User Manual

for S32K1_S32M24X BASENXP Driver

Document Number: UM2BASENXPASRR21-11 Rev0000R2.0.0 Rev. 1.0

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	04.08.2023	NXP RTD Team	S32K1_S32M24X Real-Time Drivers AUTOSAR 4.4 & R21-11 Version 2.0.0

Chapter 2

Introduction

This User Manual describes NXP Semiconductor AUTOSAR Base for S32K1_S32M24X. 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 / MWCT1014S_lqfp64

- s32k144_lqfp100 / MWCT1014S_lqfp100
- s32k144_mapbga100
- s32k144w_lqfp48
- s32k144w_lqfp64
- s32k146_lqfp64
- s32k146_lqfp100 / MWCT1015S_lqfp100
- s32k146_mapbga100 / MWCT1015S_mapbga100
- s32k146_lqfp144
- s32k148_lqfp100
- s32k148_mapbga100 / MWCT1016S_mapbga100
- s32k148_lqfp144
- s32k148_lqfp176
- s32m241_lqfp64
- s32m242_lqfp64
- s32m243_lqfp64
- s32m244_lqfp64

All of the above microcontroller devices are collectively named as S32K1_S32M24X. Note: MWCT part numbers contain NXP confidential IP for Qi Wireless Power

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 Signifigant 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 R21-11
2	Specification of Communication Stack Types	AUTOSAR Release R21-11
3	Specification of Compiler Abstraction	AUTOSAR Release R21-11
4	Specification of Platform Types	AUTOSAR Release R21-11
5	Specification of Standard Types	AUTOSAR Release R21-11
6	S32K1 Series Reference Manual	Rev. 14, 09/2021
7	S32M24x Reference Manual	Rev. 2 Draft A, 05/2023
8	S32K116 Mask Set Errata for Mask	S32K116_0N96V Rev. 22/OCT/2021
9	S32K118 Mask Set Errata for Mask	S32K118_0N97V Rev. 22/OCT/2021
10	S32K142 Mask Set Errata for Mask	S32K142_0N33V Rev. 22/OCT/2021
11	S32K144 Mask Set Errata for Mask	S32K144_0N57U Rev. 22/OCT/2021
12	S32K144W Mask Set Errata for Mask	S32K144W_0P64A Rev. 22/OCT/2021
13	S32K146 Mask Set Errata for Mask	S32K146_0N73V Rev. 22/OCT/2021
14	S32K148 Mask Set Errata for Mask	S32K148_0N20V Rev. 22/OCT/2021

#	Title	Version
15	S32M244 Mask Set Errata for Mask	S32M244_P64A+P73G, Rev. 0
16	S32M242 Mask Set Errata for Mask	S32M242_N33V+P73G, Rev. 0, 6/2023
17	S32K1xx Data Sheet	Rev. 14, 08/2021
18	S32M2xx Data Sheet	Rev. 3 Draft A, 05/2023

Chapter 3

Driver

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↩ _S32M24X RTD release. They must be re-written by the integrator.
- Files that are required by the NXP Semiconductor S32K1_S32M24X 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
Can_GeneralTypes.h	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_S32M24X 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 Can_GeneralTypes.h - ComStack_Cfg.h and Can_Cfg.h.
Compiler.h	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_S32↩ M24X RTD MCAL release provides this file and can be used as-is.
Compiler_Cfg.h	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_S32M24X 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.
ComStack_Cfg.h	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_S32M24X RTD release, it contains only the defines/typedefs that are needed by the RTD MCAL drivers.
ComStackTypes.h	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_S32M24X RTD release, it contains only the defines/typedefs that are needed by the RTD MCAL drivers.
ComStack_Types.h	RTD MCAL specific file - to be used as-is. Can be replaced by integrators to ensure compatibility in stacks where the Com↩ Stack header file name was not aligned to ComStackTypes.h.	This is a file that is specific to NXP Semiconductor S32K1_S32M24X RTD MCAL release. It is a wrapper of ComStackTypes.h to ensure compatibility of Autosar header includes.

File Name	File Type	Description
Eth_GeneralTypes.h	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_S32M24X RTD release, it contains only the defines/typedefs/constants that are needed by the RTD MCAL drivers.
Fr_GeneralTypes.h	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_S32M24X RTD release, it contains only the defines/typedefs/constants that are needed by the RTD MCAL drivers.
Lin_GeneralTypes.h	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_S32M24X RTD release, it contains only the defines/typedefs/constants that are needed by the RTD MCAL drivers.
Mcal.h	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).
PlatformTypes.h	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_S32M24X RTD MCAL release provides this file and can be used as-is.
Platform_Types.h	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 PlatformTypes.h.	This is a file that is specific to NXP Semiconductor S32K1_S32M24X RTD MCAL release. It is a wrapper of PlatformTypes.h to ensure compatibility of Autosar header includes.
RegLockMacros.h	RTD MCAL specific file - to be used as-is.	This is a file that is specific to S32K1_S32M24X RTD MCAL release. It contains defines needed by RTD MCAL drivers.

File Name	File Type	Description
Reg_eSys.h	RTD MCAL specific file - to be used as-is.	This is a file that is specific to S32K1_↩ S32M24X 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_↩ S32M24X RTD MCAL release. It contains defines needed by RTD MCAL drivers.
StandardTypes.h	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_S32↩ M24X RTD MCAL release provides this file and can be used as-is.
Std_Types.h	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 StandardTypes.h.	This is a file that is specific to NXP Semiconductor S32K1_S32M24X RTD MCAL release. It is a wrapper of StandardTypes.h 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. 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 interdepencies 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 - Precompile 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 - Precompile 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	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[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 -	MemMap section is a stub, this requirement is not implemented.
ECUC_MemMap_00002	N/S	Container Name - MemMap↩ AddressingModeSet - Description - Defines a set of addressing modes which might apply to a SwAddrMethod. - Configuration Parameters -	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 - Precompile 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	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 -	MemMap section is a stub, this requirement is not implemented.

Requirement	Status	Description	Notes
ECUC_MemMap_00017	N/S	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 atttribute 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 - – - - Post-build time - – - - Scope / Dependency - scope: ECU	MemMap section is a stub, this requirement is not implemented.

Requirement	Status	Description	Notes

Requirement	Status	Description	Notes
ECUC_MemMap_00008	N/S	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 following 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 defined similar to the representation of enumeration types in the XML schema to ensure backward compatibility. - Multiplicity - 0..* - Type - EcucStringParamDef - Default value - – - maxLength - – - minLength - – - regularExpression - – - Post-Build Variant Multiplicity -	MemMap section is a stub, this requirement is not implemented.

Requirement	Status	Description	Notes

Requirement	Status	Description	Notes
ECUC_MemMap_00007	N/S	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 - Post-Build Variant Value - false - Multiplicity Configuration Class -	MemMap section is a stub, this requirement is not implemented.

Requirement	Status	Description	Notes
ECUC_MemMap_00010	N/S	Container Name - MemMap↩ Allocation - 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. - Configuration Parameters -	MemMap section is a stub, this requirement is not implemented.
ECUC_MemMap_00019	N/S	Container Name - MemMap↩ GenericCompilerMemClass - 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. The configures the Compiler Abstraction. - Configuration Parameters -	MemMap section is a stub, this requirement is not implemented.
ECUC_MemMap_00020	N/S	Name - MemMapGeneric↩ CompilerMemClassSymbolImpl - Parent Container - MemMap↩ GenericCompilerMemClass - Description - Defines the implementation behind the generic MemClassSymbol and configures the Compiler Abstraction. - Multiplicity - 1 - Type - EcucStringParamDef - Default value - – - maxLength - – - minLength - – - regularExpression - – - Post-Build Variant Value - false - Value Configuration Class - Precompile 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_00011	N/S	Container Name - MemMap↩ GenericMapping - Description - Defines which Sw↩ AddrMethod is implemented with which MemMapAddressingMode↩ Set. 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.↩ 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 -	MemMap section is a stub, this requirement is not implemented.
ECUC_MemMap_00012	N/S	Name - MemMapAddressing↩ ModeSetRef - Parent Container - MemMap↩ GenericMapping - Description - Reference to the MemMapAddressingModeSet which applies to the MemMap↩ GenericMapping. - Multiplicity - 1 - Type - Reference to [MemMap↩ AddressingModeSet] - Post-Build Variant Value - false - Value Configuration Class - Precompile 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_00013	N/S	Name - MemMapSwAddress↩ MethodRef - Parent Container - MemMap↩ GenericMapping - 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 - – - - Scope / Dependency - scope: ECU -	MemMap section is a stub, this requirement is not implemented.
ECUC_MemMap_00014	N/S	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 -	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 requirement 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 requirement is not implemented.
SWS_Std_00999	N/S	These requirements are not applicable to this specification.	These requirements are not applicable 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 Memory↩ Sections.MemorySection. prefix needs to reference the SectionNamePrefix.	Will be analyzed after Autosar clarifications (ARTD-8691).
ECUC_MemMap_00023	N/S	Name - MemMapMapping↩ SelectorRef - Parent Container - MemMap↩ GenericMapping - Description - Reference to a MemMapPrefixSelector. The owning MemMapGenericMapping is only effective for those memories where the MemMapMapping↩ Selector matches. - Multiplicity - 0..1 - Type - Reference to [MemMap↩ MappingSelector] - 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 - MemMap↩ MappingSelector - Description - The container holds a section criteria reusable for Mem↩ MapGenericMappings. - Configuration Parameters -	Does not impact to Base driver.

Requirement	Status	Description	Notes
ECUC_MemMap_00022	N/S	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 - – - Post-Build Variant Value - false - Value Configuration Class - Precompile time - X - All Variants - Link time - – - - Post-build time - – - - Scope / Dependency - scope: ECU -	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 BaseNXP
  - 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 ModuleId
    - Parameter VendorId
    - Parameter VendorApiInfix
    - Parameter ArReleaseMajorVersion
    - Parameter ArReleaseMinorVersion
    - Parameter ArReleaseRevisionVersion
    - Parameter SwMajorVersion
    - Parameter SwMinorVersion
    - Parameter SwPatchVersion

4.1 Module BaseNXP

Configuration of BaseNXP 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

Property	Value
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

Property	Value
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. When this feature is enabled, the application will have to provide the functions expected by the OSIF API.

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
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 driver to zero or multiple ECUC partitions to make the module's API available in this partition.

The OsIf driver 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

Property	Value
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.13 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/EcucDefs/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 OsIfCounterEcucPartitionRef.

Property	Value
type	ECUC-REFERENCE-DEF

Property	Value
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/EcucDefs/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 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

Property	Value
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	0
min	0

4.17 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

4.18 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][API name from SWS].

