `ComStack_Types.h`.

### 6.2.2.5 struct [Eth\\_TimeStampType](#)

Type used to express the timestamp value.

Variables of this type are used for expressing time stamps including relative time and absolute calendar time. The absolute time starts acc. to "[5], Annex C/C1" specification at 1970-01-01. 0 to 281474976710655s == 3257812230d [0xFFFF FFFF FFFF] 0 to 999999999ns [0x3B9A C9FF] invalid value in nanoseconds: [0x3B9A CA00] to [0xFFFF FFFF] Bit 30 and 31 reserved, default: 0

Definition at line 410 of file `Eth_GeneralTypes.h`.

## Data Fields

- [uint32](#) [nanoseconds](#)  
*Nanoseconds part of the time.*
- [uint32](#) [seconds](#)  
*32 bit LSB of the 48 bits Seconds part of the time*
- [uint16](#) [secondsHi](#)  
*16 bit MSB of the 48 bits Seconds part of the time*

### 6.2.2.6 struct Eth\_TimeIntDiffType

Type used to express the diff between timestamp values.

Variables of this type are used to express time differences in a usual way. The described "TimeInterval" type referenced in ", chapter 6.3.3.3" will not be used and hereby slightly simplified.

Definition at line 424 of file Eth\_GeneralTypes.h.

#### Data Fields

- [Eth\\_TimeStampType diff](#)  
*diff time difference*
- [boolean sign](#)  
*Positive (True) Or negative (False) time.*

### 6.2.2.7 struct Eth\_RateRatioType

Type used to express frequency ratio.

Variables of this type are used to express frequency ratios.

Definition at line 435 of file Eth\_GeneralTypes.h.

#### Data Fields

- [Eth\\_TimeIntDiffType IngressTimeStampDelta](#)  
*IngressTimeStampSync2 -IngressTimeStampSync1.*
- [Eth\\_TimeIntDiffType OriginTimeStampDelta](#)  
*OriginTimeStampSync2[FUP2]-OriginTimeStampSync1[FUP1].*

### 6.2.2.8 struct Eth\_CounterType

Type used to statistic counter for diagnostics.

Variables of this type are used to statistic counter for diagnostics.

Definition at line 446 of file Eth\_GeneralTypes.h.

### 6.2.2.9 struct Eth\_RxStatsType

Type used to statistic counter for diagnostics.

Variables of this type are used to statistic counter for diagnostics.

Definition at line 472 of file Eth\_GeneralTypes.h.



**6.2.2.10 struct Eth\_TxStatsType**

Type used to statistic counter for diagnostics.

Variables of this type are used to statistic counter for diagnostics.

Definition at line 499 of file Eth\_GeneralTypes.h.

**6.2.2.11 struct Eth\_TxErrorCounterValuesType**

Type used to statistic counter for diagnostics.

Variables of this type are used to statistic counter for diagnostics.

Definition at line 511 of file Eth\_GeneralTypes.h.

**6.2.2.12 struct Eth\_MacVlanType**

Type used for VLAN management in EthSwt.

Variables of this type are used to store information related to VLAN.

Definition at line 527 of file Eth\_GeneralTypes.h.

**6.2.2.13 struct EthSwt\_MgmtInfoType**

Type for holding the management information received/transmitted on Switches (ports).

It contains the switch index and the port index of the switch

Definition at line 539 of file Eth\_GeneralTypes.h.

Data Fields

Type	Name	Description
<a href="#">uint8</a>	SwitchIdx	Switch index.
<a href="#">uint8</a>	SwitchPortIdx	Port index of the switch.

**6.2.2.14 struct EthSwt\_PortMirrorCfgType**

The [EthSwt\\_PortMirrorCfgType](#) specify the port mirror configuration which is set up per Ethernet switch. The configuration is written to the Ethernet switch driver by calling EthSwt\_WritePortMirrorConfiguration. One port mirror configuration is maintained per Ethernet Switch.

Definition at line 548 of file Eth\_GeneralTypes.h.

## Data Fields

Type	Name	Description
uint8	srcMacAddrFilter[6U]	Specifies the source MAC address [0..255,0..255,0..255,0..255,0..255,0..255] that should be mirrored. If set to 0,0,0,0,0,0, no source MAC address filtering shall take place.
uint8	dstMacAddrFilter[6U]	Specifies the destination MAC address [0..255,0..255,0..255,0..255,0..255,0..255] that should be mirrored. If set to 0,0,0,0,0,0, no destination MAC address filtering shall take place.
uint16	VlanIdFilter	Specifies the VLAN address 0..4094 that should be mirrored. If set to 65535, no VLAN filtering shall take place.
uint8	MirroringPacketDivider	Divider if only a subset of received frames should be mirrored. E.g. MirroringPacketDivider = 2 means every second frames is mirrored.
uint8	MirroringMode	specifies the mode how the mirrored traffic should be tagged : 0x00 == No VLAN retagging; 0x01 == VLAN retagging; 0x03 == VLAN Double tagging
uint32	TrafficDirectionIngressBitMask	Specifies the bit mask of Ethernet switch ingress port traffic direction to be mirrored. The bit mask is calculated depending of the values of EthSwtPortIdx. (e.g. set EthSwtPortIdx == 2 => TrafficDirectionIngressBitMask = 0b0000 0000 0000 0000 0000 0000 0000 0100). 0b0 == enable ingress port mirroring 0b1 == disable ingress port mirroring Example: TrafficDirectionIngressBitMask = 0b0000 0000 0000 0000 0000 0000 0000 0100 => Ingress traffic mirroring is enabled of Ethernet switch port with EthSwtPortIdx=2.
uint32	TrafficDirectionEgressBitMask	Specifies the bit mask of Ethernet switch egress port traffic direction to be mirrored. The bit mask is calculated depending of the values of EthSwtPortIdx. (e.g. set EthSwtPortIdx == 2 => TrafficDirectionEgressBitMask = 0b0000 0000 0000 0000 0000 0000 0000 0100). 0b0 == enable egress port mirroring 0b1 == disable egress port mirroring Example: TrafficDirectionEgressBitMask = 0b0000 0000 0000 0000 0000 0000 0000 0001 => Egress traffic mirroring is enabled of Ethernet switch port with EthSwtPortIdx=0.
uint8	CapturePortIdx	Specifies the Ethernet switch port which capture the mirrored traffic.
uint16	ReTaggingVlanId	Specifies the VLAN address 0..4094 which shall be used for re-tagging if MirroringMode is set to 0x01 (VLAN re-tagging). If the value is set to 65535, the value shall be ignored, because the VLAN address for re-tagging is provided by the Ethernet switch configuration.
uint16	DoubleTaggingVlanId	Specifies the VLAN address 0..4094 which shall be used for double-tagging if MirroringMode is set to 0x02 (VLAN double tagging). If the value is set to 65535, the value shall be ignored, because the VLAN address for double tagging is provided by the Ethernet switch configuration.

## 6.2.2.15 struct EthSwt\_MgmtObjectValidType

Will be set from EthSwt and marks EthSwt\_MgmtObject as valid or not. So the upper layer will be able to detect inconsistencies.

Definition at line 566 of file Eth\_GeneralTypes.h.

## Data Fields

Type	Name	Description
<a href="#">Std_ReturnType</a>	IngressTimestampValid	IngressTimestampValid shall be set to E_NOT_OK if ingress timestamp is not available.
<a href="#">Std_ReturnType</a>	EgressTimestampValid	EgressTimestampValid shall be set to E_NOT_OK if ingress timestamp is not available.
<a href="#">Std_ReturnType</a>	MgmtInfoValid	MgmtInfoValid shall be set to E_NOT_OK if ingress timestamp is not available(e.g. timeout).

**6.2.2.16 struct EthSwt\_MgmtObjectType**

Provides information about all struct member elements. The ownership gives information whether EthSwt has finished its activities in providing all struct member elements.

Definition at line 576 of file Eth\_GeneralTypes.h.

## Data Fields

Type	Name	Description
<a href="#">EthSwt_MgmtObjectValidType</a>	Validation	The validation information for the mgmt_obj.
<a href="#">Eth_TimeStampType</a>	IngressTimestamp	The ingress timestamp value out of the switch.
<a href="#">Eth_TimeStampType</a>	EgressTimestamp	The egress timestamp value out of the switch.
<a href="#">EthSwt_MgmtInfoType</a>	MgmtInfo	Received/Transmitted Management information of the switches.
<a href="#">EthSwt_MgmtOwner</a>	Ownership	The ownership of MgmtObj.

**6.2.2.17 struct Fr\_POCTestStatusType**

Variables of this type are used to query the flexRay controller status.

Definition at line 289 of file Fr\_GeneralTypes.h.

**6.2.2.18 struct Fr\_SlotAssignmentType**

Variables of this type are used to store information of frame indentified by Fr\_LPduIdx.

Definition at line 309 of file Fr\_GeneralTypes.h.

**6.2.2.19 struct Lin\_PduType**

The LIN identifier (0..0x3F) with its parity bits.

This Type is used to provide PID, checksum model, data length and SDU pointer from the LIN Interface to the LIN driver.

Definition at line 210 of file Lin\_GeneralTypes.h.

## Data Fields

- [Lin\\_FramePidType](#) Pid  
*LIN frame identifier.*
- [Lin\\_FrameCsModelType](#) Cs  
*Checksum model type.*
- [Lin\\_FrameResponseType](#) Drc  
*Response type.*
- [Lin\\_FrameDlType](#) Dl  
*Data length.*
- [uint8](#) \* [SduPtr](#)  
*Pointer to Sdu.*

### 6.2.2.20 struct Mcal\_DemErrorType

Typedef for DEM error management implemented by MCAL drivers.

Definition at line 568 of file Mcal.h.

#### Data Fields

Type	Name	Description
<a href="#">uint32</a>	state	enabling/disabling the DEM error: Active=STD_ON/ Inactive=STD_OFF
<a href="#">uint32</a>	id	ID of DEM error (0 if STD_OFF)

### 6.2.2.21 struct Std\_VersionInfoType

This type shall be used to request the version of a BSW module using the "ModuleName"\_\_GetVersionInfo() function.

Definition at line 181 of file StandardTypes.h.

#### Data Fields

Type	Name	Description
<a href="#">uint16</a>	vendorID	vendor ID
<a href="#">uint16</a>	moduleID	BSW module ID.
<a href="#">uint8</a>	sw_major_version	BSW module software major version.
<a href="#">uint8</a>	sw_minor_version	BSW module software minor version.
<a href="#">uint8</a>	sw_patch_version	BSW module software patch version.

## 6.2.3 Macro Definition Documentation

### 6.2.3.1 ADC\_MEMMAP\_VENDOR\_ID

```
#define ADC_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Adc\_MemMap.h.

### 6.2.3.2 MEMMAP\_ERROR [1/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Adc\_MemMap.h.

### 6.2.3.3 BASE\_MEMMAP\_VENDOR\_ID

```
#define BASE_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Base\_MemMap.h.

### 6.2.3.4 MEMMAP\_ERROR [2/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Base\_MemMap.h.

### 6.2.3.5 CAN\_43\_FLEXCAN\_MEMMAP\_VENDOR\_ID

```
#define CAN_43_FLEXCAN_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Can\_43\_FLEXCAN\_MemMap.h.

#### 6.2.3.6 MEMMAP\_ERROR [3/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Can\_43\_FLEXCAN\_MemMap.h.

#### 6.2.3.7 CAN\_43\_LLCE\_MEMMAP\_VENDOR\_ID

```
#define CAN_43_LLCE_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Can\_43\_LLCE\_MemMap.h.

#### 6.2.3.8 MEMMAP\_ERROR [4/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Can\_43\_LLCE\_MemMap.h.

#### 6.2.3.9 CAN\_GENERALTYPES\_AR\_RELEASE\_MAJOR\_VERSION

```
#define CAN_GENERALTYPES_AR_RELEASE_MAJOR_VERSION
```

Parameters that shall be published within the modules header file. The integration of incompatible files shall be avoided.

Definition at line 61 of file Can\_GeneralTypes.h.

#### 6.2.3.10 CAN\_BUSY

```
#define CAN_BUSY
```

Transmit request could not be processed because no transmit object was available.

Definition at line 82 of file Can\_GeneralTypes.h.

### 6.2.3.11 CAN\_MEMMAP\_VENDOR\_ID

```
#define CAN_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Can\_MemMap.h.

### 6.2.3.12 MEMMAP\_ERROR [5/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Can\_MemMap.h.

### 6.2.3.13 CANTRCV\_43\_TJA1145A\_MEMMAP\_VENDOR\_ID

```
#define CANTRCV_43_TJA1145A_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file CanTrcv\_43\_tja1145a\_MemMap.h.

### 6.2.3.14 MEMMAP\_ERROR [6/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file CanTrcv\_43\_tja1145a\_MemMap.h.

### 6.2.3.15 COMPILER\_VENDOR\_ID

```
#define COMPILER_VENDOR_ID
```

Parameters that shall be published within the compiler abstraction header file and also in the module's description file.

@requirements COMPILER047

Definition at line 65 of file Compiler.h.



### 6.2.3.16 AUTOMATIC

```
#define AUTOMATIC
```

The memory class AUTOMATIC shall be provided as empty definition, used for the declaration of local pointers.

