

Document Title	Specification of Ethernet Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	430

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R25-11

Document Change History			
Date	Release	Changed by	Description
2025-11-27	R25-11	AUTOSAR Release Management	• Added MACsec support • Added VLAN filtering to Eth_UpdatePhysAddrFilter • Eth_Transmit, Eth_Receive and Eth_TxConfirmation no longer raise ETH_E_INV_MODE if controller mode is not ETH_MODE_ACTIVE • Marked Eth_GetIngressTimeStamp as obsolete • Removed Eth_GetCurrentTime • Removed buffer handling
2024-11-27	R24-11	AUTOSAR Release Management	• Add EgressSchedulerAlgorithm at Ethernet driver
2023-11-23	R23-11	AUTOSAR Release Management	• Adaptation to the Deterministic Communication with TSN • Editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	• Entreny limitation • Editorial changes





2021-11-25	R21-11	AUTOSAR Release Management	• New runtime error and return code handling modified • Silent communication added • EthGetRxStatsApi added • Support SPI interface for external devices
2020-11-30	R20-11	AUTOSAR Release Management	• Eth_GeneralTypes removed from imported module list • EthGetDropCountApi renamed to EthGetCounterValuesApi • Buffer handling • WakeOnDataLine • Details MII Read/Right for Clause 22
2019-11-29	R19-11	AUTOSAR Release Management	• 2500Mbit Ethernet Support • Eth_TimeStampQualType base type defined • Changed Document Status from final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Support of host controllers with multiple cores • Asynchronous frame transmission • Timestamp improvements • Multicast MAC address handling in Switches
2017-12-08	4.3.1	AUTOSAR Release Management	• Minor corrections and adaptations
2016-11-30	4.3.0	AUTOSAR Release Management	• Quality of Service (QoS) support • Ethernet statistics counter access
2015-07-31	4.2.2	AUTOSAR Release Management	• Eth_ControllerInit functionality merged into Eth_Init API • Development Error Tracer renamed to Default Error Tracer • IRQ handler API removed





2014-10-31	4.2.1	AUTOSAR Release Management	• Change from Synchronous to Asynchronous API • gPTP Timestamp Support • Enhanced Production Errors • Changed Access to Statistic Frame Handling Registers
2014-03-31	4.1.3	AUTOSAR Release Management	• Introduction of periodic call to Eth_SetControllerMode • Support of VLANs (Virtual Local Area Networks) • Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Introduction of Eth_GeneralTypes.h • Support of API deviation for asynchronous implementation • Changes in API of EthIf_ProvideTxBuffer and EthIf_SetPhysAddr • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Configurable MAC address based filtering • Detection of lost Ethernet frames • Buffer handling enhancement
2011-12-22	4.0.3	AUTOSAR Administration	• Description of buffer behaviour in Eth_SetControllerMode extended





2010-09-30	3.1.5	AUTOSAR Administration	• Enhanced development error detection for active controller before controller access • Further post-build configurable parameters • Improved description of 'XxxCtrlIdx' semantics • 'Instance ID' removed from Version Info (concerns Eth_GetVersionInfo API) • Additional development error in Eth_GetVersionInfo API
2010-02-02	3.1.4	AUTOSAR Administration	• Initial Release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1	Introduction and functional overview	11
2	Acronyms, Abbreviations and Definition	13
2.1	Acronyms and Abbreviations	13
2.2	Definitions	13
2.2.1	Hardware supported data transfert	13
2.2.2	Data transfer session handle	13
3	Related documentation	14
3.1	Input documents	14
3.2	Related standards and norms	15
3.3	Related specification	15
4	Constraints and assumptions	16
4.1	Limitations	16
4.2	Applicability to car domains	16
5	Dependencies to other modules	17
5.1	Driver Services	17
6	Requirements Tracing	19
7	Functional specification	23
7.1	Ethernet BSW stack	23
7.1.1	Switch	23
7.1.2	External MAC	24
7.1.3	Indexing scheme	25
7.1.4	Requirements	26
7.1.5	Initialization	27
7.1.6	Communication	28
7.1.6.1	Transmission	28
7.1.6.2	Transmission confirmation	32
7.1.6.3	Reception	34
7.1.6.4	Hardware supported data transfer	35
7.1.7	Queue handling	36
7.1.7.1	Ingress queue	37
7.1.7.2	Egress queue	42
7.1.8	Support of frame preemption	54
7.1.9	HW Clock Handling	55
7.1.9.1	HW Timestamping	55
7.1.9.2	Adjustable PTP HW Clock (PHC)	56
7.1.9.3	Generation of a Pulse-Per-Second (PPS) Signal	59
7.1.10	MACsec support	61
7.1.11	Configuration description	71