E.g. Assuming that the VendorId of the implementor is 123 and the implementer chose a

VendorApiInfix of 'v11r456' an API named 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	

4.19 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

4.20 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

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	7
max	7
min	7

4.21 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

4.22 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

Property	Value
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2
max	2
min	2

4.23 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
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	0
min	0

4.24 Parameter SwPatchVersion

Patch 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	0
max	0
min	0

Chapter 5

Module Index

5.1 Software Specification

Here is a list of all modules:

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.

Data Structures

- struct OsIf_ConfigType
  OsIF configuration type. *More...*

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.
- void OsIf_Timer_Custom_Init (void)
  Initialize the custom timer.

- uint32 OsIf_Timer_Custom_GetCounter (void)
  Get counter value from custom timer.
- uint32 OsIf_Timer_Custom_GetElapsed (uint32 *const CurrentRef)
  Get elapsed value from custom timer.
- void OsIf_Timer_Custom_SetTimerFrequency (uint32 Freq)
  Set custom timer frequency.
- uint32 OsIf_Timer_Custom_MicrosToTicks (uint32 Micros)
  Convert micro second to ticks based on custom timer frequency.
- void OsIf_Timer_System_Init (void)
  Initialize the system timer.
- uint32 OsIf_Timer_System_GetCounter (void)
  Get counter value from system timer.
- uint32 OsIf_Timer_System_GetElapsed (uint32 *const CurrentRef)
  Get elapsed value from system timer.
- void OsIf_Timer_System_SetTimerFrequency (uint32 Freq)
  Set system timer frequency.
- uint32 OsIf_Timer_System_MicrosToTicks (uint32 Micros)
  Convert micro second to ticks based on system timer frequency.

6.1.2 Data Structure Documentation

6.1.2.1 struct OsIf_ConfigType

OsIF configuration type.

Definition at line 149 of file OsIf_Cfg_TypesDef.h.

6.1.3 Enum Reference

6.1.3.1 OsIf_CounterType

```
enum OsIf_CounterType
```

OsIf Counter type.

Counter type.

@detail The dummy counter of Osif is meant as a loop-counter timeout mechanism that requirement no additional resource (hardware and software). It was meant to replace the typical loop timeout of decrementing a variable each time the loop was executed until the counter reaches zero. Usage of OsIf replaced these loop counters within RTD, so the advantage is that these timeouts can be configured to be simple loop counters (using the dummy counter), not changing the RTD code.

Enumerator

OSIF_COUNTER_DUMMY	dummy counter

Definition at line 119 of file OsIf.h.

6.1.4 Function Reference

6.1.4.1 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 intialization other than Mcu.

6.1.4.2 OsIf_GetCounter()

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

Get the current value counter.

This function returns the current value of the counter.

Parameters

in	*SelectedCounter*	the type of counter to use

Returns

the current value of the counter

6.1.4.3 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	*CurrentRef*	reference counter value, updated to current counter value
in	*SelectedCounter*	the type of counter to use

Returns

the elapsed time

6.1.4.4 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	*Freq*	the new frequency
in	*SelectedCounter*	the type of counter to use

6.1.4.5 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	*Micros*	microseconds value
in	*SelectedCounter*	the type of counter to use

Returns

the equivalent value in ticks

6.1.4.6 OsIf_Timer_Custom_Init()

```
void OsIf_Timer_Custom_Init (
            void  )
```

Initialize the custom timer.

This function initialize the custom timer.

6.1.4.7 OsIf_Timer_Custom_GetCounter()

```
uint32 OsIf_Timer_Custom_GetCounter (
            void  )
```

Get counter value from custom timer.

This function get counter value from custom timer.

Returns

Counter value

6.1.4.8 OsIf_Timer_Custom_GetElapsed()

```
uint32 OsIf_Timer_Custom_GetElapsed (
            uint32 *const CurrentRef )
```

Get elapsed value from custom timer.

This function get elapsed value from custom timer.

Parameters

in	*CurrentRef*	The pointer to current reference point

Returns

Elapsed value

6.1.4.9 OsIf_Timer_Custom_SetTimerFrequency()

```
void OsIf_Timer_Custom_SetTimerFrequency (
            uint32 Freq )
```

Set custom timer frequency.

This function set custom timer frequency.

Parameters

in	*Freq*	Frequency value

6.1.4.10 OsIf_Timer_Custom_MicrosToTicks()

```
uint32 OsIf_Timer_Custom_MicrosToTicks (
            uint32 Micros )
```

Convert micro second to ticks based on custom timer frequency.

This function Convert micro second to ticks based on custom timer frequency.

Parameters

in	*Micros*	Micro second

Returns

Ticks

6.1.4.11 OsIf_Timer_System_Init()

```
void OsIf_Timer_System_Init (
            void  )
```

Initialize the system timer.

This function initialize the system timer.

6.1.4.12 OsIf_Timer_System_GetCounter()

```
uint32 OsIf_Timer_System_GetCounter (
            void  )
```

Get counter value from system timer.

This function get counter value from system timer.

Returns

Counter value

6.1.4.13 OsIf_Timer_System_GetElapsed()

```
uint32 OsIf_Timer_System_GetElapsed (
            uint32 *const CurrentRef )
```

Get elapsed value from system timer.

This function get elapsed value from system timer.

Parameters

in	*CurrentRef*	The pointer to current reference point

Returns

Elapsed value

6.1.4.14 OsIf_Timer_System_SetTimerFrequency()

```
void OsIf_Timer_System_SetTimerFrequency (
            uint32 Freq )
```

Set system timer frequency.

This function set system timer frequency.

Parameters

in	*Freq*	Frequency value

6.1.4.15 OsIf_Timer_System_MicrosToTicks()

```
uint32 OsIf_Timer_System_MicrosToTicks (
            uint32 Micros )
```

Convert micro second to ticks based on system timer frequency.

This function Convert micro second to ticks based on system timer frequency.

Parameters

in	*Micros*	Micro second

Returns

Ticks

6.2 BASENXP_COMPONENT

6.2.1 Detailed Description

Data Structures

- struct Can_PduType

  Can_PduType. *More...*
- struct Can_HwType

  Can_HwType. *More...*
- struct Can_TimeStampType

  Can_TimeStampType. *More...*
- struct CanXL_Params

  Contains CAN XL specific information. *More...*
- struct CanXL_PduType

  This type extends the classical Can_PduType with a larger PDU length, the CanXL_Params and a sec to indicate simple or extended content. *More...*
- struct CanXL_HwType

  This type defines a data structure which provides a CAN XL Hardware Object Handle including its corresponding CAN Controller and therefore CanDrv as well as the specific CAN XL parameters. *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_POCStatusType

  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...
- struct Std_TransformerError

  Std_TransformerError represents a transformer error in the context of a certain transformer chain. More...
- struct Std_TransformerForward

  Std_TransformerError represents a transformer error in the context of a certain transformer chain. 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 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 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_CANCELATION_OK
  Requested cancellation has been executed.
- #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.
- #define NTFRSLT_PARAMETER_OK
  The parameter change request has been successfully executed.
- #define NTFRSLT_E_PARAMETER_NOT_OK
  The request for the change of the parameter did not complete successfully.
- #define NTFRSLT_E_RX_ON
  The parameter change request not executed successfully due to an ongoing reception.
- #define NTFRSLT_E_VALUE_NOT_OK
  The parameter change request not executed successfully due to a wrong 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 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 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 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 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 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 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_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 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 E_MEM_SERVICE_NOT_AVAIL

  According to SWS_Mem_00070: In case a service is not relevant for a specific memory device technology, the service shall always return E_MEM_SERVICE_NOT_AVAIL. The return code is implementation-defined (see SWS_↩ Std_00011 and SWS_Std_00005).

- #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 uint16 CbkHandleIdType

  Used for the handle Ids of Com and LdCom user callbacks.
- 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..0x7FFF↩ FFFFFFFFFFFF)- 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.
- typedef uint8 Std_TransformerErrorCode

  The type of the Std_TransformerError.
- typedef Std_ReturnType(∗ Std_ExtractProtocolHeaderFieldsType) (const uint8 ∗buffer, uint32 bufferLength, Std_MessageTypeType ∗messageType, Std_MessageResultType ∗messageResult)

  Type for the function pointer to extract the relevant protocol header fields of the message and the type of the message result of a transformer. - At the time being, this is limited to the types used for C/S communication (i.e., REQUEST and RESPONSE and OK and ERROR)

Enum Reference

- enum Can_ControllerStateType

  CAN Controller State Modes of operation.
- enum Can_ErrorStateType

  CAN Controller State Modes of operation.
- enum CanTrcv_TrcvModeType

  CAN Transceiver modes.
- enum CanTrcv_TrcvWakeupModeType
- enum CanTrcv_TrcvWakeupReasonType
- enum Can_ErrorType

  CAN Controller Error Types of operation.

- 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.
- enum Std_TransformerClass

  The Std_TransformerClass represents the transformer class in which the error occurred.
- enum Std_TransformerForwardCode

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

  This type is used to encode the different type of messages. - Currently this encoding is limited to the distinction between requests and responses in C/S communication.
- enum Std_MessageResultType

  This type is used to encode different types of results for response messages. - Currently this encoding is limited to the distinction between OK and ERROR responses.

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_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.
- 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

  Seconds part of the time.
- uint16 PriorityId

  Priority ID of a CAN XL message.
- uint16 Vcid

  VCID of a CAN XL message.
- uint8 SduType

  SDU type of a CAN XL message.

- uint32 AcceptanceField

  Acceptance field of a CAN XL message.
- uint8 Sec

  Simple extended content field of a CAN XL message.
- PduIdType swPduHandle

  Contains the PDU ID.
- uint16 length

  Length of the data.
- uint8 * sdu

  SDU data pointer.
- CanXL_Params * XLParams

  Pointer to CAN XL params.
- CanXL_Params * XLParams

  Pointer to CAN XL params.
- uint8 ControllerId

  ControllerId provided by CanIf, identifies the corresponding CAN XL controller.
- Can_HwHandleType Hoh

  ID of the corresponding CAN XL Hardware Object Range.
- 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 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.
- 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 AE_MEMMAP_VENDOR_ID

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

- #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 BASENXP_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_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 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_MEMMAP_VENDOR_ID

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

- #define CANIF_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_AE_MEMMAP_VENDOR_ID

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

- #define COMPILER_VENDOR_ID

  Parameters that shall be published within the compiler abstraction 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 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 COMPILERDEFINITION_VENDOR_ID

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

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

- #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 CXPI_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 DPGA_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_ENET_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 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 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_PHY_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.

- #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 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 FLS_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.

- 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_POCStateType
- 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 FR_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 GDU_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.

- #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 ISELED_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 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.