Definition at line 89 of file Compiler.h.

### 6.2.3.17 TYPEDEF

```
#define TYPEDEF
```

The memory class TYPEDEF shall be provided as empty definition. This memory class shall be used within type definitions, where no memory qualifier can be specified. This can be necessary for defining pointer types, with e.g. P2VAR, where the macros require two parameters. First parameter can be specified in the type definition (distance to the memory location referenced by the pointer), but the second one (memory allocation of the pointer itself) cannot be defined at this time. Hence memory class TYPEDEF shall be applied.

Definition at line 99 of file Compiler.h.

### 6.2.3.18 NULL\_PTR

```
#define NULL_PTR
```

The compiler abstraction shall provide the NULL\_PTR define with a void pointer to zero definition.

Definition at line 105 of file Compiler.h.

### 6.2.3.19 FUNC

```
#define FUNC(  
    rettype,  
    memclass )
```

The compiler abstraction shall define the FUNC macro for the declaration and definition of functions, that ensures correct syntax of function declarations as required by a specific compiler.

Definition at line 284 of file Compiler.h.

### 6.2.3.20 P2VAR

```
#define P2VAR(  
    ptrtype,  
    memclass,  
    ptrclass )
```

The compiler abstraction shall define the P2VAR macro for the declaration and definition of pointers in RAM, pointing to variables.

Definition at line 290 of file Compiler.h.

### 6.2.3.21 P2CONST

```
#define P2CONST(  
    ptrtype,  
    memclass,  
    ptrclass )
```

The compiler abstraction shall define the P2CONST macro for the declaration and definition of pointers in RAM pointing to constants.

Definition at line 296 of file Compiler.h.

### 6.2.3.22 CONSTP2VAR

```
#define CONSTP2VAR(  
    ptrtype,  
    memclass,  
    ptrclass )
```

The compiler abstraction shall define the CONSTP2VAR macro for the declaration and definition of constant pointers accessing variables.

Definition at line 302 of file Compiler.h.

### 6.2.3.23 CONSTP2CONST

```
#define CONSTP2CONST(  
    ptrtype,  
    memclass,  
    ptrclass )
```

The compiler abstraction shall define the CONSTP2CONST macro for the declaration and definition of constant pointers accessing constants.

Definition at line 308 of file Compiler.h.

**6.2.3.24 P2FUNC**

```
#define P2FUNC(
    rettype,
    ptrclass,
    fctname )
```

The compiler abstraction shall define the P2FUNC macro for the type definition of pointers to functions.

Definition at line 314 of file Compiler.h.

**6.2.3.25 CONST**

```
#define CONST(
    consttype,
    memclass )
```

The compiler abstraction shall define the CONST macro for the declaration and definition of constants.

Definition at line 320 of file Compiler.h.

**6.2.3.26 VAR**

```
#define VAR(
    vartype,
    memclass )
```

The compiler abstraction shall define the VAR macro for the declaration and definition of variables.

Definition at line 326 of file Compiler.h.

**6.2.3.27 CONSTP2FUNC**

```
#define CONSTP2FUNC(
    rettype,
    ptrclass,
    fctname )
```

The compiler abstraction for const pointer to function.

Definition at line 332 of file Compiler.h.

### 6.2.3.28 FUNC\_P2CONST

```
#define FUNC_P2CONST(  
    rettype,  
    ptrclass,  
    memclass )
```

The compiler abstraction shall define the FUNC\_P2CONST macro for the declaration and definition of functions returning a pointer to a constant.

Definition at line 339 of file Compiler.h.

### 6.2.3.29 FUNC\_P2VAR

```
#define FUNC_P2VAR(  
    rettype,  
    ptrclass,  
    memclass )
```

The compiler abstraction shall define the FUNC\_P2VAR macro for the declaration and definition of functions returning a pointer to a variable.

Definition at line 345 of file Compiler.h.

### 6.2.3.30 ADC\_CODE

```
#define ADC_CODE
```

ADC memory and pointer classes.

Definition at line 67 of file Compiler\_Cfg.h.

### 6.2.3.31 CAN\_CODE

```
#define CAN_CODE
```

CAN memory and pointer classes.

Definition at line 83 of file Compiler\_Cfg.h.

#### 6.2.3.32 CAN\_43\_LLCE\_CODE

```
#define CAN_43_LLCE_CODE
```

CAN\_43\_LLCE memory and pointer classes.

Definition at line 99 of file Compiler\_Cfg.h.

#### 6.2.3.33 CANIF\_CODE

```
#define CANIF_CODE
```

CANIF memory and pointer classes.

Definition at line 115 of file Compiler\_Cfg.h.

#### 6.2.3.34 CRCU\_CODE

```
#define CRCU_CODE
```

CRCU memory and pointer classes.

Definition at line 131 of file Compiler\_Cfg.h.

#### 6.2.3.35 CSEC\_CODE

```
#define CSEC_CODE
```

CSEC memory and pointer classes.

Definition at line 147 of file Compiler\_Cfg.h.

#### 6.2.3.36 DEM\_CODE

```
#define DEM_CODE
```

DEM memory and pointer classes.

Definition at line 163 of file Compiler\_Cfg.h.

### 6.2.3.37 DET\_CODE

```
#define DET_CODE
```

DET memory and pointer classes.

Definition at line 179 of file Compiler\_Cfg.h.

### 6.2.3.38 DIO\_CODE

```
#define DIO_CODE
```

DIO memory and pointer classes.

Definition at line 195 of file Compiler\_Cfg.h.

### 6.2.3.39 EEP\_CODE

```
#define EEP_CODE
```

EEP memory and pointer classes.

Definition at line 212 of file Compiler\_Cfg.h.

### 6.2.3.40 ETH\_CODE

```
#define ETH_CODE
```

ETH memory and pointer classes.

Definition at line 229 of file Compiler\_Cfg.h.

### 6.2.3.41 ETHIF\_CODE

```
#define ETHIF_CODE
```

ETH memory and pointer classes.

Definition at line 246 of file Compiler\_Cfg.h.

#### 6.2.3.42 ETHTRCV\_CODE

```
#define ETHTRCV_CODE
```

ETH memory and pointer classes.

Definition at line 263 of file Compiler\_Cfg.h.

#### 6.2.3.43 FEE\_CODE

```
#define FEE_CODE
```

FEE memory and pointer classes.

Definition at line 279 of file Compiler\_Cfg.h.

#### 6.2.3.44 FLS\_CODE

```
#define FLS_CODE
```

FLS memory and pointer classes.

Definition at line 295 of file Compiler\_Cfg.h.

#### 6.2.3.45 FR\_CODE

```
#define FR_CODE
```

FlexRay memory and pointer classes.

Definition at line 311 of file Compiler\_Cfg.h.

#### 6.2.3.46 GPT\_CODE

```
#define GPT_CODE
```

GPT memory and pointer classes.

Definition at line 327 of file Compiler\_Cfg.h.

### 6.2.3.47 ICU\_CODE

```
#define ICU_CODE
```

ICU memory and pointer classes.

Definition at line 343 of file Compiler\_Cfg.h.

### 6.2.3.48 I2C\_CODE

```
#define I2C_CODE
```

I2C memory and pointer classes.

Definition at line 359 of file Compiler\_Cfg.h.

### 6.2.3.49 LIN\_CODE

```
#define LIN_CODE
```

LIN memory and pointer classes.

Definition at line 375 of file Compiler\_Cfg.h.

### 6.2.3.50 LIN\_43\_LLCE\_CODE

```
#define LIN_43_LLCE_CODE
```

LIN\_43\_LLCE memory and pointer classes.

Definition at line 391 of file Compiler\_Cfg.h.

### 6.2.3.51 LINIF\_CODE

```
#define LINIF_CODE
```

LIN memory and pointer classes.

Definition at line 407 of file Compiler\_Cfg.h.



#### 6.2.3.52 MCEM\_CODE

```
#define MCEM_CODE
```

MCEM memory and pointer classes.

Definition at line 423 of file Compiler\_Cfg.h.

#### 6.2.3.53 MCL\_CODE

```
#define MCL_CODE
```

MCL memory and pointer classes.

Definition at line 439 of file Compiler\_Cfg.h.

#### 6.2.3.54 MCU\_CODE

```
#define MCU_CODE
```

MCU memory and pointer classes.

Definition at line 455 of file Compiler\_Cfg.h.

#### 6.2.3.55 PMIC\_CODE

```
#define PMIC_CODE
```

PMIC memory and pointer classes.

Definition at line 471 of file Compiler\_Cfg.h.

#### 6.2.3.56 PORT\_CODE

```
#define PORT_CODE
```

PORT memory and pointer classes.

Definition at line 487 of file Compiler\_Cfg.h.

### 6.2.3.57 PWM\_CODE

```
#define PWM_CODE
```

PWM memory and pointer classes.

Definition at line 503 of file Compiler\_Cfg.h.

### 6.2.3.58 RAMTST\_CODE

```
#define RAMTST_CODE
```

RamTST memory and pointer classes.

Definition at line 520 of file Compiler\_Cfg.h.

### 6.2.3.59 SENT\_CODE

```
#define SENT_CODE
```

SENT memory and pointer classes.

Definition at line 536 of file Compiler\_Cfg.h.

### 6.2.3.60 SCHM\_CODE

```
#define SCHM_CODE
```

SchM memory and pointer classes.

Definition at line 552 of file Compiler\_Cfg.h.

### 6.2.3.61 SPI\_CODE

```
#define SPI_CODE
```

SPI memory and pointer classes.

Definition at line 568 of file Compiler\_Cfg.h.

#### 6.2.3.62 TM\_CODE

```
#define TM_CODE
```

TM memory and pointer classes.

Definition at line 584 of file Compiler\_Cfg.h.

#### 6.2.3.63 WDG\_CODE

```
#define WDG_CODE
```

WDG memory and pointer classes.

Definition at line 600 of file Compiler\_Cfg.h.

#### 6.2.3.64 WDGIF\_CODE

```
#define WDGIF_CODE
```

WDGIF memory and pointer classes.

Definition at line 616 of file Compiler\_Cfg.h.

#### 6.2.3.65 AUTOSAR\_COMSTACKDATA

```
#define AUTOSAR_COMSTACKDATA
```

Define for ComStack Data.

Definition at line 631 of file Compiler\_Cfg.h.

#### 6.2.3.66 COMPILERDEFINITION\_VENDOR\_ID

```
#define COMPILERDEFINITION_VENDOR_ID
```

Parameters that shall be published within the compiler abstraction header file and also in the module's description file.

Definition at line 58 of file CompilerDefinition.h.

### 6.2.3.67 COMTYPE\_VENDOR\_ID

```
#define COMTYPE_VENDOR_ID
```

Parameters that shall be published within the standard types header file and also in the module's description file.

Definition at line 57 of file ComStack\_Types.h.

### 6.2.3.68 NTFRSLT\_OK

```
#define NTFRSLT_OK
```

Action has been successfully finished.

General return codes for NotifResultType

Definition at line 87 of file ComStack\_Types.h.

### 6.2.3.69 NTFRSLT\_E\_NOT\_OK

```
#define NTFRSLT_E_NOT_OK
```

Message not successfully received or sent out.

General return codes for NotifResultType

Definition at line 93 of file ComStack\_Types.h.

### 6.2.3.70 NTFRSLT\_E\_TIMEOUT\_A

```
#define NTFRSLT_E_TIMEOUT_A
```

Timer N\_Ar/N\_As has passed its time-out value N\_Asmax/N\_Armx.

General return codes for NotifResultType

Definition at line 99 of file ComStack\_Types.h.

**6.2.3.71 NTFRSLT\_E\_TIMEOUT\_BS**

```
#define NTFRSLT_E_TIMEOUT_BS
```

Timer N\_Bs has passed its time-out value N\_Bsmax.

General return codes for NotifResultType

Definition at line 105 of file ComStack\_Types.h.

**6.2.3.72 NTFRSLT\_E\_TIMEOUT\_CR**

```
#define NTFRSLT_E_TIMEOUT_CR
```

Timer N\_Cr has passed its time-out value N\_Crmax.

General return codes for NotifResultType

Definition at line 111 of file ComStack\_Types.h.

**6.2.3.73 NTFRSLT\_E\_WRONG\_SN**

```
#define NTFRSLT_E_WRONG_SN
```

Unexpected sequence number (PCI.SN) value received.

General return codes for NotifResultType

Definition at line 117 of file ComStack\_Types.h.

**6.2.3.74 NTFRSLT\_E\_INVALID\_FS**

```
#define NTFRSLT_E_INVALID_FS
```

Invalid or unknown FlowStatus value has been received.

General return codes for NotifResultType

Definition at line 123 of file ComStack\_Types.h.

### 6.2.3.75 NTFRSLT\_E\_UNEXP\_PDU

```
#define NTFRSLT_E_UNEXP_PDU
```

Unexpected protocol data unit received.

General return codes for NotifResultType

Definition at line 129 of file ComStack\_Types.h.

### 6.2.3.76 NTFRSLT\_E\_WFT\_OVRN

```
#define NTFRSLT_E_WFT_OVRN
```

Flow control WAIT frame that exceeds the maximum counter N\_WFTmax received.