7.2 Error Classification	72
7.2.1 Development Errors	72
7.2.2 Runtime Errors	72
7.2.3 Production Errors	73
7.2.4 Extended Production Errors	73
8 API specification	77
8.1 API Parameters Checking	77
8.2 Imported types	77
8.3 Type definitions	78
8.3.1 Eth_ConfigType	78
8.3.2 Eth_ModeType	78
8.3.3 Eth_StateType	79
8.3.4 Eth_FrameType	79
8.3.5 Eth_DataType	80
8.3.6 Eth_BufIdxType	80
8.3.7 Eth_RxStatusType	80
8.3.8 Eth_FilterActionType	81
8.3.9 Eth_MacVlanType	81
8.3.10 Eth_CounterType	82
8.3.11 Eth_RxStatsType	84
8.3.12 Eth_TxStatsType	86
8.3.13 Eth_TxErrorCounterValuesType	87
8.3.14 Eth_SpiStatusType	88
8.3.15 Eth_StreamStatisticCounterType	89
8.4 Function definitions	89
8.4.1 Eth_Init	89
8.4.2 Eth_SetControllerMode	91
8.4.3 Eth_GetControllerMode	92
8.4.4 Eth_GetPhysAddr	92
8.4.5 Eth_SetPhysAddr	93
8.4.6 Eth_UpdatePhysAddrFilter	93
8.4.7 Eth_WriteMii	95
8.4.8 Eth_ReadMii	96
8.4.9 Eth_GetCounterValues	97
8.4.10 Eth_GetRxStats	98
8.4.11 Eth_GetTxStats	99
8.4.12 Eth_GetTxErrorCounterValues	99
8.4.13 Eth_GetSpiStatus	100
8.4.14 Eth_GetCurrentTimeTuple	101
8.4.15 Eth_SetPhcTime	102
8.4.16 Eth_SetPhcCorrection	103
8.4.17 Eth_GetPhcTime	104
8.4.18 Eth_SetPpsSignalMode	105

8.4.19 Eth_EnableEgressTimeStamp	106
8.4.20 Eth_GetEgressTimeStamp	106
8.4.21 Eth_GetIngressTimeStamp	107
8.4.22 Eth_ProvideTxBuffer	108
8.4.23 Eth_Transmit	109
8.4.24 Eth_Receive	110
8.4.25 Eth_ImmediateTransmit	111
8.4.26 Eth_ReleaseRxBuffer	111
8.4.27 Eth_TxConfirmation	112
8.4.28 Eth_GetVersionInfo	113
8.4.29 Eth_ReadMmd	113
8.4.30 Eth_WriteMmd	114
8.4.31 Eth_MacSecUpdateSecY	115
8.4.32 Eth_MacSecInitRxSc	115
8.4.33 Eth_MacSecResetRxSc	116
8.4.34 Eth_MacSecAddTxSa	117
8.4.35 Eth_MacSecUpdateTxSa	118
8.4.36 Eth_MacSecDeleteTxSa	118
8.4.37 Eth_MacSecAddRxSa	119
8.4.38 Eth_MacSecUpdateRxSa	120
8.4.39 Eth_MacSecDeleteRxSa	121
8.4.40 Eth_MacSecGetTxSaNextPn	121
8.4.41 Eth_MacSecSetControlledPortEnabled	122
8.4.42 Eth_MacSecGetMacSecStatistics	123
8.5 Callback notifications	123
8.6 Scheduled functions	123
8.6.1 Eth_MainFunction	123
8.7 Expected interfaces	124
8.7.1 Mandatory Interfaces	124
8.7.2 Optional Interfaces	125
8.7.3 Configurable interfaces	126
8.7.3.1 Eth_<IngressQueueHandlerFunction>	126
9 Sequence diagrams	128
10 Configuration specification	129
10.1 How to read this chapter	129
10.2 Containers and configuration parameters	129
10.2.1 Eth	129
10.2.2 EthConfigSet	130
10.2.3 EthCtrlConfig	130
10.2.4 EthCtrlClk and EthClkUnit	141
10.2.5 EthCtrlPulsePerSecondConfig	146
10.2.6 EthCtrlConfigEgress	149