- 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.
- enum LinTrcv_TrcvWakeupModeType

  Transceiver Wake Up Mode Types.
- enum LinTrcv_TrcvWakeupReasonType

  Transceiver Wake Up Reason Types.
- enum LinTrcv_TrcvModeType

  Operating modes of the LIN Transceiver Driver.
- 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_LPUART_FLEXIO_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 LINTRCV_43_AE_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 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 MEM_43_EEP_MEMMAP_VENDOR_ID

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

- #define MEM_43_EXFLS_MEMMAP_VENDOR_ID

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

- #define MEM_43_INFLS_MEMMAP_VENDOR_ID

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

- #define OCOTP_MEMMAP_VENDOR_ID

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

- #define MEMACC_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.

- #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 PMIC_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 PWM_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 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 UART_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 195 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 234 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 Can_TimeStampType

Can_TimeStampType.

Variables of this type are used to express time stamps based on relative time.

Definition at line 249 of file Can_GeneralTypes.h.

Data Fields

Type	Name	Description
uint32	nanoseconds	Nanoseconds part of the time.
uint32	seconds	Seconds part of the time.

6.2.2.4 struct CanXL_Params

Contains CAN XL specific information.

Definition at line 257 of file Can_GeneralTypes.h.

Data Fields

Type	Name	Description
uint16	PriorityId	Priority ID of a CAN XL message.
uint16	Vcid	VCID of a CAN XL message.
uint8	SduType	SDU type of a CAN XL message.
uint32	AcceptanceField	Acceptance field of a CAN XL message.
uint8	Sec	Simple extended content field of a CAN XL message.

6.2.2.5 struct CanXL_PduType

This type extends the classical Can_PduType with a larger PDU length, the CanXL_Params and a sec to indicate simple or extended content.

Definition at line 271 of file Can_GeneralTypes.h.

Data Fields

Type	Name	Description
PduIdType	swPduHandle	Contains the PDU ID.
uint16	length	Length of the data.
uint8 *	sdu	SDU data pointer.
CanXL_Params *	XLParams	Pointer to CAN XL params.

6.2.2.6 struct CanXL_HwType

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

Definition at line 283 of file Can_GeneralTypes.h.

Data Fields

Type	Name	Description
CanXL_Params *	XLParams	Pointer to CAN XL params.
uint8	ControllerId	ControllerId provided by CanIf, identifies the corresponding CAN XL controller.
Can_HwHandleType	Hoh	ID of the corresponding CAN XL Hardware Object Range.

6.2.2.7 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 265 of file ComStack_Types.h.

Data Fields

- uint8 * SduDataPtr
- uint8 * MetaDataPtr
- PduLengthType SduLength

6.2.2.7.1 Field Documentation

6.2.2.7.1.1 SduDataPtr uint8* SduDataPtr

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

Definition at line 267 of file ComStack_Types.h.

6.2.2.7.1.2 MetaDataPtr uint8* MetaDataPtr

pointer to the meta data of the PDU

Definition at line 268 of file ComStack_Types.h.

6.2.2.7.1.3 SduLength PduLengthType SduLength

length of the SDU in bytes

Definition at line 269 of file ComStack_Types.h.

6.2.2.8 struct RetryInfoType

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

Definition at line 276 of file ComStack_Types.h.

Data Fields

- TpDataStateType TpDataState
- PduLengthType TxTpDataCnt

6.2.2.8.1 Field Documentation

6.2.2.8.1.1 TpDataState TpDataStateType TpDataState

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

Definition at line 278 of file ComStack_Types.h.

6.2.2.8.1.2 TxTpDataCnt PduLengthType TxTpDataCnt

Offset from the current position which identifies the number of bytes to be retransmitted.

Definition at line 279 of file ComStack_Types.h.

6.2.2.9 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 [0x3FFF FFFF] Bit 30 and 31 reserved, default: 0

Definition at line 411 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

6.2.2.10 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 425 of file Eth_GeneralTypes.h.

Data Fields

- Eth_TimeStampType diff
  diff time difference
- boolean sign
  Positive (True) Or negative (False) time.

6.2.2.11 struct Eth_RateRatioType

Type used to express frequency ratio.

Variables of this type are used to express frequency ratios.

Definition at line 436 of file Eth_GeneralTypes.h.

Data Fields

- Eth_TimeIntDiffType IngressTimeStampDelta
  IngressTimeStampSync2 -IngressTimeStampSync1.
- Eth_TimeIntDiffType OriginTimeStampDelta
  OriginTimeStampSync2[FUP2]-OriginTimeStampSync1[FUP1].

6.2.2.12 struct Eth_CounterType

Type used to statistic counter for diagnostics.

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

Definition at line 447 of file Eth_GeneralTypes.h.

6.2.2.13 struct Eth_RxStatsType

Type used to statistic counter for diagnostics.

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

Definition at line 473 of file Eth_GeneralTypes.h.

6.2.2.14 struct Eth_TxStatsType

Type used to statistic counter for diagnostics.

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

Definition at line 500 of file Eth_GeneralTypes.h.

6.2.2.15 struct Eth_TxErrorCounterValuesType

Type used to statistic counter for diagnostics.

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

Definition at line 512 of file Eth_GeneralTypes.h.

6.2.2.16 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 528 of file Eth_GeneralTypes.h.

6.2.2.17 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 540 of file Eth_GeneralTypes.h.

Data Fields

Type	Name	Description
uint8	SwitchIdx	Switch index.
uint8	SwitchPortIdx	Port index of the switch.

6.2.2.18 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 549 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.

Data Fields

Type	Name	Description
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.

6.2.2.19 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 567 of file Eth_GeneralTypes.h.

Data Fields

Type	Name	Description
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).

6.2.2.20 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 577 of file Eth_GeneralTypes.h.

Data Fields

Type	Name	Description
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.

6.2.2.21 struct Fr_POCStatusType

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

Definition at line 288 of file Fr_GeneralTypes.h.

6.2.2.22 struct Fr_SlotAssignmentType

Variables of this type are used to store information of frame indentified by Fr_LPduIdx.

Definition at line 308 of file Fr_GeneralTypes.h.

6.2.2.23 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 258 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.

6.2.2.24 struct Mcal_DemErrorType

Typedef for DEM error management implemented by MCAL drivers.

Definition at line 316 of file Mcal.h.

Data Fields

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

6.2.2.25 struct Std_VersionInfoType

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

Definition at line 188 of file StandardTypes.h.

Data Fields

Type	Name	Description
uint16	vendorID	vendor ID
uint16	moduleID	BSW module ID.
uint8	sw_major_version	BSW module software major version.
uint8	sw_minor_version	BSW module software minor version.
uint8	sw_patch_version	BSW module software patch version.

6.2.2.26 struct Std_TransformerError

Std_TransformerError represents a transformer error in the context of a certain transformer chain.

Definition at line 220 of file StandardTypes.h.

Data Fields

- Std_TransformerErrorCode errorCode

  The specific meaning of the values of Std_TransformerErrorCode is to be seen for the specific transformer chain for which the data type represents the transformer error.
- Std_TransformerClass transformerClass

  The transformerClass.

6.2.2.26.1 Field Documentation

6.2.2.26.1.1 errorCode `Std_TransformerErrorCode errorCode`

The specific meaning of the values of Std_TransformerErrorCode is to be seen for the specific transformer chain for which the data type represents the transformer error.

Definition at line 222 of file StandardTypes.h.

6.2.2.26.1.2 transformerClass `Std_TransformerClass transformerClass`

The transformerClass.

Definition at line 224 of file StandardTypes.h.

6.2.2.27 struct Std_TransformerForward

Std_TransformerError represents a transformer error in the context of a certain transformer chain.

Definition at line 244 of file StandardTypes.h.

Data Fields

- Std_TransformerForwardCode errorCode

  The specific meaning of the values of Std_TransformerErrorCode is to be seen for the specific transformer chain for which the data type represents the transformer error.
- Std_TransformerClass transformerClass

  The transformerClass.

6.2.2.27.1 Field Documentation

6.2.2.27.1.1 errorCode `Std_TransformerForwardCode errorCode`

The specific meaning of the values of Std_TransformerErrorCode is to be seen for the specific transformer chain for which the data type represents the transformer error.

Definition at line 246 of file StandardTypes.h.

6.2.2.27.1.2 transformerClass `Std_TransformerClass transformerClass`

The transformerClass.

Definition at line 248 of file StandardTypes.h.

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 57 of file Adc_MemMap.h.

6.2.3.2 MEMMAP_ERROR [1/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Adc_MemMap.h.

6.2.3.3 AE_MEMMAP_VENDOR_ID

```
#define AE_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 57 of file Ae_MemMap.h.

6.2.3.4 MEMMAP_ERROR [2/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Ae_MemMap.h.

6.2.3.5 BASENXP_MEMMAP_VENDOR_ID

```
#define BASENXP_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 57 of file BaseNXP_MemMap.h.

6.2.3.6 MEMMAP_ERROR [3/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file BaseNXP_MemMap.h.

6.2.3.7 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 57 of file Can_43_FLEXCAN_MemMap.h.

6.2.3.8 MEMMAP_ERROR [4/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Can_43_FLEXCAN_MemMap.h.

6.2.3.9 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 57 of file Can_43_LLCE_MemMap.h.

6.2.3.10 MEMMAP_ERROR [5/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Can_43_LLCE_MemMap.h.

6.2.3.11 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 60 of file Can_GeneralTypes.h.

6.2.3.12 CAN_BUSY

```
#define CAN_BUSY
```

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

Definition at line 81 of file Can_GeneralTypes.h.

6.2.3.13 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 57 of file Can_MemMap.h.

6.2.3.14 MEMMAP_ERROR [6/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Can_MemMap.h.

6.2.3.15 CANIF_MEMMAP_VENDOR_ID

```
#define CANIF_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 57 of file CanIf_MemMap.h.

6.2.3.16 MEMMAP_ERROR [7/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file CanIf_MemMap.h.

6.2.3.17 CANTRCV_43_AE_MEMMAP_VENDOR_ID

```
#define CANTRCV_43_AE_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 57 of file CanTrcv_43_AE_MemMap.h.

6.2.3.18 MEMMAP_ERROR [8/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file CanTrcv_43_AE_MemMap.h.

6.2.3.19 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 57 of file CanTrcv_43_tja1145a_MemMap.h.

6.2.3.20 MEMMAP_ERROR [9/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file CanTrcv_43_tja1145a_MemMap.h.

6.2.3.21 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 64 of file Compiler.h.

6.2.3.22 AUTOMATIC

```
#define AUTOMATIC
```

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

Definition at line 88 of file Compiler.h.

6.2.3.23 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 98 of file Compiler.h.