General return codes for NotifResultType

Definition at line 135 of file ComStack\_Types.h.

### 6.2.3.77 NTFRSLT\_E\_ABORT

```
#define NTFRSLT_E_ABORT
```

Flow control (FC) N\_PDU with FlowStatus = OVFLW received.

General return codes for NotifResultType

Definition at line 141 of file ComStack\_Types.h.

### 6.2.3.78 NTFRSLT\_E\_NO\_BUFFER

```
#define NTFRSLT_E_NO_BUFFER
```

Indicates an abort of a transmission.

General return codes for NotifResultType

Definition at line 147 of file ComStack\_Types.h.

### 6.2.3.79 NTFRSLT\_E\_CANCELATION\_OK

```
#define NTFRSLT_E_CANCELATION_OK
```

Requested cancellation has been executed.

General return codes for NotifResultType

Definition at line 153 of file ComStack\_Types.h.

### 6.2.3.80 NTFRSLT\_E\_CANCELATION\_NOT\_OK

```
#define NTFRSLT_E_CANCELATION_NOT_OK
```

Request cancellation has not been executed Due to an internal error the requested cancelation has not been executed. This will happen e.g. if the to be canceled transmission has been executed already.

General return codes for NotifResultType

Definition at line 161 of file ComStack\_Types.h.

### 6.2.3.81 NTFRSLT\_PARAMETER\_OK

```
#define NTFRSLT_PARAMETER_OK
```

The parameter change request has been successfully executed.

General return codes for NotifResultType

Definition at line 167 of file ComStack\_Types.h.

### 6.2.3.82 NTFRSLT\_E\_PARAMETER\_NOT\_OK

```
#define NTFRSLT_E_PARAMETER_NOT_OK
```

The request for the change of the parameter did not complete successfully.

General return codes for NotifResultType

Definition at line 173 of file ComStack\_Types.h.

### 6.2.3.83 NTFRSLT\_E\_RX\_ON

```
#define NTFRSLT_E_RX_ON
```

The parameter change request not executed successfully due to an ongoing reception.

General return codes for NotifResultType

Definition at line 179 of file ComStack\_Types.h.

### 6.2.3.84 NTFRSLT\_E\_VALUE\_NOT\_OK

```
#define NTFRSLT_E_VALUE_NOT_OK
```

The parameter change request not executed successfully due to a wrong value.

General return codes for NotifResultType

Definition at line 185 of file ComStack\_Types.h.

### 6.2.3.85 CRC\_MEMMAP\_VENDOR\_ID

```
#define CRC_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Crc\_MemMap.h.

### 6.2.3.86 MEMMAP\_ERROR [7/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Crc\_MemMap.h.



### 6.2.3.87 CRYPTO\_MEMMAP\_VENDOR\_ID

```
#define CRYPTO_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Crypto\_MemMap.h.

### 6.2.3.88 MEMMAP\_ERROR [8/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Crypto\_MemMap.h.

### 6.2.3.89 CSEC\_MEMMAP\_VENDOR\_ID

```
#define CSEC_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Csec\_MemMap.h.

### 6.2.3.90 MEMMAP\_ERROR [9/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Csec\_MemMap.h.

### 6.2.3.91 DEM\_MEMMAP\_VENDOR\_ID

```
#define DEM_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Dem\_MemMap.h.

### 6.2.3.92 MEMMAP\_ERROR [10/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Dem\_MemMap.h.

### 6.2.3.93 DET\_MEMMAP\_VENDOR\_ID

```
#define DET_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Det\_MemMap.h.

### 6.2.3.94 MEMMAP\_ERROR [11/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Det\_MemMap.h.

### 6.2.3.95 DIO\_MEMMAP\_VENDOR\_ID

```
#define DIO_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Dio\_MemMap.h.

### 6.2.3.96 MEMMAP\_ERROR [12/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Dio\_MemMap.h.

### 6.2.3.97 ECUM\_MEMMAP\_VENDOR\_ID

```
#define ECUM_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Ecum\_MemMap.h.

### 6.2.3.98 MEMMAP\_ERROR [13/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Ecum\_MemMap.h.

### 6.2.3.99 EEP\_MEMMAP\_VENDOR\_ID

```
#define EEP_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Eep\_MemMap.h.

### 6.2.3.100 MEMMAP\_ERROR [14/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Eep\_MemMap.h.

### 6.2.3.101 ETH\_43\_PFE\_MEMMAP\_VENDOR\_ID

```
#define ETH_43_PFE_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Eth\_43\_PFE\_MemMap.h.

### 6.2.3.102 MEMMAP\_ERROR [15/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Eth\_43\_PFE\_MemMap.h.

### 6.2.3.103 ETH\_GENERALTYPES\_AR\_RELEASE\_MAJOR\_VERSION

```
#define ETH_GENERALTYPES_AR_RELEASE_MAJOR_VERSION
```

Parameters that shall be published within the modules header file. The integration of incompatible files shall be avoided.

Definition at line 59 of file Eth\_GeneralTypes.h.

### 6.2.3.104 ETH\_MEMMAP\_VENDOR\_ID

```
#define ETH_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Eth\_MemMap.h.

### 6.2.3.105 MEMMAP\_ERROR [16/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Eth\_MemMap.h.

### 6.2.3.106 ETHSWITCH\_43\_SJA1105P\_MEMMAP\_VENDOR\_ID

```
#define ETHSWITCH_43_SJA1105P_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file EthSwitch\_43\_SJA1105P\_MemMap.h.

**6.2.3.107 MEMMAP\_ERROR [17/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file EthSwitch\_43\_SJA1105P\_MemMap.h.

**6.2.3.108 ETHSWT\_43\_SJA11XX\_MEMMAP\_VENDOR\_ID**

```
#define ETHSWT_43_SJA11XX_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file EthSwt\_43\_SJA11XX\_MemMap.h.

**6.2.3.109 MEMMAP\_ERROR [18/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file EthSwt\_43\_SJA11XX\_MemMap.h.

**6.2.3.110 ETHTRCV\_43\_PHY\_MEMMAP\_VENDOR\_ID**

```
#define ETHTRCV_43_PHY_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file EthTrcv\_43\_PHY\_MemMap.h.

**6.2.3.111 MEMMAP\_ERROR [19/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file EthTrcv\_43\_PHY\_MemMap.h.

### 6.2.3.112 ETHTRCV\_43\_TJA110X\_MEMMAP\_VENDOR\_ID

```
#define ETHTRCV_43_TJA110X_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file EthTrcv\_43\_TJA110X\_MemMap.h.

### 6.2.3.113 MEMMAP\_ERROR [20/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file EthTrcv\_43\_TJA110X\_MemMap.h.

### 6.2.3.114 FEE\_MEMMAP\_VENDOR\_ID

```
#define FEE_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Fee\_MemMap.h.

### 6.2.3.115 MEMMAP\_ERROR [21/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Fee\_MemMap.h.

### 6.2.3.116 FLS\_MEMMAP\_VENDOR\_ID

```
#define FLS_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Fls\_MemMap.h.

**6.2.3.117 MEMMAP\_ERROR [22/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Fls\_MemMap.h.

**6.2.3.118 FR\_43\_LLCE\_MEMMAP\_VENDOR\_ID**

```
#define FR_43_LLCE_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Fr\_43\_LLCE\_MemMap.h.

**6.2.3.119 MEMMAP\_ERROR [23/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Fr\_43\_LLCE\_MemMap.h.

**6.2.3.120 FR\_GENERALTYPES\_AR\_RELEASE\_MAJOR\_VERSION**

```
#define FR_GENERALTYPES_AR_RELEASE_MAJOR_VERSION
```

Parameters that shall be published within the modules header file. The integration of incompatible files shall be avoided.

Definition at line 59 of file Fr\_GeneralTypes.h.

**6.2.3.121 FR\_CIDX\_GDCYCLE**

```
#define FR_CIDX_GDCYCLE
```

Macros which can be passed into Fr\_ReadCCConfig as parameter Fr\_ConfigParamIdx.

Each macro (index) uniquely identifies a configuration parameter which value can be read out of the controllers configuration using Fr\_ReadCCConfig.

Definition at line 72 of file Fr\_GeneralTypes.h.

### 6.2.3.122 FR\_SLOTMODE\_SINGLE

```
#define FR_SLOTMODE_SINGLE
```

This macro is used for backward compatibility with Autosar 3.0 definition of Fr\_SlotModeType.

Definition at line 209 of file Fr\_GeneralTypes.h.

### 6.2.3.123 FR\_MEMMAP\_VENDOR\_ID

```
#define FR_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Fr\_MemMap.h.

### 6.2.3.124 MEMMAP\_ERROR [24/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Fr\_MemMap.h.

### 6.2.3.125 GPT\_MEMMAP\_VENDOR\_ID

```
#define GPT_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Gpt\_MemMap.h.

### 6.2.3.126 MEMMAP\_ERROR [25/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Gpt\_MemMap.h.



**6.2.3.127 I2C\_MEMMAP\_VENDOR\_ID**

```
#define I2C_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file I2c\_MemMap.h.

**6.2.3.128 MEMMAP\_ERROR [26/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file I2c\_MemMap.h.

**6.2.3.129 I2S\_MEMMAP\_VENDOR\_ID**

```
#define I2S_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file I2s\_MemMap.h.

**6.2.3.130 MEMMAP\_ERROR [27/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file I2s\_MemMap.h.

**6.2.3.131 ICU\_MEMMAP\_VENDOR\_ID**

```
#define ICU_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Icu\_MemMap.h.

### 6.2.3.132 MEMMAP\_ERROR [28/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Icu\_MemMap.h.

### 6.2.3.133 ISELED\_MEMMAP\_VENDOR\_ID

```
#define ISELED_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Iseled\_MemMap.h.

### 6.2.3.134 MEMMAP\_ERROR [29/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Iseled\_MemMap.h.

### 6.2.3.135 LIN\_43\_LLCE\_MEMMAP\_VENDOR\_ID

```
#define LIN_43_LLCE_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Lin\_43\_LLCE\_MemMap.h.

### 6.2.3.136 MEMMAP\_ERROR [30/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Lin\_43\_LLCE\_MemMap.h.

**6.2.3.137 LIN\_GENERALTYPES\_AR\_RELEASE\_MAJOR\_VERSION**

```
#define LIN_GENERALTYPES_AR_RELEASE_MAJOR_VERSION
```

Parameters that shall be published within the modules header file. The integration of incompatible files shall be avoided.

Definition at line 60 of file Lin\_GeneralTypes.h.

**6.2.3.138 LIN\_MEMMAP\_VENDOR\_ID**

```
#define LIN_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Lin\_MemMap.h.

**6.2.3.139 MEMMAP\_ERROR [31/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Lin\_MemMap.h.

**6.2.3.140 MCAL\_DATA\_SYNC\_BARRIER**

```
#define MCAL_DATA_SYNC_BARRIER( )
```

Data Synchronization Barrier (DSB) completes when all instructions before this instruction complete.

Definition at line 531 of file Mcal.h.

**6.2.3.141 MCAL\_INSTRUCTION\_SYNC\_BARRIER**

```
#define MCAL_INSTRUCTION_SYNC_BARRIER( )
```

flushes the pipeline in the processor, so that all instructions following the ISB are fetched from cache or memory, after the ISB has been completed.

Definition at line 535 of file Mcal.h.

### 6.2.3.142 MCEM\_MEMMAP\_VENDOR\_ID

```
#define MCEM_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Mcem\_MemMap.h.

### 6.2.3.143 MEMMAP\_ERROR [32/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Mcem\_MemMap.h.

### 6.2.3.144 MCL\_MEMMAP\_VENDOR\_ID

```
#define MCL_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Mcl\_MemMap.h.

### 6.2.3.145 MEMMAP\_ERROR [33/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Mcl\_MemMap.h.

### 6.2.3.146 MCU\_MEMMAP\_VENDOR\_ID

```
#define MCU_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Mcu\_MemMap.h.

**6.2.3.147 MEMMAP\_ERROR [34/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Mcu\_MemMap.h.

**6.2.3.148 OCU\_MEMMAP\_VENDOR\_ID**

```
#define OCU_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Ocu\_MemMap.h.

**6.2.3.149 MEMMAP\_ERROR [35/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Ocu\_MemMap.h.

**6.2.3.150 PLATFORM\_MEMMAP\_VENDOR\_ID**

```
#define PLATFORM_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Platform\_MemMap.h.

**6.2.3.151 MEMMAP\_ERROR [36/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Platform\_MemMap.h.

### 6.2.3.152 PLATFORM\_TYPES\_VENDOR\_ID

```
#define PLATFORM_TYPES_VENDOR_ID
```

Note

It is not allowed to add any extension to this file. Any extension invalidates the AUTOSAR conformity

Definition at line 66 of file PlatformTypes.h.

### 6.2.3.153 CPU\_TYPE\_8

```
#define CPU_TYPE_8
```

8bit Type Processor

Definition at line 89 of file PlatformTypes.h.

### 6.2.3.154 CPU\_TYPE\_16

```
#define CPU_TYPE_16
```

16bit Type Processor

Definition at line 95 of file PlatformTypes.h.