10.2.6.1	EthCtrlConfigEgressQueue - DRAFT	150
10.2.6.2	EthCtrlConfigEgressQueueSortingType - DRAFT	152
10.2.6.3	EthCtrlConfigEgressQueueSortingEntry - DRAFT	153
10.2.6.4	EthCtrlConfigEgressQueueTransmissionSelection - DRAFT	155
10.2.6.5	EthCtrlConfigEgressQueueTransmissionSelectionCBSConfig - DRAFT	159
10.2.6.6	EthCtrlConfigScheduler	161
10.2.6.7	EthCtrlConfigSchedulerPredecessor	162
10.2.7	EthCtrlConfigIngress	163
10.2.7.1	EthCtrlConfigIngressQueue - DRAFT	164
10.2.8	EthCtrlConfigSpiConfiguration	175
10.2.9	EthCtrlMacSecConfig	183
10.2.10	EthDemEventParameterRefs	185
10.2.11	EthGeneral	190
10.2.11.1	EthCtrlOffloading	196
10.3	Published Information	197
A	Not applicable requirements	198
B	Change history of AUTOSAR traceable items	199
B.1	Traceable item history of this document according to AUTOSAR Release R25-11	199
B.1.1	Added Specification Items in R25-11	199
B.1.2	Changed Specification Items in R25-11	199
B.1.3	Deleted Specification Items in R25-11	199
B.1.4	Added Constraints in R25-11	200
B.1.5	Changed Constraints in R25-11	200
B.1.6	Deleted Constraints in R25-11	200
B.2	Traceable item history of this document according to AUTOSAR Release R24-11	200
B.2.1	Added Specification Items in R24-11	200
B.2.2	Changed Specification Items in R24-11	201
B.2.3	Deleted Specification Items in R24-11	201
B.2.4	Added Constraints in R24-11	201
B.2.5	Changed Constraints in R24-11	201
B.2.6	Deleted Constraints in R24-11	202
B.3	Traceable item history of this document according to AUTOSAR Release R23-11	202
B.3.1	Added Specification Items in R23-11	202
B.3.2	Changed Specification Items in R23-11	202
B.3.3	Deleted Specification Items in R23-11	202
B.3.4	Added Constraints in R23-11	203
B.3.5	Changed Constraints in R23-11	203
B.3.6	Deleted Constraints in R23-11	203

Known Limitations

Currently, chapter [5](#) does not describe the versions of dependent modules. Thus, a version check will extend the chapter.

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module Ethernet Driver.

In the AUTOSAR Layered Software Architecture, the Ethernet Driver belongs to the Microcontroller Abstraction Layer, or more precisely, to the Communication Drivers.

This indicates the main task of the Ethernet Driver:

Provide to the upper layer (Ethernet Interface) a hardware independent interface comprising multiple equal controllers. This interface shall be uniform for all controllers. Thus, the upper layer (Ethernet Interface) may access the underlying bus system in a uniform manner. The interface provides functionality for initialization, configuration and data transmission. The configuration of the Ethernet Driver however is bus specific, since it takes into account the specific features of the communication controller.