6.2.3.24 NULL_PTR

```
#define NULL_PTR
```

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

Definition at line 104 of file Compiler.h.

6.2.3.25 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 437 of file Compiler.h.

6.2.3.26 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 443 of file Compiler.h.

6.2.3.27 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 449 of file Compiler.h.

6.2.3.28 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 455 of file Compiler.h.

6.2.3.29 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 461 of file Compiler.h.

6.2.3.30 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 467 of file Compiler.h.

6.2.3.31 CONST

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

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

Definition at line 473 of file Compiler.h.

6.2.3.32 VAR

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

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

Definition at line 479 of file Compiler.h.

6.2.3.33 CONSTP2FUNC

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

The compiler abstraction for const pointer to function.

Definition at line 485 of file Compiler.h.

6.2.3.34 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 492 of file Compiler.h.

6.2.3.35 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 498 of file Compiler.h.

6.2.3.36 ADC_CODE

```
#define ADC_CODE
```

ADC memory and pointer classes.

Definition at line 66 of file Compiler_Cfg.h.

6.2.3.37 CAN_CODE

```
#define CAN_CODE
```

CAN memory and pointer classes.

Definition at line 82 of file Compiler_Cfg.h.

6.2.3.38 CAN_43_LLCE_CODE

```
#define CAN_43_LLCE_CODE
```

CAN_43_LLCE memory and pointer classes.

Definition at line 98 of file Compiler_Cfg.h.

6.2.3.39 CANIF_CODE

```
#define CANIF_CODE
```

CANIF memory and pointer classes.

Definition at line 114 of file Compiler_Cfg.h.

6.2.3.40 CRCU_CODE

```
#define CRCU_CODE
```

CRCU memory and pointer classes.

Definition at line 130 of file Compiler_Cfg.h.

6.2.3.41 CSEC_CODE

```
#define CSEC_CODE
```

CSEC memory and pointer classes.

Definition at line 146 of file Compiler_Cfg.h.

6.2.3.42 DEM_CODE

```
#define DEM_CODE
```

DEM memory and pointer classes.

Definition at line 162 of file Compiler_Cfg.h.

6.2.3.43 DET_CODE

```
#define DET_CODE
```

DET memory and pointer classes.

Definition at line 178 of file Compiler_Cfg.h.

6.2.3.44 DIO_CODE

```
#define DIO_CODE
```

DIO memory and pointer classes.

Definition at line 194 of file Compiler_Cfg.h.

6.2.3.45 EEP_CODE

```
#define EEP_CODE
```

EEP memory and pointer classes.

Definition at line 211 of file Compiler_Cfg.h.

6.2.3.46 ETH_CODE

```
#define ETH_CODE
```

ETH memory and pointer classes.

Definition at line 228 of file Compiler_Cfg.h.

6.2.3.47 ETHIF_CODE

```
#define ETHIF_CODE
```

ETH memory and pointer classes.

Definition at line 245 of file Compiler_Cfg.h.

6.2.3.48 ETHTRCV_CODE

```
#define ETHTRCV_CODE
```

ETH memory and pointer classes.

Definition at line 262 of file Compiler_Cfg.h.

6.2.3.49 FEE_CODE

```
#define FEE_CODE
```

FEE memory and pointer classes.

Definition at line 278 of file Compiler_Cfg.h.

6.2.3.50 FLS_CODE

```
#define FLS_CODE
```

FLS memory and pointer classes.

Definition at line 294 of file Compiler_Cfg.h.

6.2.3.51 FR_CODE

```
#define FR_CODE
```

FlexRay memory and pointer classes.

Definition at line 310 of file Compiler_Cfg.h.

6.2.3.52 GPT_CODE

```
#define GPT_CODE
```

GPT memory and pointer classes.

Definition at line 326 of file Compiler_Cfg.h.

6.2.3.53 ICU_CODE

```
#define ICU_CODE
```

ICU memory and pointer classes.

Definition at line 342 of file Compiler_Cfg.h.

6.2.3.54 I2C_CODE

```
#define I2C_CODE
```

I2C memory and pointer classes.

Definition at line 358 of file Compiler_Cfg.h.

6.2.3.55 LIN_CODE

```
#define LIN_CODE
```

LIN memory and pointer classes.

Definition at line 374 of file Compiler_Cfg.h.

6.2.3.56 LIN_43_LLCE_CODE

```
#define LIN_43_LLCE_CODE
```

LIN_43_LLCE memory and pointer classes.

Definition at line 390 of file Compiler_Cfg.h.

6.2.3.57 LINIF_CODE

```
#define LINIF_CODE
```

LIN memory and pointer classes.

Definition at line 406 of file Compiler_Cfg.h.

6.2.3.58 MCEM_CODE

```
#define MCEM_CODE
```

MCEM memory and pointer classes.

Definition at line 422 of file Compiler_Cfg.h.

6.2.3.59 MCL_CODE

```
#define MCL_CODE
```

MCL memory and pointer classes.

Definition at line 438 of file Compiler_Cfg.h.

6.2.3.60 MCU_CODE

```
#define MCU_CODE
```

MCU memory and pointer classes.

Definition at line 454 of file Compiler_Cfg.h.

6.2.3.61 PMIC_CODE

```
#define PMIC_CODE
```

PMIC memory and pointer classes.

Definition at line 470 of file Compiler_Cfg.h.

6.2.3.62 PORT_CODE

```
#define PORT_CODE
```

PORT memory and pointer classes.

Definition at line 486 of file Compiler_Cfg.h.

6.2.3.63 PWM_CODE

```
#define PWM_CODE
```

PWM memory and pointer classes.

Definition at line 502 of file Compiler_Cfg.h.

6.2.3.64 RAMTST_CODE

```
#define RAMTST_CODE
```

RamTST memory and pointer classes.

Definition at line 519 of file Compiler_Cfg.h.

6.2.3.65 SENT_CODE

```
#define SENT_CODE
```

SENT memory and pointer classes.

Definition at line 535 of file Compiler_Cfg.h.

6.2.3.66 SCHM_CODE

```
#define SCHM_CODE
```

SchM memory and pointer classes.

Definition at line 551 of file Compiler_Cfg.h.

6.2.3.67 SPI_CODE

```
#define SPI_CODE
```

SPI memory and pointer classes.

Definition at line 567 of file Compiler_Cfg.h.

6.2.3.68 TM_CODE

```
#define TM_CODE
```

TM memory and pointer classes.

Definition at line 583 of file Compiler_Cfg.h.

6.2.3.69 WDG_CODE

```
#define WDG_CODE
```

WDG memory and pointer classes.

Definition at line 599 of file Compiler_Cfg.h.

6.2.3.70 WDGIF_CODE

```
#define WDGIF_CODE
```

WDGIF memory and pointer classes.

Definition at line 615 of file Compiler_Cfg.h.

6.2.3.71 AUTOSAR_COMSTACKDATA

```
#define AUTOSAR_COMSTACKDATA
```

Define for ComStack Data.

Definition at line 630 of file Compiler_Cfg.h.

6.2.3.72 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 57 of file CompilerDefinition.h.

6.2.3.73 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 56 of file ComStack_Types.h.

6.2.3.74 NTFRSLT_OK

```
#define NTFRSLT_OK
```

Action has been successfully finished.

General return codes for NotifResultType

Definition at line 86 of file ComStack_Types.h.

6.2.3.75 NTFRSLT_E_NOT_OK

```
#define NTFRSLT_E_NOT_OK
```

Message not successfully received or sent out.

General return codes for NotifResultType

Definition at line 92 of file ComStack_Types.h.

6.2.3.76 NTFRSLT_E_TIMEOUT_A

```
#define NTFRSLT_E_TIMEOUT_A
```

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

General return codes for NotifResultType

Definition at line 98 of file ComStack_Types.h.

6.2.3.77 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 104 of file ComStack_Types.h.

6.2.3.78 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 110 of file ComStack_Types.h.

6.2.3.79 NTFRSLT_E_WRONG_SN

```
#define NTFRSLT_E_WRONG_SN
```

Unexpected sequence number (PCI.SN) value received.

General return codes for NotifResultType

Definition at line 116 of file ComStack_Types.h.

6.2.3.80 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 122 of file ComStack_Types.h.

6.2.3.81 NTFRSLT_E_UNEXP_PDU

```
#define NTFRSLT_E_UNEXP_PDU
```

Unexpected protocol data unit received.

General return codes for NotifResultType

Definition at line 128 of file ComStack_Types.h.

6.2.3.82 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 134 of file ComStack_Types.h.

6.2.3.83 NTFRSLT_E_ABORT

```
#define NTFRSLT_E_ABORT
```

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

General return codes for NotifResultType

Definition at line 140 of file ComStack_Types.h.

6.2.3.84 NTFRSLT_E_NO_BUFFER

```
#define NTFRSLT_E_NO_BUFFER
```

Indicates an abort of a transmission.

General return codes for NotifResultType

Definition at line 146 of file ComStack_Types.h.

6.2.3.85 NTFRSLT_E_CANCELATION_OK

```
#define NTFRSLT_E_CANCELATION_OK
```

Requested cancellation has been executed.

General return codes for NotifResultType

Definition at line 152 of file ComStack_Types.h.

6.2.3.86 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 160 of file ComStack_Types.h.

6.2.3.87 NTFRSLT_PARAMETER_OK

```
#define NTFRSLT_PARAMETER_OK
```

The parameter change request has been successfully executed.

General return codes for NotifResultType

Definition at line 166 of file ComStack_Types.h.

6.2.3.88 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 172 of file ComStack_Types.h.

6.2.3.89 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 178 of file ComStack_Types.h.

6.2.3.90 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 184 of file ComStack_Types.h.

6.2.3.91 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 57 of file Crc_MemMap.h.

6.2.3.92 MEMMAP_ERROR [10/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Crc_MemMap.h.

6.2.3.93 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 57 of file Crypto_MemMap.h.

6.2.3.94 MEMMAP_ERROR [11/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Crypto_MemMap.h.

6.2.3.95 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 57 of file Csec_MemMap.h.

6.2.3.96 MEMMAP_ERROR [12/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Csec_MemMap.h.

6.2.3.97 CXPI_MEMMAP_VENDOR_ID

```
#define CXPI_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 57 of file Cxpi_MemMap.h.

6.2.3.98 MEMMAP_ERROR [13/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Cxpi_MemMap.h.

6.2.3.99 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 57 of file Dem_MemMap.h.

6.2.3.100 MEMMAP_ERROR [14/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Dem_MemMap.h.

6.2.3.101 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 57 of file Det_MemMap.h.

6.2.3.102 MEMMAP_ERROR [15/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Det_MemMap.h.

6.2.3.103 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 57 of file Dio_MemMap.h.

6.2.3.104 MEMMAP_ERROR [16/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Dio_MemMap.h.

6.2.3.105 DPGA_MEMMAP_VENDOR_ID

```
#define DPGA_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 57 of file Dpga_MemMap.h.