### 6.2.3.155 CPU\_TYPE\_32

```
#define CPU_TYPE_32
```

32bit Type Processor

Definition at line 101 of file PlatformTypes.h.

**6.2.3.156 CPU\_TYPE\_64**

```
#define CPU_TYPE_64
```

64bit Type Processor

Definition at line 107 of file PlatformTypes.h.

**6.2.3.157 MSB\_FIRST**

```
#define MSB_FIRST
```

MSB First Processor.

Definition at line 113 of file PlatformTypes.h.

**6.2.3.158 LSB\_FIRST**

```
#define LSB_FIRST
```

LSB First Processor.

Definition at line 119 of file PlatformTypes.h.

**6.2.3.159 HIGH\_BYTE\_FIRST**

```
#define HIGH_BYTE_FIRST
```

HIGH\_BYTE\_FIRST Processor.

Definition at line 125 of file PlatformTypes.h.

**6.2.3.160 LOW\_BYTE\_FIRST**

```
#define LOW_BYTE_FIRST
```

LOW\_BYTE\_FIRST Processor.

Definition at line 131 of file PlatformTypes.h.

### 6.2.3.161 CPU\_\_TYPE

```
#define CPU__TYPE
```

Processor type.

Definition at line 137 of file PlatformTypes.h.

### 6.2.3.162 CPU\_\_BIT\_ORDER

```
#define CPU__BIT_ORDER
```

Bit order on register level.

Definition at line 146 of file PlatformTypes.h.

### 6.2.3.163 CPU\_\_BYTE\_ORDER

```
#define CPU__BYTE_ORDER
```

The byte order on memory level shall be indicated in the platform types header file using the symbol CPU\_\_BYTE\_ORDER.

Definition at line 153 of file PlatformTypes.h.

### 6.2.3.164 TRUE

```
#define TRUE
```

Boolean true value.

Definition at line 161 of file PlatformTypes.h.

### 6.2.3.165 FALSE

```
#define FALSE
```

Boolean false value.

Definition at line 176 of file PlatformTypes.h.



**6.2.3.166 PMIC\_MEMMAP\_VENDOR\_ID**

```
#define PMIC_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Pmic\_MemMap.h.

**6.2.3.167 MEMMAP\_ERROR [37/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Pmic\_MemMap.h.

**6.2.3.168 PORT\_MEMMAP\_VENDOR\_ID**

```
#define PORT_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Port\_MemMap.h.

**6.2.3.169 MEMMAP\_ERROR [38/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Port\_MemMap.h.

**6.2.3.170 PWM\_MEMMAP\_VENDOR\_ID**

```
#define PWM_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Pwm\_MemMap.h.

### 6.2.3.171 MEMMAP\_ERROR [39/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Pwm\_MemMap.h.

### 6.2.3.172 QDEC\_MEMMAP\_VENDOR\_ID

```
#define QDEC_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Qdec\_MemMap.h.

### 6.2.3.173 MEMMAP\_ERROR [40/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Qdec\_MemMap.h.

### 6.2.3.174 MCAL\_AXBS\_REG\_PROT\_AVAILABLE

```
#define MCAL_AXBS_REG_PROT_AVAILABLE
```

Macros defined for the IPVs that are protected.

Definition at line 80 of file Reg\_eSys.h.

### 6.2.3.175 AXBS\_PROT\_MEM\_U32

```
#define AXBS_PROT_MEM_U32
```

Macros defined for the protection size.

Definition at line 102 of file Reg\_eSys.h.

**6.2.3.176 RLM\_REG\_WRITE8**

```
#define RLM_REG_WRITE8(  
    address,  
    value )
```

8 bits memory write macro

Definition at line 87 of file RegLockMacros.h.

**6.2.3.177 RLM\_REG\_WRITE16**

```
#define RLM_REG_WRITE16(  
    address,  
    value )
```

16 bits memory write macro.

Definition at line 91 of file RegLockMacros.h.

**6.2.3.178 RLM\_REG\_WRITE32**

```
#define RLM_REG_WRITE32(  
    address,  
    value )
```

32 bits memory write macro.

Definition at line 95 of file RegLockMacros.h.

**6.2.3.179 RLM\_REG\_READ8**

```
#define RLM_REG_READ8(  
    address )
```

8 bits memory read macro.

Definition at line 101 of file RegLockMacros.h.

### 6.2.3.180 RLM\_REG\_READ16

```
#define RLM_REG_READ16(  
    address )
```

16 bits memory read macro.

Definition at line 105 of file RegLockMacros.h.

### 6.2.3.181 RLM\_REG\_READ32

```
#define RLM_REG_READ32(  
    address )
```

32 bits memory read macro.

Definition at line 109 of file RegLockMacros.h.

### 6.2.3.182 RLM\_REG\_BIT\_CLEAR8

```
#define RLM_REG_BIT_CLEAR8(  
    address,  
    mask )
```

8 bits bits clearing macro.

Definition at line 114 of file RegLockMacros.h.

### 6.2.3.183 RLM\_REG\_BIT\_CLEAR16

```
#define RLM_REG_BIT_CLEAR16(  
    address,  
    mask )
```

16 bits bits clearing macro.

Definition at line 118 of file RegLockMacros.h.

**6.2.3.184 RLM\_REG\_BIT\_CLEAR32**

```
#define RLM_REG_BIT_CLEAR32(  
    address,  
    mask )
```

32 bits clearing macro.

Definition at line 122 of file RegLockMacros.h.

**6.2.3.185 RLM\_REG\_BIT\_GET8**

```
#define RLM_REG_BIT_GET8(  
    address,  
    mask )
```

8 bits getting macro.

Definition at line 128 of file RegLockMacros.h.

**6.2.3.186 RLM\_REG\_BIT\_GET16**

```
#define RLM_REG_BIT_GET16(  
    address,  
    mask )
```

16 bits getting macro.

Definition at line 132 of file RegLockMacros.h.

**6.2.3.187 RLM\_REG\_BIT\_GET32**

```
#define RLM_REG_BIT_GET32(  
    address,  
    mask )
```

32 bits getting macro.

Definition at line 136 of file RegLockMacros.h.

### 6.2.3.188 RLM\_REG\_BIT\_SET8

```
#define RLM_REG_BIT_SET8(  
    address,  
    mask )
```

8 bits bits setting macro.

Definition at line 142 of file RegLockMacros.h.

### 6.2.3.189 RLM\_REG\_BIT\_SET16

```
#define RLM_REG_BIT_SET16(  
    address,  
    mask )
```

16 bits bits setting macro.

Definition at line 146 of file RegLockMacros.h.

### 6.2.3.190 RLM\_REG\_BIT\_SET32

```
#define RLM_REG_BIT_SET32(  
    address,  
    mask )
```

32 bits bits setting macro.

Definition at line 150 of file RegLockMacros.h.

### 6.2.3.191 RLM\_REG\_RMW8

```
#define RLM_REG_RMW8(  
    address,  
    mask,  
    value )
```

8 bit clear bits and set with new value

Note

In the current implementation, it is caller's (user's) responsibility to make sure that value has only "mask" bits set -  $(value \& \sim mask) == 0$

Definition at line 158 of file RegLockMacros.h.

**6.2.3.192 RLM\_REG\_RMW16**

```
#define RLM_REG_RMW16(
    address,
    mask,
    value )
```

16 bit clear bits and set with new value

## Note

In the current implementation, it is caller's (user's) responsibility to make sure that value has only "mask" bits set -  $(value \& \sim mask) == 0$

Definition at line 164 of file RegLockMacros.h.

**6.2.3.193 RLM\_REG\_RMW32**

```
#define RLM_REG_RMW32(
    address,
    mask,
    value )
```

32 bit clear bits and set with new value

## Note

In the current implementation, it is caller's (user's) responsibility to make sure that value has only "mask" bits set -  $(value \& \sim mask) == 0$

Definition at line 170 of file RegLockMacros.h.

**6.2.3.194 SLBR\_SET\_BIT\_8BIT\_REG\_MASK\_U8**

```
#define SLBR_SET_BIT_8BIT_REG_MASK_U8
```

Mask for setting SLB bit(s) in a SLBR register (for 8/16/32bit registers)

Definition at line 187 of file RegLockMacros.h.

### 6.2.3.195 SLBR\_CLR\_BIT\_8BIT\_REG\_MASK\_U8

```
#define SLBR_CLR_BIT_8BIT_REG_MASK_U8
```

Mask for clearing WE bit(s) in a SLBR register (for 8/16/32bit registers)

Definition at line 194 of file RegLockMacros.h.

### 6.2.3.196 SLBR\_GET\_BIT\_8BIT\_REG\_MASK\_U8

```
#define SLBR_GET_BIT_8BIT_REG_MASK_U8
```

Mask for getting SLB bit(s) in a SLBR register (for 8/16/32bit registers)

Definition at line 201 of file RegLockMacros.h.

### 6.2.3.197 SLBR\_XOR\_8BIT\_REG\_MASK\_U8

```
#define SLBR_XOR_8BIT_REG_MASK_U8
```

Masks for inverting bit positions in a SLBR register.

Definition at line 213 of file RegLockMacros.h.

### 6.2.3.198 MODULO\_4\_BIT\_MASK\_U32

```
#define MODULO_4_BIT_MASK_U32
```

Mask used for getting the alignment of an address inside a 32 bit word.

Definition at line 234 of file RegLockMacros.h.

### 6.2.3.199 MIRRORED\_ADDR\_OFFSET\_U32

```
#define MIRRORED_ADDR_OFFSET_U32
```

Offset to REG\_PROT mirrored registers area of an IP module.

Definition at line 273 of file RegLockMacros.h.



**6.2.3.200 SLBR\_ADDR\_OFFSET\_U32**

```
#define SLBR_ADDR_OFFSET_U32
```

Offset to baseAddress of the SLBR registers area of an IP module.

Definition at line 313 of file RegLockMacros.h.

**6.2.3.201 SLBR\_ADDR32**

```
#define SLBR_ADDR32(  
    baseAddr,  
    regAddr,  
    prot_mem )
```

Macro for getting the address of a lockable register's corresponding SLBR register.

Definition at line 320 of file RegLockMacros.h.

**6.2.3.202 GCR\_OFFSET\_U32**

```
#define GCR_OFFSET_U32
```

Offset to baseAddress of the REG\_PROT GCR register of an IP module.

Definition at line 372 of file RegLockMacros.h.

**6.2.3.203 REGPROT\_GCR\_HLB\_MASK\_U32**

```
#define REGPROT_GCR_HLB_MASK_U32
```

REG\_PROT GCR bit masks.

Definition at line 378 of file RegLockMacros.h.

### 6.2.3.204 REGPROT\_GCR\_HLB\_POS\_U32

```
#define REGPROT_GCR_HLB_POS_U32
```

REG\_PROT GCR bit positions.

Definition at line 384 of file RegLockMacros.h.

### 6.2.3.205 REG\_SET\_SOFT\_LOCK8

```
#define REG_SET_SOFT_LOCK8(  
    baseAddr,  
    regAddr,  
    prot_mem )
```

Soft locks a register by setting it's corresponding soft lock bit.

Based on the address of the register to be soft locked and on the address of the IP where the register belongs to, the corresponding soft lock bit is set

Parameters

in	<i>baseAddr</i>	- base address of the IP the register belongs to
in	<i>regAddr</i>	- address of the register to soft lock
in	<i>prot_mem</i>	- protection size of the IP

Returns

void

Definition at line 414 of file RegLockMacros.h.

### 6.2.3.206 REG\_CLR\_SOFT\_LOCK8

```
#define REG_CLR_SOFT_LOCK8(  
    baseAddr,  
    regAddr,  
    prot_mem )
```

Soft unlocks a register by clearing it's corresponding soft lock bit.

Based on the address of the register to be soft unlocked and on the address of the IP where the register belongs to, the corresponding soft lock bit is cleared

Parameters

in	<i>baseAddr</i>	- base address of the IP the register belongs to
in	<i>regAddr</i>	- address of the register to soft unlock
in	<i>prot_mem</i>	- protection size of the IP

Returns

void

Definition at line 457 of file RegLockMacros.h.

### 6.2.3.207 REG\_GET\_SOFT\_LOCK8

```
#define REG_GET_SOFT_LOCK8(
    baseAddr,
    regAddr,
    prot_mem )
```

Reads the status of the soft lock bit of a register.

Parameters

in	<i>baseAddr</i>	- base address of the IP the register belongs to
in	<i>regAddr</i>	- address of the register for which to get soft lock bit status
in	<i>prot_mem</i>	- protection size of the IP

Returns

uint8 - 1 if the register's soft lock is enabled

- 0 if the register's soft lock is disabled

Definition at line 497 of file RegLockMacros.h.

### 6.2.3.208 REG\_BIT\_SET\_LOCK8

```
#define REG_BIT_SET_LOCK8(
    baseAddr,
    regAddr,
    prot_mem,
    mask )
```

Sets one bit in a 8 bit register and locks the register automatically.

Clears first the corresponding soft lock bit and writes the REG\_PROT mirrored value of the register the bit belongs to, which automatically soft locks the register

Parameters

in	<i>baseAddr</i>	- base address of the IP the bit belongs to
in	<i>regAddr</i>	- address of the register the bit belongs to
in	<i>prot_mem</i>	- protection size of the IP
in	<i>mask</i>	- 8 bit mask of the bit

Returns

void

Definition at line 542 of file RegLockMacros.h.