A single Ethernet Driver module supports only one type of controller hardware, but several controllers of the same type. Additionally, the Ethernet Driver has to be able to be interoperable with the Switch Driver, if it is in a managed mode. In this case, a special treatment of the Ethernet frame might be necessary to fit a specific interpretation by a Switch device afterwards. The Ethernet Driver's prefix requires a unique namespace. The Ethernet Interface can access different controller types using different Ethernet Drivers using this prefix. The decision which driver to use to access a particular controller is a configuration parameter of the Ethernet Interface.

Figure 1.1 depicts the lower part of the Ethernet stack. One Ethernet Interface accesses several controllers using one or several Ethernet Drivers.

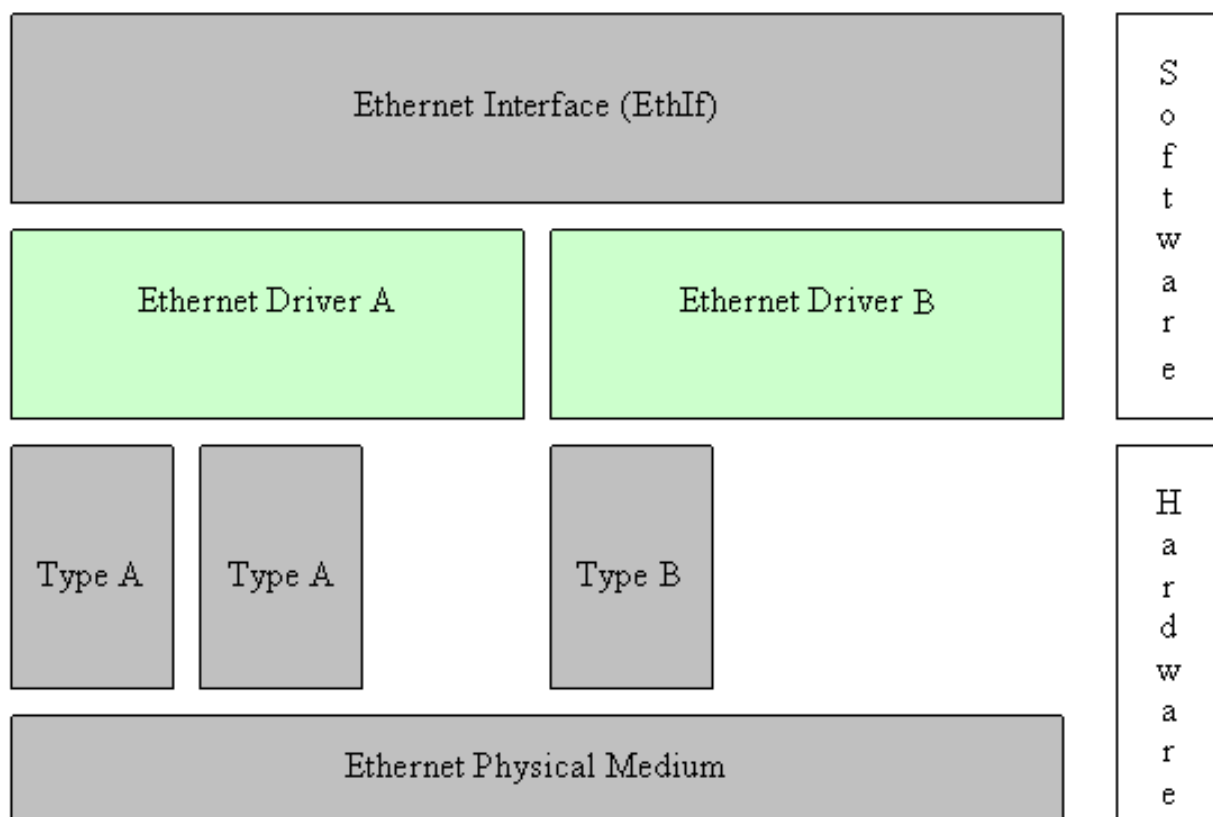


Figure 1.1: Ethernet stack module overview

Note: The Ethernet Driver is specified in a way that allows for object code delivery of the code module, following the "one-fits-all" principle, i.e. the entire configuration of the Ethernet Interface can be carried out without modifying any source code. Thus, the configuration of the Ethernet Driver can be carried out largely without detailed knowledge of the Ethernet Driver software.