6.2.3.106 MEMMAP_ERROR [17/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Dpga_MemMap.h.

6.2.3.107 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 57 of file Ecum_MemMap.h.

6.2.3.108 MEMMAP_ERROR [18/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Ecum_MemMap.h.

6.2.3.109 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 57 of file Eep_MemMap.h.

6.2.3.110 MEMMAP_ERROR [19/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Eep_MemMap.h.

6.2.3.111 Eth_43_ENET_MemMap_VENDOR_ID

```
#define Eth_43_ENET_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 57 of file Eth_43_ENET_MemMap.h.

6.2.3.112 MEMMAP_ERROR [20/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Eth_43_ENET_MemMap.h.

6.2.3.113 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 57 of file Eth_43_PFE_MemMap.h.

6.2.3.114 MEMMAP_ERROR [21/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Eth_43_PFE_MemMap.h.

6.2.3.115 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 58 of file Eth_GeneralTypes.h.

6.2.3.116 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 57 of file Eth_MemMap.h.

6.2.3.117 MEMMAP_ERROR [22/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Eth_MemMap.h.

6.2.3.118 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 57 of file EthSwitch_43_SJA1105P_MemMap.h.

6.2.3.119 MEMMAP_ERROR [23/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file EthSwitch_43_SJA1105P_MemMap.h.

6.2.3.120 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 57 of file EthSwt_43_SJA11XX_MemMap.h.

6.2.3.121 MEMMAP_ERROR [24/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file EthSwt_43_SJA11XX_MemMap.h.

6.2.3.122 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 57 of file EthTrcv_43_PHY_MemMap.h.

6.2.3.123 MEMMAP_ERROR [25/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file EthTrcv_43_PHY_MemMap.h.

6.2.3.124 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 57 of file EthTrcv_43_TJA110X_MemMap.h.

6.2.3.125 MEMMAP_ERROR [26/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file EthTrcv_43_TJA110X_MemMap.h.

6.2.3.126 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 57 of file Fee_MemMap.h.

6.2.3.127 MEMMAP_ERROR [27/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Fee_MemMap.h.

6.2.3.128 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 57 of file Fls_MemMap.h.

6.2.3.129 MEMMAP_ERROR [28/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Fls_MemMap.h.

6.2.3.130 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 57 of file Fr_43_LLCE_MemMap.h.

6.2.3.131 MEMMAP_ERROR [29/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Fr_43_LLCE_MemMap.h.

6.2.3.132 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 58 of file Fr_GeneralTypes.h.

6.2.3.133 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 71 of file Fr_GeneralTypes.h.

6.2.3.134 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 208 of file Fr_GeneralTypes.h.

6.2.3.135 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 57 of file Fr_MemMap.h.

6.2.3.136 MEMMAP_ERROR [30/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Fr_MemMap.h.

6.2.3.137 GDU_MEMMAP_VENDOR_ID

```
#define GDU_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 57 of file Gdu_MemMap.h.

6.2.3.138 MEMMAP_ERROR [31/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Gdu_MemMap.h.

6.2.3.139 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 57 of file Gpt_MemMap.h.

6.2.3.140 MEMMAP_ERROR [32/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Gpt_MemMap.h.

6.2.3.141 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 57 of file I2c_MemMap.h.

6.2.3.142 MEMMAP_ERROR [33/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file I2c_MemMap.h.

6.2.3.143 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 58 of file I2s_MemMap.h.

6.2.3.144 MEMMAP_ERROR [34/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 82 of file I2s_MemMap.h.

6.2.3.145 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 57 of file Icu_MemMap.h.

6.2.3.146 MEMMAP_ERROR [35/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Icu_MemMap.h.

6.2.3.147 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 57 of file Iseled_MemMap.h.

6.2.3.148 MEMMAP_ERROR [36/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Iseled_MemMap.h.

6.2.3.149 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 57 of file Lin_43_LLCE_MemMap.h.

6.2.3.150 MEMMAP_ERROR [37/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Lin_43_LLCE_MemMap.h.

6.2.3.151 LIN_43_LPUART_FLEXIO_MEMMAP_VENDOR_ID

```
#define LIN_43_LPUART_FLEXIO_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 58 of file Lin_43_LPUART_FLEXIO_MemMap.h.

6.2.3.152 MEMMAP_ERROR [38/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 82 of file Lin_43_LPUART_FLEXIO_MemMap.h.

6.2.3.153 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 59 of file Lin_GeneralTypes.h.

6.2.3.154 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 57 of file Lin_MemMap.h.

6.2.3.155 MEMMAP_ERROR [39/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Lin_MemMap.h.

6.2.3.156 LINTRCV_43_AE_MEMMAP_VENDOR_ID

```
#define LINTRCV_43_AE_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 57 of file LinTrcv_43_AE_MemMap.h.

6.2.3.157 MEMMAP_ERROR [40/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file LinTrcv_43_AE_MemMap.h.

6.2.3.158 MCAL_DATA_SYNC_BARRIER

```
#define MCAL_DATA_SYNC_BARRIER( )
```

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

Definition at line 279 of file Mcal.h.

6.2.3.159 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 283 of file Mcal.h.

6.2.3.160 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 57 of file Mcem_MemMap.h.

6.2.3.161 MEMMAP_ERROR [41/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Mcem_MemMap.h.

6.2.3.162 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 57 of file Mcl_MemMap.h.

6.2.3.163 MEMMAP_ERROR [42/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Mcl_MemMap.h.

6.2.3.164 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 57 of file Mcu_MemMap.h.

6.2.3.165 MEMMAP_ERROR [43/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Mcu_MemMap.h.

6.2.3.166 MEM_43_EEP_MEMMAP_VENDOR_ID

```
#define MEM_43_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 57 of file Mem_43_EEP_MemMap.h.

6.2.3.167 MEMMAP_ERROR [44/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Mem_43_EEP_MemMap.h.

6.2.3.168 MEM_43_EXFLS_MEMMAP_VENDOR_ID

```
#define MEM_43_EXFLS_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 57 of file Mem_43_EXFLS_MemMap.h.

6.2.3.169 MEMMAP_ERROR [45/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Mem_43_EXFLS_MemMap.h.

6.2.3.170 MEM_43_INFLS_MEMMAP_VENDOR_ID

```
#define MEM_43_INFLS_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 57 of file Mem_43_INFLS_MemMap.h.

6.2.3.171 MEMMAP_ERROR [46/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Mem_43_INFLS_MemMap.h.

6.2.3.172 MEMACC_MEMMAP_VENDOR_ID

```
#define MEMACC_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 57 of file MemAcc_MemMap.h.

6.2.3.173 MEMMAP_ERROR [47/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file MemAcc_MemMap.h.

6.2.3.174 OCOTP_MEMMAP_VENDOR_ID

```
#define OCOTP_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 57 of file Ocotp_MemMap.h.

6.2.3.175 MEMMAP_ERROR [48/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Ocotp_MemMap.h.

6.2.3.176 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 57 of file Ocu_MemMap.h.

6.2.3.177 MEMMAP_ERROR [49/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Ocu_MemMap.h.

6.2.3.178 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 57 of file Platform_MemMap.h.

6.2.3.179 MEMMAP_ERROR [50/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Platform_MemMap.h.

6.2.3.180 PLATFORM_VENDOR_ID

```
#define PLATFORM_VENDOR_ID
```

Note

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

Definition at line 65 of file PlatformTypes.h.

6.2.3.181 CPU_TYPE_8

```
#define CPU_TYPE_8
```

8bit Type Processor

Definition at line 88 of file PlatformTypes.h.

6.2.3.182 CPU_TYPE_16

```
#define CPU_TYPE_16
```

16bit Type Processor

Definition at line 94 of file PlatformTypes.h.

6.2.3.183 CPU_TYPE_32

```
#define CPU_TYPE_32
```

32bit Type Processor

Definition at line 100 of file PlatformTypes.h.

6.2.3.184 CPU_TYPE_64

```
#define CPU_TYPE_64
```

64bit Type Processor

Definition at line 106 of file PlatformTypes.h.

6.2.3.185 MSB_FIRST

```
#define MSB_FIRST
```

MSB First Processor.

Definition at line 112 of file PlatformTypes.h.

6.2.3.186 LSB_FIRST

```
#define LSB_FIRST
```

LSB First Processor.

Definition at line 118 of file PlatformTypes.h.

6.2.3.187 HIGH_BYTE_FIRST

```
#define HIGH_BYTE_FIRST
```

HIGH_BYTE_FIRST Processor.

Definition at line 124 of file PlatformTypes.h.

6.2.3.188 LOW_BYTE_FIRST

```
#define LOW_BYTE_FIRST
```

LOW_BYTE_FIRST Processor.

Definition at line 130 of file PlatformTypes.h.

6.2.3.189 CPU_TYPE

```
#define CPU_TYPE
```

Processor type.

Definition at line 136 of file PlatformTypes.h.

6.2.3.190 CPU_BIT_ORDER

```
#define CPU_BIT_ORDER
```

Bit order on register level.

Definition at line 145 of file PlatformTypes.h.

6.2.3.191 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_BYT↩ E_ORDER.

Definition at line 152 of file PlatformTypes.h.

6.2.3.192 TRUE

```
#define TRUE
```

Boolean true value.

Definition at line 160 of file PlatformTypes.h.

6.2.3.193 FALSE

```
#define FALSE
```

Boolean false value.

Definition at line 175 of file PlatformTypes.h.

6.2.3.194 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 57 of file Pmic_MemMap.h.

6.2.3.195 MEMMAP_ERROR [51/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Pmic_MemMap.h.

6.2.3.196 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 57 of file Port_MemMap.h.

6.2.3.197 MEMMAP_ERROR [52/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Port_MemMap.h.

6.2.3.198 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 57 of file Pwm_MemMap.h.

6.2.3.199 MEMMAP_ERROR [53/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Pwm_MemMap.h.

6.2.3.200 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 57 of file Qdec_MemMap.h.

6.2.3.201 MEMMAP_ERROR [54/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Qdec_MemMap.h.

6.2.3.202 MCAL_AXBS_REG_PROT_AVAILABLE