### 6.2.3.209 REG\_BIT\_SET\_LOCK16

```
#define REG_BIT_SET_LOCK16(  
    baseAddr,  
    regAddr,  
    prot_mem,  
    mask )
```

Sets one bit in a 16 bit register and locks the register automatically.

Clears first the corresponding soft lock bit and writes the REG\_PROT mirrored value of the register the bit belongs to, which automatically soft locks the register

Parameters

in	<i>baseAddr</i>	- base address of the IP the bit belongs to
in	<i>regAddr</i>	- address of the register the bit belongs to
in	<i>prot_mem</i>	- protection size of the IP
in	<i>mask</i>	- 8 bit mask of the bit

Returns

void

Definition at line 568 of file RegLockMacros.h.

## 6.2.3.210 REG\_BIT\_SET\_LOCK32

```
#define REG_BIT_SET_LOCK32(  
    baseAddr,  
    regAddr,  
    prot_mem,  
    mask )
```

Sets one bit in a 32 bit register and locks the register automatically.

Clears first the corresponding soft lock bit and writes the REG\_PROT mirrored value of the register the bit belongs to, which automatically soft locks the register

Parameters

in	<i>baseAddr</i>	- base address of the IP the bit belongs to
in	<i>regAddr</i>	- address of the register the bit belongs to
in	<i>prot_mem</i>	- protection size of the IP
in	<i>mask</i>	- 8 bit mask of the bit

Returns

void

Definition at line 594 of file RegLockMacros.h.

## 6.2.3.211 REG\_BIT\_CLEAR\_LOCK8

```
#define REG_BIT_CLEAR_LOCK8(  
    baseAddr,  
    regAddr,  
    prot_mem,  
    mask )
```

Clears one bit in a 8 bit register and locks the register automatically.

Clears first the corresponding soft lock bit and writes the REG\_PROT mirrored value of the register the bit belongs to, which automatically soft locks the register

Parameters

in	<i>baseAddr</i>	- base address of the IP the bit belongs to
in	<i>regAddr</i>	- address of the register the bit belongs to
in	<i>prot_mem</i>	- protection size of the IP
in	<i>mask</i>	- 8 bit mask of the bit

Returns

void

Definition at line 620 of file RegLockMacros.h.

### 6.2.3.212 REG\_BIT\_CLEAR\_LOCK16

```
#define REG_BIT_CLEAR_LOCK16(  
    baseAddr,  
    regAddr,  
    prot_mem,  
    mask )
```

Clears one bit in a 16 bit register and locks the register automatically.

Clears first the corresponding soft lock bit and writes the REG\_PROT mirrored value of the register the bit belongs to, which automatically soft locks the register

Parameters

in	<i>baseAddr</i>	- base address of the IP the bit belongs to
in	<i>regAddr</i>	- address of the register the bit belongs to
in	<i>prot_mem</i>	- protection size of the IP
in	<i>mask</i>	- 8 bit mask of the bit

Returns

void

Definition at line 646 of file RegLockMacros.h.

### 6.2.3.213 REG\_BIT\_CLEAR\_LOCK32

```
#define REG_BIT_CLEAR_LOCK32(  
    baseAddr,  
    regAddr,  
    prot_mem,  
    mask )
```

Clears one bit in a 32 bit register and locks the register automatically.

Clears first the corresponding soft lock bit and writes the REG\_PROT mirrored value of the register the bit belongs to, which automatically soft locks the register

## Module Documentation

### Parameters

in	<i>baseAddr</i>	- base address of the IP the bit belongs to
in	<i>regAddr</i>	- address of the register the bit belongs to
in	<i>prot_mem</i>	- protection size of the IP
in	<i>mask</i>	- 8 bit mask of the bit

### Returns

void

Definition at line 672 of file RegLockMacros.h.

### 6.2.3.214 REG\_WRITE\_LOCK8

```
#define REG_WRITE_LOCK8(  
    baseAddr,  
    regAddr,  
    prot_mem,  
    value )
```

Writes the content of a 8 bit register and locks it automatically.

Clears first the corresponding soft lock bit and writes the REG\_PROT mirrored value of the register, which automatically soft locks the register

### Parameters

in	<i>baseAddr</i>	- base address of the IP the register belongs to
in	<i>regAddr</i>	- address of the register to write and soft lock
in	<i>prot_mem</i>	- protection size of the IP
in	<i>value</i>	- 8 bit value the register will be written with

### Returns

void

Definition at line 698 of file RegLockMacros.h.



**6.2.3.215 REG\_WRITE\_LOCK16**

```
#define REG_WRITE_LOCK16(
    baseAddr,
    regAddr,
    prot_mem,
    value )
```

Writes the content of a 16 bit register and locks it automatically.

Clears first the corresponding soft lock bit and writes the REG\_PROT mirrored value of the register, which automatically soft locks the register

Parameters

in	<i>baseAddr</i>	- base address of the IP the register belongs to
in	<i>regAddr</i>	- address of the register to write and soft lock
in	<i>prot_mem</i>	- protection size of the IP
in	<i>value</i>	- 16 bit value the register will be written with

Returns

void

Definition at line 724 of file RegLockMacros.h.

**6.2.3.216 REG\_WRITE\_LOCK32**

```
#define REG_WRITE_LOCK32(
    baseAddr,
    regAddr,
    prot_mem,
    value )
```

Writes the content of a 32 bit register and locks it automatically.

Clears first the corresponding soft lock bit and writes the REG\_PROT mirrored value of the register, which automatically soft locks the register

Parameters

in	<i>baseAddr</i>	- base address of the IP the register belongs to
in	<i>regAddr</i>	- address of the register to write and soft lock
in	<i>prot_mem</i>	- protection size of the IP
in	<i>value</i>	- 32 bit value the register will be written with

## Module Documentation

Returns

void

Definition at line 750 of file RegLockMacros.h.

### 6.2.3.217 REG\_RMW\_LOCK8

```
#define REG_RMW_LOCK8(  
    baseAddr,  
    regAddr,  
    prot_mem,  
    mask,  
    value )
```

Clears the content of a 8 bit register, writes it with the value in 'value' parameter masked with the one in 'mask' parameter and locks it automatically.

Clears first the corresponding soft lock bit and writes the REG\_PROT mirrored value of the register, which automatically soft locks the register

Parameters

in	<i>baseAddr</i>	- base address of the IP the register belongs to
in	<i>regAddr</i>	- address of the register to write and soft lock
in	<i>prot_mem</i>	- protection size of the IP
in	<i>mask</i>	- 8 bit mask the register will be written with
in	<i>value</i>	- 8 bit value the register will be written with

Returns

void

Definition at line 780 of file RegLockMacros.h.

### 6.2.3.218 REG\_RMW\_LOCK16

```
#define REG_RMW_LOCK16(  
    baseAddr,  
    regAddr,  
    prot_mem,  
    mask,  
    value )
```

Clears the content of a 16 bit register, writes it with the value in 'value' parameter masked with the one in 'mask' parameter and locks it automatically.

Clears first the corresponding soft lock bit and writes the REG\_PROT mirrored value of the register, which automatically soft locks the register

Parameters

in	<i>baseAddr</i>	- base address of the IP the register belongs to
in	<i>regAddr</i>	- address of the register to write and soft lock
in	<i>prot_mem</i>	- protection size of the IP
in	<i>mask</i>	- 16 bit mask the register will be written with
in	<i>value</i>	- 16 bit value the register will be written with

Returns

void

Definition at line 811 of file RegLockMacros.h.

#### 6.2.3.219 REG\_RMW\_LOCK32

```
#define REG_RMW_LOCK32 (
    baseAddr,
    regAddr,
    prot_mem,
    mask,
    value )
```

Clears the content of a 32 bit register, writes it with the value in 'value' parameter masked with the one in 'mask' parameter and locks it automatically.

Clears first the corresponding soft lock bit and writes the REG\_PROT mirrored value of the register, which automatically soft locks the register

Parameters

in	<i>baseAddr</i>	- base address of the IP the register belongs to
in	<i>regAddr</i>	- address of the register to write and soft lock
in	<i>prot_mem</i>	- protection size of the IP
in	<i>mask</i>	- 32 bit mask the register will be written with
in	<i>value</i>	- 32 bit value the register will be written with

## Module Documentation

Returns

void

Definition at line 840 of file RegLockMacros.h.

### 6.2.3.220 SET\_HARD\_LOCK

```
#define SET_HARD_LOCK(  
    baseAddr,  
    prot_mem )
```

Sets the hardlock bit of an IP module.

Parameters

in	<i>baseAddr</i>	- base address of the IP to be hard locked
in	<i>prot_mem</i>	- the protection size of the IP

Returns

void

Definition at line 852 of file RegLockMacros.h.

### 6.2.3.221 GET\_HARD\_LOCK

```
#define GET_HARD_LOCK(  
    baseAddr,  
    prot_mem )
```

Reads the Hard Lock bit of an IP module.

Parameters

in	<i>baseAddr</i>	- base address of the IP for which hard lock status is read
in	<i>prot_mem</i>	- the protection size of the IP

Returns

uint8 - 1 if hard lock is enabled

- 0 if hard lock is disabled

Definition at line 865 of file RegLockMacros.h.

### 6.2.3.222 SET\_USER\_ACCESS\_ALLOWED

```
#define SET_USER_ACCESS_ALLOWED(  
    baseAddr,  
    prot_mem )
```

Sets the User Access Allowed bit of an IP module.

Parameters

in	<i>baseAddr</i>	- base address of the IP for which UAA bit is set
in	<i>prot_mem</i>	- the protection size of the IP

Returns

void

Definition at line 888 of file RegLockMacros.h.

### 6.2.3.223 CLR\_USER\_ACCESS\_ALLOWED

```
#define CLR_USER_ACCESS_ALLOWED(  
    baseAddr,  
    prot_mem )
```

Clears the User Access Allowed bit of an IP module.

Parameters

in	<i>baseAddr</i>	- base address of the IP for which UAA bit is cleared
in	<i>prot_mem</i>	- the protection size of the IP

Returns

void

Definition at line 913 of file RegLockMacros.h.

### 6.2.3.224 GET\_USER\_ACCESS\_ALLOWED

```
#define GET_USER_ACCESS_ALLOWED(
    baseAddr,
    prot_mem )
```

Reads the User Access Allowed bit of an IP module.

Parameters

in	<i>baseAddr</i>	- base address of the IP for which UAA is read
in	<i>prot_mem</i>	- the protection size of the IP

Returns

- uint8 - 1 if User Access Allow is enabled
- 0 if User Access Allow is disabled

Definition at line 938 of file RegLockMacros.h.

### 6.2.3.225 RM\_MEMMAP\_VENDOR\_ID

```
#define RM_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Rm\_MemMap.h.

### 6.2.3.226 MEMMAP\_ERROR [41/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Rm\_MemMap.h.

### 6.2.3.227 RTE\_MEMMAP\_VENDOR\_ID

```
#define RTE_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Rte\_MemMap.h.

**6.2.3.228 MEMMAP\_ERROR [42/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Rte\_MemMap.h.

**6.2.3.229 SBC\_FS26\_MEMMAP\_VENDOR\_ID**

```
#define SBC_FS26_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Sbc\_fs26\_MemMap.h.

**6.2.3.230 MEMMAP\_ERROR [43/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Sbc\_fs26\_MemMap.h.

**6.2.3.231 SENT\_MEMMAP\_VENDOR\_ID**

```
#define SENT_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Sent\_MemMap.h.

**6.2.3.232 MEMMAP\_ERROR [44/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Sent\_MemMap.h.

### 6.2.3.233 SPI\_MEMMAP\_VENDOR\_ID

```
#define SPI_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Spi\_MemMap.h.

### 6.2.3.234 MEMMAP\_ERROR [45/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Spi\_MemMap.h.

### 6.2.3.235 STD\_VENDOR\_ID

```
#define STD_VENDOR_ID
```

Include compiler abstraction.

Parameters that shall be published within the standard types header file and also in the module's description file

Definition at line 64 of file StandardTypes.h.

### 6.2.3.236 STD\_HIGH

```
#define STD_HIGH
```

Physical state 5V or 3.3V.

Definition at line 98 of file StandardTypes.h.

### 6.2.3.237 STD\_LOW

```
#define STD_LOW
```

Physical state 0V.

Definition at line 104 of file StandardTypes.h.



**6.2.3.238 STD\_ACTIVE**

```
#define STD_ACTIVE
```

Logical state active.

Definition at line 110 of file StandardTypes.h.

**6.2.3.239 STD\_IDLE**

```
#define STD_IDLE
```

Logical state idle.

Definition at line 116 of file StandardTypes.h.

**6.2.3.240 STD\_ON**

```
#define STD_ON
```

ON State.

Definition at line 122 of file StandardTypes.h.

**6.2.3.241 STD\_OFF**

```
#define STD_OFF
```

OFF state.

Definition at line 128 of file StandardTypes.h.

**6.2.3.242 E\_NOT\_OK**

```
#define E_NOT_OK
```

Return code for failure/error.

Definition at line 134 of file StandardTypes.h.

### 6.2.3.243 STATUSTYPEDEFINED

```
#define STATUSTYPEDEFINED
```

Because E\_OK is already defined within OSEK, the symbol E\_OK has to be shared. To avoid name clashes and redefinition problems, the symbols have to be defined in the following way (approved within implementation).

Definition at line 158 of file StandardTypes.h.

### 6.2.3.244 E\_OK

```
#define E_OK
```

Success return code.

Definition at line 162 of file StandardTypes.h.

### 6.2.3.245 UART\_MEMMAP\_VENDOR\_ID

```
#define UART_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Uart\_MemMap.h.