2 Acronyms, Abbreviations and Definition

2.1 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the Ethernet Driver module that are not included in the *AUTOSAR glossary* [1].

Abbreviation / Acronym:	Description:
EC	Ethernet controller
Eth	Ethernet Driver (AUTOSAR BSW module)
EthIf	Ethernet Interface (AUTOSAR BSW module)
EthTrcv	Ethernet Transceiver Driver (AUTOSAR BSW module)
ISR	Interrupt Service Routine
MACPHY	Ethernet controller and PHY integrated in one module
MCG	Module Configuration Generator
MII	Media Independent Interface (standardized Interface provided by Ethernet controllers to access Ethernet transceivers)
OA TC06	OPEN ALLIANCE Technical Committee 6 "10BASE-T1x MACPHY Serial interface"
OA TC10 [2]	OPEN ALLIANCE Technical Committee 10 "Automotive Ethernet Sleep/Wake-Up"
PLCA	Physical Layer Collision Avoidance - Media acces
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
MDIO	Managment Data Input Ouput
MMD	MDIO Manageable Device

Table 2.1: Acronyms and Abbreviations

2.2 Definitions

2.2.1 Hardware supported data transfert

A "Hardware supported data transfer" represents a copy action where data is transferred from a source address to an destination address asynchronously by hardware (e.g. DMA).

2.2.2 Data transfer session handle

A "Data transfer session handle" represents an id to identify a specific hardware supported data transfer. This id could be used by hardware to confirm the finalization of the data transfer.

3 Related documentation

3.1 Input documents

- [1] Glossary
AUTOSAR_FO_TR_Glossary
- [2] OPEN Sleep/Wake-up Specification for Automotive Ethernet
<http://www.opensig.org/Automotive-Ethernet-Specifications/>
- [3] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral
- [4] Specification of Ethernet Interface
AUTOSAR_CP_SWS_EthernetInterface
- [5] Specification of Ethernet Transceiver Driver
AUTOSAR_CP_SWS_EthernetTransceiverDriver
- [6] Specification of Ethernet Switch Driver
AUTOSAR_CP_SWS_EthernetSwitchDriver
- [7] General Requirements on SPAL
AUTOSAR_CP_RS_SPALGeneral
- [8] Specification of ECU State Manager
AUTOSAR_CP_SWS_ECUSTateManager
- [9] Requirements on Ethernet Support in AUTOSAR
AUTOSAR_CP_RS_Ethernet
- [10] IEEE 802.3cg-2019
<https://www.ieee802.org/3/>
- [11] OPEN ALLIANCE 10BASE-T1S MACPHY Serial interface (Sep 2020)
<http://www.opensig.org/Automotive-Ethernet-Specifications/>
- [12] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer
- [13] IEEE 802.1Q-2022 - IEEE Standard for Local and Metropolitan Area Network - Bridges and Bridged Networks
<https://ieeexplore.ieee.org/>
- [14] System Template
AUTOSAR_CP_TPS_SystemTemplate
- [15] Explanation of Time Sensitive Network features
AUTOSAR_FO_EXP_TimeSensitiveNetworkFeatures
- [16] IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security

<https://ieeexplore.ieee.org/document/8585421>

[17] IEEE 802.3-2015

<https://www.ieee802.org/3/>

[18] STD 59 RFC 2819

<https://www.rfc-editor.org/info/rfc2819>

3.2 Related standards and norms

No content.

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules *SWS BSW General*, [3], which is also valid for Ethernet Driver.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Ethernet Driver.

Additional support added for clause 45 MII access defined by IEEE 802.3ae.

4 Constraints and assumptions

4.1 Limitations

It is not possible to transmit data which exceeds the available buffer size of the used controller. Longer data has to be transmitted using the Internet Protocol (IP) or Transmission Control Protocol (TCP).