```
#define MCAL_AXBS_REG_PROT_AVAILABLE
```

Macros defined for the IPVs that are protected.

Definition at line 79 of file Reg_eSys.h.

6.2.3.203 RLM_REG_WRITE8

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

8 bits memory write macro

Definition at line 86 of file RegLockMacros.h.

6.2.3.204 RLM_REG_WRITE16

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

16 bits memory write macro.

Definition at line 90 of file RegLockMacros.h.

6.2.3.205 RLM_REG_WRITE32

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

32 bits memory write macro.

Definition at line 94 of file RegLockMacros.h.

6.2.3.206 RLM_REG_READ8

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

8 bits memory read macro.

Definition at line 100 of file RegLockMacros.h.

6.2.3.207 RLM_REG_READ16

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

16 bits memory read macro.

Definition at line 104 of file RegLockMacros.h.

6.2.3.208 RLM_REG_READ32

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

32 bits memory read macro.

Definition at line 108 of file RegLockMacros.h.

6.2.3.209 RLM_REG_BIT_CLEAR8

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

8 bits bits clearing macro.

Definition at line 113 of file RegLockMacros.h.

6.2.3.210 RLM_REG_BIT_CLEAR16

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

16 bits bits clearing macro.

Definition at line 117 of file RegLockMacros.h.

6.2.3.211 RLM_REG_BIT_CLEAR32

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

32 bits bits clearing macro.

Definition at line 121 of file RegLockMacros.h.

6.2.3.212 RLM_REG_BIT_GET8

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

8 bits bits getting macro.

Definition at line 127 of file RegLockMacros.h.

6.2.3.213 RLM_REG_BIT_GET16

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

16 bits bits getting macro.

Definition at line 131 of file RegLockMacros.h.

6.2.3.214 RLM_REG_BIT_GET32

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

32 bits bits getting macro.

Definition at line 135 of file RegLockMacros.h.

6.2.3.215 RLM_REG_BIT_SET8

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

8 bits bits setting macro.

Definition at line 141 of file RegLockMacros.h.

6.2.3.216 RLM_REG_BIT_SET16

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

16 bits bits setting macro.

Definition at line 145 of file RegLockMacros.h.

6.2.3.217 RLM_REG_BIT_SET32

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

32 bits bits setting macro.

Definition at line 149 of file RegLockMacros.h.

6.2.3.218 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) responsability to make sure that value has only "mask" bits set - (value&∼mask)==0

Definition at line 157 of file RegLockMacros.h.

6.2.3.219 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) responsability to make sure that value has only "mask" bits set - (value&∼mask)==0

Definition at line 163 of file RegLockMacros.h.

6.2.3.220 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) responsability to make sure that value has only "mask" bits set - (value&∼mask)==0

Definition at line 169 of file RegLockMacros.h.

6.2.3.221 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 186 of file RegLockMacros.h.

6.2.3.222 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 193 of file RegLockMacros.h.

6.2.3.223 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 200 of file RegLockMacros.h.

6.2.3.224 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 212 of file RegLockMacros.h.

6.2.3.225 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 233 of file RegLockMacros.h.

6.2.3.226 MIRRORED_ADDR_OFFSET_U32

```
#define MIRRORED_ADDR_OFFSET_U32
```

Offset to REG_PROT mirrored registers area of an IP module.

Definition at line 272 of file RegLockMacros.h.

6.2.3.227 SLBR_ADDR_OFFSET_U32

```
#define SLBR_ADDR_OFFSET_U32
```

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

Definition at line 312 of file RegLockMacros.h.

6.2.3.228 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 319 of file RegLockMacros.h.

6.2.3.229 GCR_OFFSET_U32

```
#define GCR_OFFSET_U32
```

Offset to baseAddress of the REG_PROT GCR register of an IP module.

Definition at line 371 of file RegLockMacros.h.

6.2.3.230 REGPROT_GCR_HLB_MASK_U32

```
#define REGPROT_GCR_HLB_MASK_U32
```

REG_PROT GCR bit masks.

Definition at line 377 of file RegLockMacros.h.

6.2.3.231 REGPROT_GCR_HLB_POS_U32

```
#define REGPROT_GCR_HLB_POS_U32
```

REG_PROT GCR bit positions.

Definition at line 383 of file RegLockMacros.h.

6.2.3.232 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	*baseAddr*	- base address of the IP the register belongs to
in	*regAddr*	- address of the register to soft lock
in	*prot_mem*	- protection size of the IP

Returns

void

Definition at line 413 of file RegLockMacros.h.

6.2.3.233 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	*baseAddr*	- base address of the IP the register belongs to
in	*regAddr*	- address of the register to soft unlock
in	*prot_mem*	- protection size of the IP

Returns

void

Definition at line 456 of file RegLockMacros.h.

6.2.3.234 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	*baseAddr*	- base address of the IP the register belongs to
in	*regAddr*	- address of the register for which to get soft lock bit status
in	*prot_mem*	- 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 496 of file RegLockMacros.h.

6.2.3.235 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	*baseAddr*	- base address of the IP the bit belongs to
in	*regAddr*	- address of the register the bit belongs to
in	*prot_mem*	- protection size of the IP
in	*mask*	- 8 bit mask of the bit

Returns

void

Definition at line 541 of file RegLockMacros.h.

6.2.3.236 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	*baseAddr*	- base address of the IP the bit belongs to
in	*regAddr*	- address of the register the bit belongs to
in	*prot_mem*	- protection size of the IP
in	*mask*	- 8 bit mask of the bit

Returns

void

Definition at line 567 of file RegLockMacros.h.

6.2.3.237 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	*baseAddr*	- base address of the IP the bit belongs to
in	*regAddr*	- address of the register the bit belongs to
in	*prot_mem*	- protection size of the IP
in	*mask*	- 8 bit mask of the bit

Returns

void

Definition at line 593 of file RegLockMacros.h.

6.2.3.238 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	*baseAddr*	- base address of the IP the bit belongs to
in	*regAddr*	- address of the register the bit belongs to
in	*prot_mem*	- protection size of the IP
in	*mask*	- 8 bit mask of the bit

Returns

void

Definition at line 619 of file RegLockMacros.h.

6.2.3.239 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	*baseAddr*	- base address of the IP the bit belongs to
in	*regAddr*	- address of the register the bit belongs to
in	*prot_mem*	- protection size of the IP
in	*mask*	- 8 bit mask of the bit

Returns

void

Definition at line 645 of file RegLockMacros.h.

6.2.3.240 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

Parameters

in	*baseAddr*	- base address of the IP the bit belongs to
in	*regAddr*	- address of the register the bit belongs to
in	*prot_mem*	- protection size of the IP
in	*mask*	- 8 bit mask of the bit

Returns

void

Definition at line 671 of file RegLockMacros.h.

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

Returns

void

Definition at line 697 of file RegLockMacros.h.

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

Returns

void

Definition at line 723 of file RegLockMacros.h.

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

Returns

void

Definition at line 749 of file RegLockMacros.h.

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

Returns

void

Definition at line 779 of file RegLockMacros.h.

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

Returns

void

Definition at line 810 of file RegLockMacros.h.

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

Returns

void

Definition at line 839 of file RegLockMacros.h.

6.2.3.247 SET_HARD_LOCK

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

Sets the hardlock bit of an IP module.

Parameters

in	*baseAddr*	- base address of the IP to be hard locked
in	*prot_mem*	- the protection size of the IP

Returns

void

Definition at line 851 of file RegLockMacros.h.

6.2.3.248 GET_HARD_LOCK

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

Reads the Hard Lock bit of an IP module.

Parameters

in	*baseAddr*	- base address of the IP for which hard lock status is read
in	*prot_mem*	- the protection size of the IP

Returns

uint8 - 1 if hard lock is enabled

- 0 if hard lock is disabled

Definition at line 864 of file RegLockMacros.h.

6.2.3.249 SET_USER_ACCESS_ALLOWED

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

Sets the User Access Allowed bit of an IP module.

Parameters

in	*baseAddr*	- base address of the IP for which UAA bit is set
in	*prot_mem*	- the protection size of the IP

Returns

void

Definition at line 887 of file RegLockMacros.h.

6.2.3.250 CLR_USER_ACCESS_ALLOWED

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

Clears the User Access Allowed bit of an IP module.

Parameters

in	*baseAddr*	- base address of the IP for which UAA bit is cleared
in	*prot_mem*	- the protection size of the IP

Returns

void

Definition at line 912 of file RegLockMacros.h.

6.2.3.251 GET_USER_ACCESS_ALLOWED

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

Reads the User Access Allowed bit of an IP module.

Parameters

in	*baseAddr*	- base address of the IP for which UAA is read
in	*prot_mem*	- 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 937 of file RegLockMacros.h.

6.2.3.252 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 57 of file Rm_MemMap.h.

6.2.3.253 MEMMAP_ERROR [55/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Rm_MemMap.h.

6.2.3.254 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 57 of file Rte_MemMap.h.

6.2.3.255 MEMMAP_ERROR [56/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Rte_MemMap.h.

6.2.3.256 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 57 of file Sbc_fs26_MemMap.h.

6.2.3.257 MEMMAP_ERROR [57/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Sbc_fs26_MemMap.h.

6.2.3.258 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 57 of file Sent_MemMap.h.

6.2.3.259 MEMMAP_ERROR [58/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Sent_MemMap.h.

6.2.3.260 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 57 of file Spi_MemMap.h.

6.2.3.261 MEMMAP_ERROR [59/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Spi_MemMap.h.

6.2.3.262 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 63 of file StandardTypes.h.

6.2.3.263 STD_HIGH

```
#define STD_HIGH
```

Physical state 5V or 3.3V.

Definition at line 97 of file StandardTypes.h.

6.2.3.264 STD_LOW

```
#define STD_LOW
```

Physical state 0V.

Definition at line 103 of file StandardTypes.h.

6.2.3.265 STD_ACTIVE

```
#define STD_ACTIVE
```

Logical state active.

Definition at line 109 of file StandardTypes.h.

6.2.3.266 STD_IDLE

```
#define STD_IDLE
```

Logical state idle.

Definition at line 115 of file StandardTypes.h.

6.2.3.267 STD_ON

```
#define STD_ON
```

ON State.

Definition at line 121 of file StandardTypes.h.

6.2.3.268 STD_OFF

```
#define STD_OFF
```

OFF state.

Definition at line 127 of file StandardTypes.h.

6.2.3.269 E_NOT_OK

```
#define E_NOT_OK
```

Return code for failure/error.

Definition at line 133 of file StandardTypes.h.

6.2.3.270 E_MEM_SERVICE_NOT_AVAIL

```
#define E_MEM_SERVICE_NOT_AVAIL
```

According to SWS_Mem_00070: In case a service is not relevant for a specific memory device technology, the service shall always return E_MEM_SERVICE_NOT_AVAIL. The return code is implementation-defined (see SWS_Std_00011 and SWS_Std_00005).

Definition at line 141 of file StandardTypes.h.

6.2.3.271 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 165 of file StandardTypes.h.

6.2.3.272 E_OK