### 6.2.3.246 MEMMAP\_ERROR [46/48]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Uart\_MemMap.h.

### 6.2.3.247 WDG\_43\_FS26\_MEMMAP\_VENDOR\_ID

```
#define WDG_43_FS26_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Wdg\_43\_fs26\_MemMap.h.

**6.2.3.248 MEMMAP\_ERROR [47/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Wdg\_43\_fs26\_MemMap.h.

**6.2.3.249 WDG\_MEMMAP\_VENDOR\_ID**

```
#define WDG_MEMMAP_VENDOR_ID
```

Parameters that shall be published within the memory map header file and also in the module's description file.

Definition at line 53 of file Wdg\_MemMap.h.

**6.2.3.250 MEMMAP\_ERROR [48/48]**

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 77 of file Wdg\_MemMap.h.

**6.2.4 Types Reference****6.2.4.1 Can\_IdType**

```
typedef uint32 Can_IdType
```

Can\_IdType.

Represents the Identifier of an L-PDU. The two most significant bits specify the frame type: -00 CAN message with Standard CAN ID -01 CAN FD frame with Standard CAN ID -10 CAN message with Extended CAN ID -11 CAN FD frame with Extended CAN ID

Definition at line 166 of file Can\_GeneralTypes.h.

### 6.2.4.2 Can\_HwHandleType

```
typedef uint16 Can_HwHandleType
```

Can\_HwHandleType.

Represents the hardware object handles of a CAN hardware unit. For CAN hardware units with more than 255 HW objects use extended range. used by "Can\_Write" function. The driver does not distinguish between Extended and Mixed transmission modes. Extended transmission mode of operation behaves the same as Mixed mode.

Definition at line 204 of file Can\_GeneralTypes.h.

### 6.2.4.3 PduIdType

```
typedef uint16 PduIdType
```

This type serve as a unique identifier of a PDU within a software module. Allowed ranges: uint8 .. uint16.

Definition at line 69 of file ComStack\_Cfg.h.

### 6.2.4.4 PduLengthType

```
typedef uint32 PduLengthType
```

This type serve as length information of a PDU in bytes. Allowed ranges: uint8 .. uint32.

Definition at line 76 of file ComStack\_Cfg.h.

### 6.2.4.5 NotifResultType

```
typedef uint8 NotifResultType
```

Variables of this type are used to store the result status of a notification (confirmation or indication).

Definition at line 244 of file ComStack\_Types.h.

#### 6.2.4.6 NetworkHandleType

```
typedef uint8 NetworkHandleType
```

Variables of the type NetworkHandleType are used to store the identifier of a communication channel.

Definition at line 251 of file ComStack\_\_Types.h.

#### 6.2.4.7 PNCHandleType

```
typedef uint8 PNCHandleType
```

Variables of the type PNCHandleType used to store the identifier of a partial network cluster.

Definition at line 258 of file ComStack\_\_Types.h.

#### 6.2.4.8 IcomConfigIdType

```
typedef uint8 IcomConfigIdType
```

Variables of the type IcomConfigIdType defines the configuration ID. An ID of 0 is the default configuration. An ID greater than 0 shall identify a configuration for Pretended Networking.

Definition at line 289 of file ComStack\_\_Types.h.

#### 6.2.4.9 Eth\_FrameType

```
typedef uint16 Eth_FrameType
```

Frame type.

This type is used to pass the value of type or length field in the Ethernet frame header. It is 16 bits long unsigned integer.

- Values less than or equal to 1500 represent the length.
- Values greater than 1500 represent the type (i.e. 0x800 = IP).

Definition at line 379 of file Eth\_GeneralTypes.h.

### 6.2.4.10 Eth\_DataType

```
typedef uint8 Eth_DataType
```

Type used to pass transmit or receive data to or from the driver.

This type was defined as 8 bit wide unsigned integer because this definition is available on all CPU types.

Definition at line 387 of file Eth\_GeneralTypes.h.

### 6.2.4.11 Eth\_BufIdxType

```
typedef uint32 Eth_BufIdxType
```

Type used to identify the ethernet buffer.

This type was defined for index of buffer used in transmitted and received data.

Definition at line 394 of file Eth\_GeneralTypes.h.

### 6.2.4.12 Lin\_FrameDlType

```
typedef uint8 Lin_FrameDlType
```

Data length of a LIN Frame.

This type is used to specify the number of SDU data bytes to copy.

Definition at line 193 of file Lin\_GeneralTypes.h.

### 6.2.4.13 Lin\_FramePidType

```
typedef uint8 Lin_FramePidType
```

The LIN identifier (0..0x3F) with its parity bits.

Represents all valid protected Identifier used by Lin\_SendFrame().

Definition at line 201 of file Lin\_GeneralTypes.h.

#### 6.2.4.14 boolean

```
typedef bool boolean
```

The standard AUTOSAR type boolean shall be implemented on basis of an eight bits long unsigned integer.

Definition at line 201 of file PlatformTypes.h.

#### 6.2.4.15 uint8

```
typedef uint8_t uint8
```

Unsigned 8 bit integer with range of 0 ..+255 (0x00..0xFF) - 8 bit.

Definition at line 209 of file PlatformTypes.h.

#### 6.2.4.16 uint16

```
typedef uint16_t uint16
```

Unsigned 16 bit integer with range of 0 ..+65535 (0x0000..0xFFFF) - 16 bit.

Definition at line 216 of file PlatformTypes.h.

#### 6.2.4.17 uint32

```
typedef uint32_t uint32
```

Unsigned 32 bit integer with range of 0 ..+4294967295 (0x00000000..0xFFFFFFFF) - 32 bit.

Definition at line 223 of file PlatformTypes.h.

#### 6.2.4.18 uint64

```
typedef uint64_t uint64
```

Unsigned 64 bit integer with range of 0..18446744073709551615 (0x0000000000000000..0xFFFFFFFFFFFFFFFF)- 64 bit.

Definition at line 230 of file PlatformTypes.h.

### 6.2.4.19 sint8

```
typedef int8_t sint8
```

Signed 8 bit integer with range of -128 ..+127 (0x80..0x7F) - 7 bit + 1 sign bit.

Definition at line 238 of file PlatformTypes.h.

### 6.2.4.20 sint16

```
typedef int16_t sint16
```

Signed 16 bit integer with range of -32768 ..+32767 (0x8000..0x7FFF) - 15 bit + 1 sign bit.

Definition at line 245 of file PlatformTypes.h.

### 6.2.4.21 sint32

```
typedef int32_t sint32
```

Signed 32 bit integer with range of -2147483648.. +2147483647 (0x80000000..0x7FFFFFFF) - 31 bit + 1 sign bit.

Definition at line 252 of file PlatformTypes.h.

### 6.2.4.22 sint64

```
typedef int64_t sint64
```

Signed 64 bit integer with range of -9223372036854775808..9223372036854775807 (0x8000000000000000..0x7FFFFFFF↵FFFFFFFFFFFFFFFF) - 63 bit + 1 sign bit.

Definition at line 259 of file PlatformTypes.h.

### 6.2.4.23 uint8\_least

```
typedef uint_least8_t uint8_least
```

Unsigned integer at least 8 bit long. Range of at least 0 ..+255 (0x00..0xFF) - 8 bit.

Definition at line 266 of file PlatformTypes.h.



**6.2.4.24 uint16\_least**

```
typedef uint_least16_t uint16_least
```

Unsigned integer at least 16 bit long. Range of at least 0 ..+65535 (0x0000..0xFFFF) - 16 bit.

Definition at line 273 of file PlatformTypes.h.

**6.2.4.25 uint32\_least**

```
typedef uint_least32_t uint32_least
```

Unsigned integer at least 32 bit long. Range of at least 0 ..+4294967295 (0x00000000..0xFFFFFFFF) - 32 bit.

Definition at line 280 of file PlatformTypes.h.

**6.2.4.26 sint8\_least**

```
typedef int_least8_t sint8_least
```

Signed integer at least 8 bit long. Range - at least -128 ..+127. At least 7 bit + 1 bit sign.

Definition at line 287 of file PlatformTypes.h.

**6.2.4.27 sint16\_least**

```
typedef int_least16_t sint16_least
```

Signed integer at least 16 bit long. Range - at least -32768 ..+32767. At least 15 bit + 1 bit sign.

Definition at line 294 of file PlatformTypes.h.

**6.2.4.28 sint32\_least**

```
typedef int_least32_t sint32_least
```

Signed integer at least 32 bit long. Range - at least -2147483648.. +2147483647. At least 31 bit + 1 bit sign.

Definition at line 301 of file PlatformTypes.h.

### 6.2.4.29 float32

```
typedef float float32
```

32bit long floating point data type

Definition at line 307 of file PlatformTypes.h.

### 6.2.4.30 float64

```
typedef double float64
```

64bit long floating point data type

Definition at line 313 of file PlatformTypes.h.

### 6.2.4.31 StatusType

```
typedef uint8 StatusType
```

This type is defined for OSEK compliance.

Definition at line 166 of file StandardTypes.h.

### 6.2.4.32 Std\_ReturnType

```
typedef uint8 Std_ReturnType
```

This type can be used as standard API return type which is shared between the RTE and the BSW modules.

Definition at line 174 of file StandardTypes.h.

## 6.2.5 Enum Reference

### 6.2.5.1 Can\_ControllerStateType

```
enum Can_ControllerStateType
```

CAN Controller State Modes of operation.

States that are used by the several ControllerMode functions

Enumerator

CAN_CS_UNINIT	CAN controller state UNINIT.
CAN_CS_STARTED	CAN controller state STARTED.
CAN_CS_STOPPED	CAN controller state STOPPED.
CAN_CS_SLEEP	CAN controller state SLEEP.

Definition at line 92 of file Can\_GeneralTypes.h.

### 6.2.5.2 Can\_ErrorStateType

enum `Can_ErrorStateType`

CAN Controller State Modes of operation.

Error states of a CAN controller

Enumerator

CAN_ERRORSTATE_ACTIVE	The CAN controller takes fully part in communication.
CAN_ERRORSTATE_PASSIVE	The CAN controller takes part in communication, but does not send active error frames.
CAN_ERRORSTATE_BUSOFF	The CAN controller does not take part in communication.

Definition at line 105 of file Can\_GeneralTypes.h.

### 6.2.5.3 CanTrcv\_TrcvModeType

enum `CanTrcv_TrcvModeType`

CAN Transceiver modes.

Operating modes of the CAN Transceiver Driver.

Enumerator

CANTRCV_TRCVMODE_NORMAL	Transceiver mode NORMAL.
CANTRCV_TRCVMODE_STANDBY	Transceiver mode STANDBY.
CANTRCV_TRCVMODE_SLEEP	Transceiver mode SLEEP.

Definition at line 117 of file Can\_GeneralTypes.h.

### 6.2.5.4 CanTrcv\_\_TrcvWakeupModeType

```
enum CanTrcv__TrcvWakeupModeType
```

This type shall be used to control the CAN transceiver concerning wake up events and wake up notifications. According to [SWS\_CanTrcv\_00164] it should be present in [Can\\_\\_GeneralTypes.h](#)

Definition at line 129 of file Can\_\_GeneralTypes.h.

### 6.2.5.5 CanTrcv\_\_TrcvWakeupReasonType

```
enum CanTrcv__TrcvWakeupReasonType
```

This type denotes the wake up reason detected by the CAN transceiver in detail. According to [SWS\_CanTrcv\_00165] it should be present in [Can\\_\\_GeneralTypes.h](#)

Definition at line 142 of file Can\_\_GeneralTypes.h.

### 6.2.5.6 BufReq\_\_ReturnType