Depending on the Ethernet hardware, it may become necessary that implementations deviate from API specifications in respect to the asynchronous/synchronous behaviour.

4.2 Applicability to car domains

The Ethernet BSW stack is intended to be used wherever high data rates are required but no hard real-time is required. Of course, it can also be used for less-demanding use cases, i.e. for low data rates.

5 Dependencies to other modules

This chapter lists the modules interacting with the Ethernet Driver module.

Modules that use Ethernet Driver module (Calls between drivers are abstracted by the EthIf):

- Ethernet Interface (EthIf, see [4])
- Ethernet Transceiver Driver (EthTrcv see [5])
- Ethernet Switch Driver (EthSwt, see [6])

Modules used by the Ethernet Driver module:

- BSW Scheduler mechanisms for data consistency and main function handling.

Dependencies to other Modules:

- On certain systems the controller might share resources with other components (e.g. the MCU, Port), and may depend on their configuration. If those resources are within scope of the other modules (e.g. PLL configuration, memory mapping, etc.) the Ethernet Driver module does not take care of configuring those components but requires their preceding initialization.

5.1 Driver Services

[SWS_Eth_00282]

Status: DRAFT

Upstream requirements: [SRS_BSW_00005](#)

[If the Ethernet controller is on-chip, the Eth module shall not use any service of other drivers.]

Note: Not in case of MACPHY

[SWS_Eth_00283]

Status: DRAFT

Upstream requirements: [SRS_BSW_00377](#)

[The function Eth_Init shall initialize all on-chip hardware resources that are used by the Ethernet controller. The only exception to this is the digital I/O pin configuration (of pins used by Ethernet controller), which is done by the port driver.]

[SWS_Eth_00284]

Status: DRAFT

Upstream requirements: [SRS_BSW_00005](#)

[The Mcu module (SPAL see *SPAL General*[7]) shall configure register settings that are "shared" with other modules.]

Implementation hint: The Mcu module shall be initialized before initializing the Ethernet module.

[SWS_Eth_00285]

Status: DRAFT

Upstream requirements: [SRS_BSW_00005](#)

[If an off-chip Ethernet controller is used (i.e. MACPHY), the Ethernet controller module shall use services of other MCAL drivers (e.g. SPI).]

Implementation hint: If the Ethernet driver module uses services of other MCAL drivers (e.g. SPI), it must be ensured that these drivers are up and running before initializing the Ethernet module. The sequence of initialization of different drivers is partly specified in *SWS ECUStateManager* [8].

6 Requirements Tracing

The following tables reference the requirements specified in [9] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[FO_RS_MACsec_00004]	Configure which Ethernet ports use MACsec	[SWS_Eth_00418] [SWS_Eth_00419] [SWS_Eth_00420] [SWS_Eth_00421] [SWS_Eth_00422] [SWS_Eth_00423] [SWS_Eth_00424] [SWS_Eth_00425] [SWS_Eth_00426] [SWS_Eth_00427] [SWS_Eth_00428] [SWS_Eth_00429] [SWS_Eth_00430] [SWS_Eth_00431] [SWS_Eth_00432] [SWS_Eth_00433] [SWS_Eth_00434] [SWS_Eth_00435] [SWS_Eth_00436] [SWS_Eth_00437] [SWS_Eth_00438] [SWS_Eth_00439] [SWS_Eth_00440] [SWS_Eth_00441] [SWS_Eth_00442] [SWS_Eth_00443] [SWS_Eth_00444] [SWS_Eth_00445] [SWS_Eth_00446] [SWS_Eth_00447] [SWS_Eth_00448] [SWS_Eth_CONSTR_00023]
[SRS_BSW_00005]	Modules of the μ C Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	[SWS_Eth_00282] [SWS_Eth_00284] [SWS_Eth_00285]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_Eth_00364]
[SRS_BSW_00159]	All modules of the AUTOSAR Basic Software shall support a tool based configuration	[SWS_Eth_00296]
[SRS_BSW_00171]	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	[SWS_Eth_00349] [SWS_Eth_00355] [SWS_Eth_00363] [SWS_Eth_00368] [SWS_Eth_00372]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/disabling of detection and reporting of development errors.	[SWS_Eth_00313] [SWS_Eth_00314] [SWS_Eth_00315] [SWS_Eth_00316] [SWS_Eth_00317] [SWS_Eth_00318] [SWS_Eth_00319] [SWS_Eth_00320] [SWS_Eth_00321] [SWS_Eth_00322] [SWS_Eth_00323] [SWS_Eth_00324] [SWS_Eth_00325] [SWS_Eth_00327] [SWS_Eth_00328] [SWS_Eth_00329] [SWS_Eth_00331] [SWS_Eth_00332] [SWS_Eth_00333] [SWS_Eth_00334] [SWS_Eth_00335] [SWS_Eth_00336] [SWS_Eth_00393] [SWS_Eth_00413] [SWS_Eth_00415] [SWS_Eth_00416] [SWS_Eth_00417] [SWS_Eth_CONSTR_00005] [SWS_Eth_CONSTR_00006] [SWS_Eth_CONSTR_00007] [SWS_Eth_CONSTR_00008] [SWS_Eth_CONSTR_00018] [SWS_Eth_CONSTR_00019] [SWS_Eth_CONSTR_00020]
[SRS_BSW_00377]	A Basic Software Module can return a module specific types	[SWS_Eth_00283]