```
#define E_OK
```

Success return code.

Definition at line 169 of file StandardTypes.h.

6.2.3.273 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 57 of file Uart_MemMap.h.

6.2.3.274 MEMMAP_ERROR [60/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Uart_MemMap.h.

6.2.3.275 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 57 of file Wdg_43_fs26_MemMap.h.

6.2.3.276 MEMMAP_ERROR [61/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 of file Wdg_43_fs26_MemMap.h.

6.2.3.277 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 57 of file Wdg_MemMap.h.

6.2.3.278 MEMMAP_ERROR [62/62]

```
#define MEMMAP_ERROR
```

Symbol used for checking correctness of the includes.

Definition at line 81 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 theframe 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 186 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 224 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 68 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 75 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 243 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 250 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 257 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 287 of file ComStack_Types.h.

6.2.4.9 CbkHandleIdType

```
typedef uint16 CbkHandleIdType
```

Used for the handle Ids of Com and LdCom user callbacks.

Definition at line 293 of file ComStack_Types.h.

6.2.4.10 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 grater than 1500 represent the type (i.e. 0x800 = IP).

Definition at line 380 of file Eth_GeneralTypes.h.

6.2.4.11 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 388 of file Eth_GeneralTypes.h.

6.2.4.12 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 395 of file Eth_GeneralTypes.h.

6.2.4.13 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 241 of file Lin_GeneralTypes.h.

6.2.4.14 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 249 of file Lin_GeneralTypes.h.

6.2.4.15 boolean

```
typedef bool boolean
```

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

Definition at line 200 of file PlatformTypes.h.

6.2.4.16 uint8

```
typedef uint8_t uint8
```

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

Definition at line 208 of file PlatformTypes.h.

6.2.4.17 uint16

```
typedef uint16_t uint16
```

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

Definition at line 215 of file PlatformTypes.h.

6.2.4.18 uint32

```
typedef uint32_t uint32
```

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

Definition at line 222 of file PlatformTypes.h.

6.2.4.19 uint64

```
typedef uint64_t uint64
```

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

Definition at line 229 of file PlatformTypes.h.

6.2.4.20 sint8

```
typedef int8_t sint8
```

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

Definition at line 237 of file PlatformTypes.h.

6.2.4.21 sint16

```
typedef int16_t sint16
```

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

Definition at line 244 of file PlatformTypes.h.

6.2.4.22 sint32

```
typedef int32_t sint32
```

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

Definition at line 251 of file PlatformTypes.h.

6.2.4.23 sint64

```
typedef int64_t sint64
```

Signed 64 bit integer with range of -9223372036854775808..9223372036854775807 (0x8000000000000000..0x7FFF↩
FFFFFFFFFF)- 63 bit + 1 sign bit.

Definition at line 258 of file PlatformTypes.h.

6.2.4.24 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 265 of file PlatformTypes.h.

6.2.4.25 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 272 of file PlatformTypes.h.

6.2.4.26 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 279 of file PlatformTypes.h.

6.2.4.27 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 286 of file PlatformTypes.h.

6.2.4.28 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 293 of file PlatformTypes.h.

6.2.4.29 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 300 of file PlatformTypes.h.

6.2.4.30 float32

```
typedef float float32
```

32bit long floating point data type

Definition at line 306 of file PlatformTypes.h.

6.2.4.31 float64

```
typedef double float64
```

64bit long floating point data type

Definition at line 312 of file PlatformTypes.h.

6.2.4.32 StatusType

```
typedef uint8 StatusType
```

This type is defined for OSEK compliance.

Definition at line 173 of file StandardTypes.h.

6.2.4.33 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 181 of file StandardTypes.h.

6.2.4.34 Std_TransformerErrorCode

```
typedef uint8 Std_TransformerErrorCode
```

The type of the Std_TransformerError.

Definition at line 201 of file StandardTypes.h.

6.2.4.35 Std_ExtractProtocolHeaderFieldsType

```
typedef Std_ReturnType(* Std_ExtractProtocolHeaderFieldsType) (const uint8 *buffer, uint32 bufferLength,
Std_MessageTypeType *messageType, Std_MessageResultType *messageResult)
```

Type for the function pointer to extract the relevant protocol header fields of the message and the type of the message result of a transformer. - At the time being, this is limited to the types used for C/S communication (i.e., REQUEST and RESPONSE and OK and ERROR)

Definition at line 282 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 91 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 104 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 116 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 128 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 141 of file Can_GeneralTypes.h.

6.2.5.6 Can_ErrorType

enum Can_ErrorType

CAN Controller Error Types of operation.

The enumeration represents a superset of CAN Error Types which typical CAN HW is able to report. That means not all CAN HW will be able to support the complete set.

Definition at line 158 of file Can_GeneralTypes.h.

6.2.5.7 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 193 of file ComStack_Types.h.

6.2.5.8 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 c 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_CONFPENDING	Indicates that the previously copied data must remain in the TP.

Definition at line 205 of file ComStack_Types.h.

6.2.5.9 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 219 of file ComStack_Types.h.

6.2.5.10 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 230 of file ComStack_Types.h.

6.2.5.11 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 85 of file Eth_GeneralTypes.h.

6.2.5.12 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
ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST	Controller is active with wakeup request.

Definition at line 98 of file Eth_GeneralTypes.h.

6.2.5.13 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.

6.2.5.14 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.

6.2.5.15 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.

6.2.5.16 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.

6.2.5.17 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.

6.2.5.18 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.

6.2.5.19 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.

6.2.5.20 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.

6.2.5.21 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.

6.2.5.22 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.

6.2.5.23 EthTrcv_PhyTestModeType

enum EthTrcv_PhyTestModeType

Describes the possible PHY test modes.

Enumerator

ETHTRCV_PHYTESTMODE_NONE	normal operation test transmitter droop

Enumerator

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.

6.2.5.24 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.

6.2.5.25 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.

6.2.5.26 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.

6.2.5.27 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_PORTINIT_COMPLETED	Port initialization is completed.
ETHSWT_STATE_ACTIVE	The Eth Switch driver is active.

Definition at line 329 of file Eth_GeneralTypes.h.

6.2.5.28 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 341 of file Eth_GeneralTypes.h.

6.2.5.29 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 352 of file Eth_GeneralTypes.h.

6.2.5.30 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 362 of file Eth_GeneralTypes.h.

6.2.5.31 Fr_TxLPduStatusType

enum Fr_TxLPduStatusType

Transmit resource status is stored to variable of this type.

Definition at line 145 of file Fr_GeneralTypes.h.

6.2.5.32 Fr_RxLPduStatusType

enum Fr_RxLPduStatusType

Transmit resource status is stored to variable of this type.

Definition at line 157 of file Fr_GeneralTypes.h.

6.2.5.33 Fr_POCStateType

enum Fr_POCStateType

Variables of this type are used to store the POC:state of the controller.

Definition at line 170 of file Fr_GeneralTypes.h.

6.2.5.34 Fr_SlotModeType

enum Fr_SlotModeType

This type is used to store the slot mode of the controller.

Definition at line 195 of file Fr_GeneralTypes.h.

6.2.5.35 Fr_ErrorModeType

enum Fr_ErrorModeType

Variables of this type are used for storage of FlexRay controller error mode.

Definition at line 215 of file Fr_GeneralTypes.h.

6.2.5.36 Fr_WakeupStatusType

enum Fr_WakeupStatusType

Variable of this type is used to query the FlexRay controller Wakeup status.

Definition at line 229 of file Fr_GeneralTypes.h.

6.2.5.37 Fr_StartupStateType

enum Fr_StartupStateType

Variable of this type is used to query the FlexRay controller Startup state.

Definition at line 246 of file Fr_GeneralTypes.h.

6.2.5.38 Fr_ChannelType

enum Fr_ChannelType

This type is used to select the channel.

Definition at line 273 of file Fr_GeneralTypes.h.

6.2.5.39 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 82 of file Lin_GeneralTypes.h.

6.2.5.40 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 96 of file Lin_GeneralTypes.h.

6.2.5.41 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).
LIN_TX_HEADER_ERROR	Erroneous header transmission such as: • Mismatch between sent and read back data • Identifier parity error • Physical bus error.
LIN_TX_ERROR	Erroneous transmission such as: • Mismatch between sent and read back data • Physical bus error.
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: • Framing error • Overrun error • Checksum error • Short response.

Enumerator

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;. • The related LIN channel is ready to transmit next header • No data from previous frame available (e.g. after initialization).
LIN_CH_SLEEP	Sleep mode operation;. • In this mode wake-up detection from slave nodes is enabled.

Definition at line 116 of file Lin_GeneralTypes.h.

6.2.5.42 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 167 of file Lin_GeneralTypes.h.

6.2.5.43 LinTrcv_TrcvWakeupModeType

`enum LinTrcv_TrcvWakeupModeType`

Transceiver Wake Up Mode Types.

Wake up operating modes of the LIN Transceiver Driver.

Enumerator

LINTRCV_WUMODE_ENABLE	The notification for wakeup events is enabled on the addressed network.
LINTRCV_WUMODE_DISABLE	The notification for wakeup events is disabled on the addressed network.
LINTRCV_WUMODE_CLEAR	A stored wakeup event is cleared on the addressed network.

Definition at line 188 of file Lin_GeneralTypes.h.

6.2.5.44 LinTrcv_TrcvWakeupReasonType

`enum LinTrcv_TrcvWakeupReasonType`

Transceiver Wake Up Reason Types.

This type denotes the wake up reason detected by the LIN transceiver.

Enumerator

LINTRCV_WU_ERROR	Due to an error wake up reason was not detected.
LINTRCV_WU_NOT_SUPPORTED	The transceiver does not support any information for the wake up reason.
LINTRCV_WU_BY_BUS	The transceiver has detected, that the network has caused the wake up of the ECU.
LINTRCV_WU_BY_PIN	The transceiver has detected a wake-up event at one of the transceiver's pins (not at the LIN bus).
LINTRCV_WU_INTERNALLY	The transceiver has detected, that the network has been woken up by the ECU via a request to NORMAL mode.
LINTRCV_WU_RESET	The transceiver has detected, that the wake up is due to an ECU reset.
LINTRCV_WU_POWER_ON	The transceiver has detected, that the wake up is due to an ECU reset after power on.

Definition at line 203 of file Lin_GeneralTypes.h.

6.2.5.45 LinTrcv_TrcvModeType

`enum LinTrcv_TrcvModeType`

Operating modes of the LIN Transceiver Driver.

Enumerator

LINTRCV_TRCV_MODE_NORMAL	Transceiver mode NORMAL.
LINTRCV_TRCV_MODE_STANDBY	Transceiver mode STANDBY.
LINTRCV_TRCV_MODE_SLEEP	Transceiver mode SLEEP.

Definition at line 225 of file Lin_GeneralTypes.h.

6.2.5.46 Std_TransformerClass

`enum Std_TransformerClass`

The Std_TransformerClass represents the transformer class in which the error occurred.

Enumerator

STD_TRANSFORMER_UNSPECIFIED	Transformer of a unspecified transformer class.
STD_TRANSFORMER_SERIALIZER	Transformer of a serializer class.
STD_TRANSFORMER_SAFETY	Transformer of a safety class.
STD_TRANSFORMER_SECURITY	Transformer of a security class.
STD_TRANSFORMER_CUSTOM	Transformer of a custom class not standardized by AUTOSAR.

Definition at line 207 of file StandardTypes.h.

6.2.5.47 Std_TransformerForwardCode

`enum Std_TransformerForwardCode`

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

Enumerator

ERROR_OK	No specific error to be injected.
E_SAFETY_INVALID_REP	Repeat the last used sequence number.
E_SAFETY_INVALID_SEQ	Generate a deliberately wrong CRC.
E_SAFETY_INVALID_CRC	Use a wrong sequence number.

Definition at line 232 of file StandardTypes.h.

6.2.5.48 Std_MessageTypeType

enum Std_MessageTypeType

This type is used to encode the different type of messages. - Currently this encoding is limited to the distinction between requests and responses in C/S communication.

Enumerator

STD_MESSAGETYPE_REQUEST	Message type for a request message.
STD_MESSAGETYPE_RESPONSE	Message type for a response message.

Definition at line 257 of file StandardTypes.h.

6.2.5.49 Std_MessageResultType

enum Std_MessageResultType

This type is used to encode different types of results for response messages. - Currently this encoding is limited to the distinction between OK and ERROR responses.

Enumerator

STD_MESSAGERESULT_OK	STD_MESSAGERESULT_OK.
STD_MESSAGERESULT_ERROR	Messageresult for an ERROR response.

Definition at line 269 of file StandardTypes.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 197 of file Can_GeneralTypes.h.

6.2.6.2 swPduHandle [1/2]