```
enum BufReq__ReturnType
```

Variables of this type are used to store the result of a buffer request.

Enumerator

BUFREQ_OK	Buffer request accomplished successful.
BUFREQ_E_NOT_OK	Buffer request not successful. Buffer cannot be accessed.
BUFREQ_E_BUSY	Temporarily no buffer available. It's up the requestor to retry request for a certain time.
BUFREQ_E_OVFL	No Buffer of the required length can be provided.

Definition at line 194 of file ComStack\_\_Types.h.

### 6.2.5.7 TpDataStateType

enum `TpDataStateType`

Variables of this type shall be used to store the state of TP buffer.

Enumerator

TP_DATACONF	Indicates that all data, that have been copied so far, are confirmed and can be removed from the TP buffer.
TP_DATARETRY	Indicates that this API call shall copy already copied data in order to recover from an error.
TP_CONFENDING	Indicates that the previously copied data must remain in the TP.

Definition at line 206 of file `ComStack_Types.h`.

### 6.2.5.8 TPParameterType

enum `TPParameterType`

Specify the parameter to which the value has to be changed (BS or STmin)

Enumerator

TP_STMIN	Separation Time.
TP_BS	Block Size.
TP_BC	Band width control parameter used in FlexRay transport protocol module

Definition at line 220 of file `ComStack_Types.h`.

### 6.2.5.9 IcomSwitch\_ErrorType

enum `IcomSwitch_ErrorType`

`IcomSwitch_ErrorType` defines the errors which can occur when activating or deactivating Pretended Networking.

Enumerator

ICOM_SWITCH_E_OK	The activation of Pretended Networking was successful.
ICOM_SWITCH_E_FAILED	The activation of Pretended Networking was not successful.

Definition at line 231 of file ComStack\_Types.h.

### 6.2.5.10 Eth\_StateType

enum [Eth\\_StateType](#)

The Ethernet driver state.

A variable of this type holds the state of the Ethernet driver module. The driver is at the ETH\_STATE\_UNINIT at the beginning until the Eth\_Init() function is called. The state remains equal to the ETH\_STATE\_INIT until the Eth\_ControllerInit() function is called. Then the state is ETH\_STATE\_ACTIVE.

Enumerator

ETH_STATE_UNINIT	The driver has not been initialized yet
ETH_STATE_INIT	The driver has not been configured and the controller was configured

Definition at line 86 of file Eth\_GeneralTypes.h.

### 6.2.5.11 Eth\_ModeType

enum [Eth\\_ModeType](#)

The Ethernet controller mode.

This type is used to store the information whether the Ethernet controller is stopped or running.

Enumerator

ETH_MODE_DOWN	Controller is shut down
ETH_MODE_ACTIVE	Controller is active

Definition at line 99 of file Eth\_GeneralTypes.h.

### 6.2.5.12 Eth\_RxStatusType

enum [Eth\\_RxStatusType](#)

The Ethernet reception status.

This status is returned by the Eth\_Receive() function to indicate whether any frame has been received and if yes, whether there is any frame still waiting in the queue (for another Eth\_Receive() call).

Enumerator

ETH_RECEIVED	A frame has been received and there are no more frames in the queue
ETH_NOT_RECEIVED	No frames received
ETH_RECEIVED_MORE_DATA_AVAILABLE	A frame received and at least another one in the queue detected

Definition at line 112 of file Eth\_GeneralTypes.h.

### 6.2.5.13 Eth\_FilterActionType

```
enum Eth_FilterActionType
```

Action type for PHY address filtering.

The Enumeration type describes the action to be taken for the MAC address given in \*PhysAddrPtr

Enumerator

ETH_ADD_TO_FILTER	Add address to the filter
ETH_REMOVE_FROM_FILTER	Remove address

Definition at line 125 of file Eth\_GeneralTypes.h.

### 6.2.5.14 Eth\_TimeStampQualType

```
enum Eth_TimeStampQualType
```

The Ethernet quality of timestamp type.

Depending on the HW, quality information regarding the evaluated time stamp might be supported. If not supported, the value shall be always Valid. For Uncertain and Invalid values, the upper layer shall discard the time stamp.

Enumerator

ETH_VALID	Success
ETH_INVALID	General failure
ETH_UNCERTAIN	Ethernet hardware access failure

Definition at line 138 of file Eth\_GeneralTypes.h.

6.2.5.15 EthTrcv\_ModeType

enum EthTrcv\_ModeType

This type defines the transceiver modes.

The Enumeration type describes the transceiver modes

Enumerator

ETHTRCV_MODE_DOWN	Transceiver disabled
ETHTRCV_MODE_ACTIVE	Transceiver enable

Definition at line 150 of file Eth\_GeneralTypes.h.

6.2.5.16 EthTrcv\_LinkStateType

enum EthTrcv\_LinkStateType

This type defines the Ethernet link state. The link state changes after an Ethernet cable gets plugged in and the transceivers on both ends negotiated the transmission parameters (i.e. baud rate and duplex mode)

Enumerator

ETHTRCV_LINK_STATE_DOWN	No physical Ethernet connection established. Physical Ethernet connection established.
-------------------------	--

Definition at line 159 of file Eth\_GeneralTypes.h.

6.2.5.17 EthTrcv\_StateType

enum EthTrcv\_StateType

This type defines the Ethernet link state. The link state changes after an Ethernet cable gets plugged in and the transceivers on both ends negotiated the transmission parameters (i.e. baud rate and duplex mode)

Enumerator

ETHTRCV_STATE_UNINIT	Driver is not yet configured. Driver is configured.
----------------------	---



Definition at line 172 of file Eth\_GeneralTypes.h.

#### 6.2.5.18 EthTrcv\_BaudRateType

enum `EthTrcv_BaudRateType`

This type defines the Ethernet baud rate. The baud rate gets either negotiated between the connected transceivers or has to be configured.

Enumerator

ETHTRCV_BAUD_RATE_10MBIT	10MBIT Ethernet connection 100MBIT Ethernet connection
ETHTRCV_BAUD_RATE_100MBIT	1000MBIT Ethernet connection
ETHTRCV_BAUD_RATE_1000MBIT	2500MBIT Ethernet connection

Definition at line 185 of file Eth\_GeneralTypes.h.

#### 6.2.5.19 EthTrcv\_DuplexModeType

enum `EthTrcv_DuplexModeType`

This type defines the Ethernet duplex mode. The duplex mode gets either negotiated between the connected transceivers or has to be configured.

Enumerator

ETHTRCV_DUPLEX_MODE_HALF	Half duplex Ethernet connection. Full duplex Ethernet connection
--------------------------	--

Definition at line 202 of file Eth\_GeneralTypes.h.

#### 6.2.5.20 EthTrcv\_WakeupModeType

enum `EthTrcv_WakeupModeType`

This type controls the transceiver wake up modes and/or clears the wake-up reason.

Enumerator

ETHTRCV_WUM_DISABLE	Transceiver wake up disabled. Transceiver wake up enabled
ETHTRCV_WUM_ENABLE	Transceiver wake up reason cleared.

Definition at line 215 of file Eth\_GeneralTypes.h.

### 6.2.5.21 EthTrcv\_WakeupReasonType

enum `EthTrcv_WakeupReasonType`

This type defines the transceiver wake up reasons.

Enumerator

ETHTRCV_WUR_NONE	No wake up reason detected. General wake up detected, no distinct reason supported by hardware.
ETHTRCV_WUR_GENERAL	Bus wake up detected. Available if supported by hardware.
ETHTRCV_WUR_BUS	Internal wake up detected. Available if supported by hardware.
ETHTRCV_WUR_INTERNAL	Reset wake up detected. Available if supported by hardware.
ETHTRCV_WUR_RESET	Power on wake up detected. Available if supported by hardware.
ETHTRCV_WUR_POWER_ON	Pin wake up detected. Available if supported by hardware.
ETHTRCV_WUR_PIN	System error wake up detected. Available if supported by hardware.

Definition at line 230 of file Eth\_GeneralTypes.h.

### 6.2.5.22 EthTrcv\_PhyTestModeType

enum `EthTrcv_PhyTestModeType`

Describes the possible PHY test modes.

Enumerator

ETHTRCV_PHYTESTMODE_NONE	normal operation test transmitter droop
ETHTRCV_PHYTESTMODE_1	test master timing jitter
ETHTRCV_PHYTESTMODE_2	test slave timing jitter
ETHTRCV_PHYTESTMODE_3	test transmitter distortion
ETHTRCV_PHYTESTMODE_4	test power spectral density (PSD) mask

Definition at line 255 of file Eth\_GeneralTypes.h.

### 6.2.5.23 EthTrcv\_PhyLoopbackModeType

enum `EthTrcv_PhyLoopbackModeType`

Describes the possible PHY loopback modes.

Enumerator

ETHTRCV_PHYLOOPBACK_NONE	normal operation internal loopback
ETHTRCV_PHYLOOPBACK_INTERNAL	external loopback
ETHTRCV_PHYLOOPBACK_EXTERNAL	remote loopback

Definition at line 276 of file Eth\_GeneralTypes.h.

#### 6.2.5.24 EthTrcv\_PhyTxModeType

enum `EthTrcv_PhyTxModeType`

Describes the possible PHY transmit modes.

Enumerator

ETHTRCV_PHYTXMODE_NORMAL	normal operation transmitter disabled
ETHTRCV_PHYTXMODE_TX_OFF	scrambler disabled

Definition at line 293 of file Eth\_GeneralTypes.h.

#### 6.2.5.25 EthTrcv\_CableDiagResultType

enum `EthTrcv_CableDiagResultType`

Describes the results of the cable diagnostics.

Enumerator

ETHTRCV_CABLEDIAG_OK	Cable diagnostic ok. Cable diagnostic failed
ETHTRCV_CABLEDIAG_ERROR	Short circuit detected.
ETHTRCV_CABLEDIAG_SHORT	Open circuit detected.
ETHTRCV_CABLEDIAG_OPEN	cable diagnostic is still running
ETHTRCV_CABLEDIAG_PENDING	cable diagnostics has detected wrong polarity of the "Ethernet physical+" or "Ethernet physical-" lines

Definition at line 308 of file Eth\_GeneralTypes.h.

### 6.2.5.26 EthSwt\_StateType

enum `EthSwt_StateType`

Status supervision used for Development Error Detection. The state shall be available for debugging.

Enumerator

ETHSWT_STATE_UNINIT	The Eth Switch Driver is not yet configured.
ETHSWT_STATE_INIT	The Eth Switch Driver is configured.
ETHSWT_STATE_ACTIVE	The Eth Switch driver is active.

Definition at line 329 of file `Eth_GeneralTypes.h`.

### 6.2.5.27 EthSwt\_MacLearningType

enum `EthSwt_MacLearningType`

MAC learning type enumeration.

Enumerator

ETHSWT_MACLEARNING_HWDISABLED	If hardware learning disabled, the switch must not learn new MAC addresses.
ETHSWT_MACLEARNING_HWENABLED	If hardware learning enabled, the switch learns new MAC addresses.
ETHSWT_MACLEARNING_SWENABLED	If software learning enabled, the hardware learning is disabled and the switch forwards packets with an unknown source address to a host CPU.

Definition at line 340 of file `Eth_GeneralTypes.h`.

### 6.2.5.28 EthSwt\_PortMirrorStateType

enum `EthSwt_PortMirrorStateType`

Type to request or obtain the port mirroring state (enable/disable) for a particular port mirror configuration per Ethernet switch.

Enumerator

PORT_MIRROR_DISABLED	port mirroring disabled.
PORT_MIRROR_ENABLED	port mirroring enabled.

Definition at line 351 of file Eth\_GeneralTypes.h.

#### 6.2.5.29 EthSwt\_MgmtOwner

enum `EthSwt_MgmtOwner`

Holds information if upper layer or EthSwt is owner of mgmt\_obj.

Enumerator

ETHSWT_MGMT_OBJ_UNUSED	Object unused.
ETHSWT_MGMT_OBJ_OWNED_BY_ETHSWT	Object used and EthSwt collects needed data.
ETHSWT_MGMT_OBJ_OWNED_BY_UPPER_LAYER	Object used and the upper layer does calculations.

Definition at line 361 of file Eth\_GeneralTypes.h.

#### 6.2.5.30 Fr\_TxLPduStatusType

enum `Fr_TxLPduStatusType`

Transmit resource status is stored to variable of this type.

Definition at line 146 of file Fr\_GeneralTypes.h.

#### 6.2.5.31 Fr\_RxLPduStatusType

enum `Fr_RxLPduStatusType`

Transmit resource status is stored to variable of this type.

Definition at line 158 of file Fr\_GeneralTypes.h.

#### 6.2.5.32 Fr\_POCTestType

enum `Fr_POCTestType`

Variables of this type are used to store the POC:state of the controller.

Definition at line 171 of file Fr\_GeneralTypes.h.

### 6.2.5.33 Fr\_SlotModeType

enum [Fr\\_SlotModeType](#)

This type is used to store the slot mode of the controller.

Definition at line 196 of file Fr\_GeneralTypes.h.

### 6.2.5.34 Fr\_ErrorModeType

enum [Fr\\_ErrorModeType](#)

Variables of this type are used for storage of FlexRay controller error mode.

Definition at line 216 of file Fr\_GeneralTypes.h.

### 6.2.5.35 Fr\_WakeupStatusType

enum [Fr\\_WakeupStatusType](#)

Variable of this type is used to query the FlexRay controller Wakeup status.

Definition at line 230 of file Fr\_GeneralTypes.h.

### 6.2.5.36 Fr\_StartupStateType

enum [Fr\\_StartupStateType](#)

Variable of this type is used to query the FlexRay controller Startup state.

Definition at line 247 of file Fr\_GeneralTypes.h.

### 6.2.5.37 Fr\_ChannelType

enum [Fr\\_ChannelType](#)

This type is used to select the channel.

Definition at line 274 of file Fr\_GeneralTypes.h.

### 6.2.5.38 Lin\_FrameCsModelType

enum [Lin\\_FrameCsModelType](#)

Checksum models for the LIN Frame.

This type is used to specify the Checksum model to be used for the LIN Frame.

Enumerator

LIN_ENHANCED_CS	Enhanced checksum model.
LIN_CLASSIC_CS	Classic checksum model.

Definition at line 83 of file Lin\_GeneralTypes.h.

### 6.2.5.39 Lin\_FrameResponseType

enum `Lin_FrameResponseType`

Frame response types.

This type is used to specify whether the frame processor is required to transmit the response part of the LIN frame.

Enumerator

LIN_FRAMERESPONSE_TX	Response is generated from this (master) node.
LIN_FRAMERESPONSE_RX	Response is generated from a remote slave node.
LIN_FRAMERESPONSE_IGNORE	Response is generated from one slave to another slave. For the master the response will be anonymous, it does not have to receive the response.

Definition at line 97 of file Lin\_GeneralTypes.h.

### 6.2.5.40 Lin\_StatusType

enum `Lin_StatusType`

LIN Frame and Channel states operation.

LIN operation states for a LIN channel or frame, as returned by the API service `Lin_GetStatus()`. part of the LIN frame.

Enumerator

LIN_NOT_OK	Development or production error occurred.
LIN_TX_OK	Successful transmission.
LIN_TX_BUSY	Ongoing transmission (Header or Response).

## Enumerator

LIN_TX_HEADER_ERROR	Erroneous header transmission such as: <ul style="list-style-type: none"> <li>• Mismatch between sent and read back data</li> <li>• Identifier parity error</li> <li>• Physical bus error.</li> </ul>
LIN_TX_ERROR	Erroneous transmission such as: <ul style="list-style-type: none"> <li>• Mismatch between sent and read back data</li> <li>• Physical bus error.</li> </ul>
LIN_RX_OK	Reception of correct response.
LIN_RX_BUSY	Ongoing reception: at least one response byte has been received, but the checksum byte has not been received.
LIN_RX_ERROR	Erroneous reception such as: <ul style="list-style-type: none"> <li>• Framing error</li> <li>• Overrun error</li> <li>• Checksum error</li> <li>• Short response.</li> </ul>
LIN_RX_NO_RESPONSE	No response byte has been received so far. This is a mess !! Frame status is mixed with channel status but i kept it here only because of LIN168.
LIN_OPERATIONAL	Normal operation;. <ul style="list-style-type: none"> <li>• The related LIN channel is ready to transmit next header</li> <li>• No data from previous frame available (e.g. after initialization).</li> </ul>
LIN_CH_SLEEP	Sleep mode operation;. <ul style="list-style-type: none"> <li>• In this mode wake-up detection from slave nodes is enabled.</li> </ul>

Definition at line 117 of file Lin\_GeneralTypes.h.

#### 6.2.5.41 Lin\_SlaveErrorType