Requirement	Description	Satisfied by
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_Eth_00313] [SWS_Eth_00314] [SWS_Eth_00315] [SWS_Eth_00316] [SWS_Eth_00317] [SWS_Eth_00318] [SWS_Eth_00319] [SWS_Eth_00320] [SWS_Eth_00321] [SWS_Eth_00322] [SWS_Eth_00323] [SWS_Eth_00324] [SWS_Eth_00325] [SWS_Eth_00327] [SWS_Eth_00328] [SWS_Eth_00329] [SWS_Eth_00331] [SWS_Eth_00332] [SWS_Eth_00333] [SWS_Eth_00334] [SWS_Eth_00335] [SWS_Eth_00336] [SWS_Eth_00393] [SWS_Eth_00413] [SWS_Eth_00415] [SWS_Eth_00416] [SWS_Eth_00417] [SWS_Eth_CONSTR_00005] [SWS_Eth_CONSTR_00006] [SWS_Eth_CONSTR_00007] [SWS_Eth_CONSTR_00008] [SWS_Eth_CONSTR_00018] [SWS_Eth_CONSTR_00019] [SWS_Eth_CONSTR_00020]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_Eth_00350]
[SRS_BSW_00450]	A Main function of a un-initialized module shall return immediately	[SWS_Eth_00393]
[SRS_BSW_00458]	Classification of production errors	[SWS_Eth_00173] [SWS_Eth_00219] [SWS_Eth_00220] [SWS_Eth_00221] [SWS_Eth_00222] [SWS_Eth_00223] [SWS_Eth_00224] [SWS_Eth_00225]
[SRS_BSW_00459]	It shall be possible to concurrently execute a service offered by a BSW module in different partitions	[SWS_Eth_00351] [SWS_Eth_00357] [SWS_Eth_00365] [SWS_Eth_00387]
[SRS_Eth_00053]	SWS shall specify configuration	[SWS_Eth_00251] [SWS_Eth_00255]
[SRS_Eth_00072]	The Ethernet Interface shall provide VLAN support	[SWS_Eth_91001]
[SRS_Eth_00120]	Hardware access via MII and/or SPI	[SWS_Eth_91012] [SWS_Eth_91013]
[SRS_Eth_00121]	Configuration of forwarding rules	[SWS_Eth_00408] [SWS_Eth_91001]
[SRS_Eth_00127]	The Ethernet Driver shall provide statistic counter values	[SWS_Eth_00026] [SWS_Eth_00226] [SWS_Eth_00233] [SWS_Eth_91002] [SWS_Eth_91003] [SWS_Eth_91004] [SWS_Eth_91005] [SWS_Eth_91006]
[SRS_Eth_00146]	The Ethernet Driver shall provide 10 BASE-T1S support	[SWS_Eth_00279] [SWS_Eth_00287] [SWS_Eth_00289] [SWS_Eth_00290] [SWS_Eth_00295] [SWS_Eth_00390] [SWS_Eth_00391]
[SRS_Eth_00147]	The Ethernet Driver shall support SPI	[SWS_Eth_00287] [SWS_Eth_00290] [SWS_Eth_00295] [SWS_Eth_00390] [SWS_Eth_00391] [SWS_Eth_91012] [SWS_Eth_91013]
[SRS_Eth_00148]	The Ethernet Driver shall support MII	[SWS_Eth_00273] [SWS_Eth_00274] [SWS_Eth_00278] [SWS_Eth_00279] [SWS_Eth_00289] [SWS_Eth_00290] [SWS_Eth_00390] [SWS_Eth_00391]
[SRS_Eth_00167]	PTP Physical Clock Adjustment	[SWS_Eth_00339] [SWS_Eth_00340] [SWS_Eth_00341] [SWS_Eth_00373] [SWS_Eth_00374] [SWS_Eth_00375] [SWS_Eth_91018] [SWS_Eth_91019] [SWS_Eth_CONSTR_00010] [SWS_Eth_CONSTR_00011]