```
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 201 of file Can_GeneralTypes.h.

6.2.6.3 length [1/2]

```
uint8 length
```

DLC = Data Length Code (part of L-PDU that describes the SDU length).

Definition at line 206 of file Can_GeneralTypes.h.

6.2.6.4 sdu [1/2]

```
uint8* sdu
```

CAN L-SDU = Link Layer Service Data Unit. Data that is transported inside the L-PDU.

Definition at line 208 of file Can_GeneralTypes.h.

6.2.6.5 CanId

```
Can_IdType CanId
```

Standard/Extended CAN ID of CAN L-PDU.

Definition at line 236 of file Can_GeneralTypes.h.

6.2.6.6 Hoh [1/2]

```
Can_HwHandleType Hoh
```

ID of the corresponding Hardware Object Range.

Definition at line 238 of file Can_GeneralTypes.h.

6.2.6.7 ControllerId [1/2]

```
uint8 ControllerId
```

ControllerId provided by CanIf clearly identify the corresponding controller.

Definition at line 240 of file Can_GeneralTypes.h.

6.2.6.8 nanoseconds [1/2]

```
uint32 nanoseconds
```

Nanoseconds part of the time.

Definition at line 251 of file Can_GeneralTypes.h.

6.2.6.9 seconds [1/2]

```
uint32 seconds
```

Seconds part of the time.

Definition at line 252 of file Can_GeneralTypes.h.

6.2.6.10 PriorityId

```
uint16 PriorityId
```

Priority ID of a CAN XL message.

Definition at line 259 of file Can_GeneralTypes.h.

6.2.6.11 Vcid

```
uint16 Vcid
```

VCID of a CAN XL message.

Definition at line 261 of file Can_GeneralTypes.h.

6.2.6.12 SduType

```
uint8 SduType
```

SDU type of a CAN XL message.

Definition at line 263 of file Can_GeneralTypes.h.

6.2.6.13 AcceptanceField

```
uint32 AcceptanceField
```

Acceptance field of a CAN XL message.

Definition at line 265 of file Can_GeneralTypes.h.

6.2.6.14 Sec

```
uint8 Sec
```

Simple extended content field of a CAN XL message.

Definition at line 267 of file Can_GeneralTypes.h.

6.2.6.15 swPduHandle [2/2]

```
PduIdType swPduHandle
```

Contains the PDU ID.

Definition at line 273 of file Can_GeneralTypes.h.

6.2.6.16 length [2/2]

```
uint16 length
```

Length of the data.

Definition at line 275 of file Can_GeneralTypes.h.

6.2.6.17 sdu [2/2]

```
uint8* sdu
```

SDU data pointer.

Definition at line 277 of file Can_GeneralTypes.h.

6.2.6.18 XLParams [1/2]

```
CanXL_Params* XLParams
```

Pointer to CAN XL params.

Definition at line 279 of file Can_GeneralTypes.h.

6.2.6.19 XLParams [2/2]

```
CanXL_Params* XLParams
```

Pointer to CAN XL params.

Definition at line 285 of file Can_GeneralTypes.h.

6.2.6.20 ControllerId [2/2]

```
uint8 ControllerId
```

ControllerId provided by CanIf, identifies the corresponding CAN XL controller.

Definition at line 287 of file Can_GeneralTypes.h.

6.2.6.21 Hoh [2/2]

```
Can_HwHandleType Hoh
```

ID of the corresponding CAN XL Hardware Object Range.

Definition at line 289 of file Can_GeneralTypes.h.

6.2.6.22 nanoseconds [2/2]

```
uint32 nanoseconds
```

Nanoseconds part of the time.

Definition at line 413 of file Eth_GeneralTypes.h.

6.2.6.23 seconds [2/2]

```
uint32 seconds
```

32 bit LSB of the 48 bits Seconds part of the time

Definition at line 414 of file Eth_GeneralTypes.h.

6.2.6.24 secondsHi

```
uint16 secondsHi
```

16 bit MSB of the 48 bits Seconds part of the time

Definition at line 415 of file Eth_GeneralTypes.h.

6.2.6.25 diff

```
Eth_TimeStampType diff
```

diff time difference

Definition at line 427 of file Eth_GeneralTypes.h.

6.2.6.26 sign

```
boolean sign
```

Positive (True) Or negative (False) time.

Definition at line 428 of file Eth_GeneralTypes.h.

6.2.6.27 IngressTimeStampDelta

```
Eth_TimeIntDiffType IngressTimeStampDelta
```

IngressTimeStampSync2 -IngressTimeStampSync1.

Definition at line 438 of file Eth_GeneralTypes.h.

6.2.6.28 OriginTimeStampDelta

```
Eth_TimeIntDiffType OriginTimeStampDelta
```

OriginTimeStampSync2[FUP2]-OriginTimeStampSync1[FUP1].

Definition at line 439 of file Eth_GeneralTypes.h.

6.2.6.29 SwitchIdx

```
uint8 SwitchIdx
```

Switch index.

Definition at line 542 of file Eth_GeneralTypes.h.

6.2.6.30 SwitchPortIdx

```
uint8 SwitchPortIdx
```

Port index of the switch.

Definition at line 543 of file Eth_GeneralTypes.h.

6.2.6.31 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 0,0,0,0,0,0, no source MAC address filtering shall take place.

Definition at line 551 of file Eth_GeneralTypes.h.

6.2.6.32 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 0,0,0,0,0,0, no destination MAC address filtering shall take place.

Definition at line 552 of file Eth_GeneralTypes.h.

6.2.6.33 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 553 of file Eth_GeneralTypes.h.

6.2.6.34 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 554 of file Eth_GeneralTypes.h.

6.2.6.35 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 555 of file Eth_GeneralTypes.h.

6.2.6.36 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 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.

Definition at line 556 of file Eth_GeneralTypes.h.

6.2.6.37 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 557 of file Eth_GeneralTypes.h.

6.2.6.38 CapturePortIdx

```
uint8 CapturePortIdx
```

Specifies the Ethernet switch port which capture the mirrored traffic.

Definition at line 558 of file Eth_GeneralTypes.h.

6.2.6.39 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 559 of file Eth_GeneralTypes.h.

6.2.6.40 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 560 of file Eth_GeneralTypes.h.

6.2.6.41 IngressTimestampValid

`Std_ReturnType IngressTimestampValid`

IngressTimestampValid shall be set to E_NOT_OK if ingress timestamp is not available.

Definition at line 569 of file Eth_GeneralTypes.h.

6.2.6.42 EgressTimestampValid

`Std_ReturnType EgressTimestampValid`

EgressTimestampValid shall be set to E_NOT_OK if ingress timestamp is not available.

Definition at line 570 of file Eth_GeneralTypes.h.

6.2.6.43 MgmtInfoValid

`Std_ReturnType MgmtInfoValid`

MgmtInfoValid shall be set to E_NOT_OK if ingress timestamp is not available(e.g. timeout).

Definition at line 571 of file Eth_GeneralTypes.h.

6.2.6.44 Validation

`EthSwt_MgmtObjectValidType Validation`

The validation information for the mgmt_obj.

Definition at line 579 of file Eth_GeneralTypes.h.

6.2.6.45 IngressTimestamp

```
Eth_TimeStampType IngressTimestamp
```

The ingress timestamp value out of the switch.

Definition at line 580 of file Eth_GeneralTypes.h.

6.2.6.46 EgressTimestamp

```
Eth_TimeStampType EgressTimestamp
```

The egress timestamp value out of the switch.

Definition at line 581 of file Eth_GeneralTypes.h.

6.2.6.47 MgmtInfo

```
EthSwt_MgmtInfoType MgmtInfo
```

Received/Transmitted Management information of the switches.

Definition at line 582 of file Eth_GeneralTypes.h.

6.2.6.48 Ownership

```
EthSwt_MgmtOwner Ownership
```

The ownership of MgmtObj.

Definition at line 583 of file Eth_GeneralTypes.h.

6.2.6.49 Pid

```
Lin_FramePidType Pid
```

LIN frame identifier.

Definition at line 260 of file Lin_GeneralTypes.h.

6.2.6.50 Cs

```
Lin_FrameCsModelType Cs
```

Checksum model type.

Definition at line 261 of file Lin_GeneralTypes.h.

6.2.6.51 Drc

```
Lin_FrameResponseType Drc
```

Response type.

Definition at line 262 of file Lin_GeneralTypes.h.

6.2.6.52 Dl

```
Lin_FrameDlType Dl
```

Data length.

Definition at line 263 of file Lin_GeneralTypes.h.

6.2.6.53 SduPtr

```
uint8* SduPtr
```

Pointer to Sdu.

Definition at line 264 of file Lin_GeneralTypes.h.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

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

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