```
enum Lin_SlaveErrorType
```

LIN Slave error type.

This type represents the slave error types that are detected during header reception and response transmission / reception



Enumerator

LIN_ERR_HEADER	Error in header.
LIN_ERR_RESP_STOPBIT	Framing error in response.
LIN_ERR_RESP_CHKSUM	Checksum error.
LIN_ERR_RESP_DATABIT	Monitoring error of transmitted data bit in response.
LIN_ERR_NO_RESP	No response.
LIN_ERR_INC_RESP	Incomplete response.

Definition at line 169 of file Lin\_GeneralTypes.h.

## 6.2.6 Variable Documentation

### 6.2.6.1 id

`Can_IdType id`

CAN L-PDU = Data Link Layer Protocol Data Unit. Consists of Identifier, DLC and Data(SDU) It is uint32 for CAN\_EXTENDEDID=STD\_ON, else is uint16.

Definition at line 177 of file Can\_GeneralTypes.h.

### 6.2.6.2 swPduHandle

`PduIdType swPduHandle`

The L-PDU Handle = defined and placed inside the CanIf module layer. Each handle represents an L-PDU, which is a constant structure with information for Tx/Rx processing.

Definition at line 181 of file Can\_GeneralTypes.h.

### 6.2.6.3 length

`uint8 length`

DLC = Data Length Code (part of L-PDU that describes the SDU length).

Definition at line 186 of file Can\_GeneralTypes.h.

### 6.2.6.4 sdu

`uint8* sdu`

CAN L-SDU = Link Layer Service Data Unit. Data that is transported inside the L-PDU.

Definition at line 188 of file `Can_GeneralTypes.h`.

### 6.2.6.5 CanId

`Can_IdType CanId`

Standard/Extended CAN ID of CAN L-PDU.

Definition at line 216 of file `Can_GeneralTypes.h`.

### 6.2.6.6 Hoh

`Can_HwHandleType Hoh`

ID of the corresponding Hardware Object Range.

Definition at line 218 of file `Can_GeneralTypes.h`.

### 6.2.6.7 ControllerId

`uint8 ControllerId`

ControllerId provided by CanIf clearly identify the corresponding controller.

Definition at line 220 of file `Can_GeneralTypes.h`.

### 6.2.6.8 nanoseconds

`uint32 nanoseconds`

Nanoseconds part of the time.

Definition at line 412 of file `Eth_GeneralTypes.h`.

#### 6.2.6.9 seconds

`uint32` seconds

32 bit LSB of the 48 bits Seconds part of the time

Definition at line 413 of file Eth\_GeneralTypes.h.

#### 6.2.6.10 secondsHi

`uint16` secondsHi

16 bit MSB of the 48 bits Seconds part of the time

Definition at line 414 of file Eth\_GeneralTypes.h.

#### 6.2.6.11 diff

`Eth_TimeStampType` diff

diff time difference

Definition at line 426 of file Eth\_GeneralTypes.h.

#### 6.2.6.12 sign

`boolean` sign

Positive (True) Or negative (False) time.

Definition at line 427 of file Eth\_GeneralTypes.h.

#### 6.2.6.13 IngressTimeStampDelta

`Eth_TimeIntDiffType` IngressTimeStampDelta

IngressTimeStampSync2 -IngressTimeStampSync1.

Definition at line 437 of file Eth\_GeneralTypes.h.

### 6.2.6.14 OriginTimeStampDelta

`Eth_TimeIntDiffType` OriginTimeStampDelta

OriginTimeStampSync2[FUP2]-OriginTimeStampSync1[FUP1].

Definition at line 438 of file Eth\_GeneralTypes.h.

### 6.2.6.15 SwitchIdx

`uint8` SwitchIdx

Switch index.

Definition at line 541 of file Eth\_GeneralTypes.h.

### 6.2.6.16 SwitchPortIdx

`uint8` SwitchPortIdx

Port index of the switch.

Definition at line 542 of file Eth\_GeneralTypes.h.

### 6.2.6.17 srcMacAddrFilter

`uint8` srcMacAddrFilter[6U]

Specifies the source MAC address [0..255,0..255,0..255,0..255,0..255,0..255] that should be mirrored. If set to 0,0,0,0,0, no source MAC address filtering shall take place.

Definition at line 550 of file Eth\_GeneralTypes.h.

### 6.2.6.18 dstMacAddrFilter

`uint8` dstMacAddrFilter[6U]

Specifies the destination MAC address [0..255,0..255,0..255,0..255,0..255,0..255] that should be mirrored. If set to 0,0,0,0,0, no destination MAC address filtering shall take place.

Definition at line 551 of file Eth\_GeneralTypes.h.

#### 6.2.6.19 VlanIdFilter

`uint16 VlanIdFilter`

Specifies the VLAN address 0..4094 that should be mirrored. If set to 65535, no VLAN filtering shall take place.

Definition at line 552 of file `Eth_GeneralTypes.h`.

#### 6.2.6.20 MirroringPacketDivider

`uint8 MirroringPacketDivider`

Divider if only a subset of received frames should be mirrored. E.g. `MirroringPacketDivider = 2` means every second frames is mirrored.

Definition at line 553 of file `Eth_GeneralTypes.h`.

#### 6.2.6.21 MirroringMode

`uint8 MirroringMode`

specifies the mode how the mirrored traffic should be tagged : 0x00 == No VLAN retagging; 0x01 == VLAN retagging; 0x03 == VLAN Double tagging

Definition at line 554 of file `Eth_GeneralTypes.h`.

#### 6.2.6.22 TrafficDirectionIngressBitMask

`uint32 TrafficDirectionIngressBitMask`

Specifies the bit mask of Ethernet switch ingress port traffic direction to be mirrored. The bit mask is calculated depending of the values of `EthSwtPortIdx`. (e.g. set `EthSwtPortIdx == 2` => `TrafficDirectionIngressBitMask = 0b0000 0000 0000 0000 0000 0000 0100`). 0b0 == enable ingress port mirroring 0b1 == disable ingress port mirroring Example: `TrafficDirectionIngressBitMask = 0b0000 0000 0000 0000 0000 0000 0100` => Ingress traffic mirroring is enabled of Ethernet switch port with `EthSwtPortIdx=2`.

Definition at line 555 of file `Eth_GeneralTypes.h`.

### 6.2.6.23 TrafficDirectionEgressBitMask

`uint32` TrafficDirectionEgressBitMask

Specifies the bit mask of Ethernet switch egress port traffic direction to be mirrored. The bit mask is calculated depending of the values of EthSwtPortIdx. (e.g. set EthSwtPortIdx == 2 => TrafficDirectionEgressBitMask = 0b0000 0000 0000 0000 0000 0000 0000 0100). 0b0 == enable egress port mirroring 0b1 == disable egress port mirroring Example: TrafficDirectionEgressBitMask = 0b0000 0000 0000 0000 0000 0000 0000 0001 => Egress traffic mirroring is enabled of Ethernet switch port with EthSwtPortIdx=0.

Definition at line 556 of file Eth\_GeneralTypes.h.

### 6.2.6.24 CapturePortIdx

`uint8` CapturePortIdx

Specifies the Ethernet switch port which capture the mirrored traffic.

Definition at line 557 of file Eth\_GeneralTypes.h.

### 6.2.6.25 ReTaggingVlanId

`uint16` ReTaggingVlanId

Specifies the VLAN address 0..4094 which shall be used for re-tagging if MirroringMode is set to 0x01 (VLAN re-tagging). If the value is set to 65535, the value shall be ignored, because the VLAN address for re-tagging is provided by the Ethernet switch configuration.

Definition at line 558 of file Eth\_GeneralTypes.h.

### 6.2.6.26 DoubleTaggingVlanId

`uint16` DoubleTaggingVlanId

Specifies the VLAN address 0..4094 which shall be used for double-tagging if MirroringMode is set to 0x02 (VLAN double tagging). If the value is set to 65535, the value shall be ignored, because the VLAN address for double tagging is provided by the Ethernet switch configuration.

Definition at line 559 of file Eth\_GeneralTypes.h.

#### 6.2.6.27 IngressTimestampValid

`Std_ReturnType` IngressTimestampValid

IngressTimestampValid shall be set to E\_NOT\_OK if ingress timestamp is not available.

Definition at line 568 of file Eth\_GeneralTypes.h.

#### 6.2.6.28 EgressTimestampValid

`Std_ReturnType` EgressTimestampValid

EgressTimestampValid shall be set to E\_NOT\_OK if ingress timestamp is not available.

Definition at line 569 of file Eth\_GeneralTypes.h.

#### 6.2.6.29 MgmtInfoValid

`Std_ReturnType` MgmtInfoValid

MgmtInfoValid shall be set to E\_NOT\_OK if ingress timestamp is not available(e.g. timeout).

Definition at line 570 of file Eth\_GeneralTypes.h.

#### 6.2.6.30 Validation

`EthSwt_MgmtObjectValidType` Validation

The validation information for the mgmt\_obj.

Definition at line 578 of file Eth\_GeneralTypes.h.

#### 6.2.6.31 IngressTimestamp

`Eth_TimeStampType` IngressTimestamp

The ingress timestamp value out of the switch.

Definition at line 579 of file Eth\_GeneralTypes.h.

### 6.2.6.32 EgressTimestamp

`Eth_TimeStampType` EgressTimestamp

The egress timestamp value out of the switch.

Definition at line 580 of file `Eth_GeneralTypes.h`.

### 6.2.6.33 MgmtInfo

`EthSwt_MgmtInfoType` MgmtInfo

Received/Transmitted Management information of the switches.

Definition at line 581 of file `Eth_GeneralTypes.h`.

### 6.2.6.34 Ownership

`EthSwt_MgmtOwner` Ownership

The ownership of MgmtObj.

Definition at line 582 of file `Eth_GeneralTypes.h`.

### 6.2.6.35 Pid

`Lin_FramePidType` Pid

LIN frame identifier.

Definition at line 212 of file `Lin_GeneralTypes.h`.

### 6.2.6.36 Cs

`Lin_FrameCsModelType` Cs

Checksum model type.

Definition at line 213 of file `Lin_GeneralTypes.h`.



#### 6.2.6.37 Drc

`Lin_FrameResponseType` Drc

Response type.

Definition at line 214 of file `Lin_GeneralTypes.h`.

#### 6.2.6.38 Dl

`Lin_FrameDlType` Dl

Data length.

Definition at line 215 of file `Lin_GeneralTypes.h`.

#### 6.2.6.39 SduPtr

`uint8*` SduPtr

Pointer to Sdu.

Definition at line 216 of file `Lin_GeneralTypes.h`.

**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).

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. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. 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.

© 2022 NXP B.V.