Requirement	Description	Satisfied by
[SRS_Eth_00168]	Pulse Per Second Signal Configuration	[SWS_Eth_00342] [SWS_Eth_00343] [SWS_Eth_00344] [SWS_Eth_00376] [SWS_Eth_00377] [SWS_Eth_00378] [SWS_Eth_00379] [SWS_Eth_CONSTR_00012]
[SRS_Eth_00171]	Ethernet Driver ingress and egress queues	[SWS_Eth_00325] [SWS_Eth_00331] [SWS_Eth_00332] [SWS_Eth_00333] [SWS_Eth_00334] [SWS_Eth_00335] [SWS_Eth_00336] [SWS_Eth_00415] [SWS_Eth_00416] [SWS_Eth_00417] [SWS_Eth_CONSTR_00005] [SWS_Eth_CONSTR_00006] [SWS_Eth_CONSTR_00007] [SWS_Eth_CONSTR_00008] [SWS_Eth_CONSTR_00018] [SWS_Eth_CONSTR_00019] [SWS_Eth_CONSTR_00020]
[SRS_Eth_00172]	Ethernet Driver hardware supported data transfer	[SWS_Eth_00317] [SWS_Eth_00318] [SWS_Eth_00319] [SWS_Eth_00320] [SWS_Eth_91023]
[SRS_Eth_00173]	Ethernet Driver transmission requests with direct data provision	[SWS_Eth_00313] [SWS_Eth_00314] [SWS_Eth_00315] [SWS_Eth_00316] [SWS_Eth_00317] [SWS_Eth_00318] [SWS_Eth_00321] [SWS_Eth_00322] [SWS_Eth_00323] [SWS_Eth_00324] [SWS_Eth_00327] [SWS_Eth_00328] [SWS_Eth_00329] [SWS_Eth_91022]
[SRS_Eth_00174]	Ethernet Driver ingress queue handling	[SWS_Eth_91024]
[SRS_Eth_00175]	The Ethernet Interface shall support access to PTP Physical Clocks	[SWS_Eth_91017] [SWS_Eth_91020]
[SRS_Eth_00176]	The Ethernet Interface shall support control of pulse per second signal generation	[SWS_Eth_91021]
[SRS_Eth_00177]	Ethernet Driver Transmission Selection Algorithm	[SWS_Eth_00401] [SWS_Eth_00402] [SWS_Eth_00403] [SWS_Eth_00404] [SWS_Eth_00405] [SWS_Eth_00406] [SWS_Eth_00407] [SWS_Eth_00409] [SWS_Eth_00410] [SWS_Eth_00411] [SWS_Eth_00412] [SWS_Eth_CONSTR_00013] [SWS_Eth_CONSTR_00014] [SWS_Eth_CONSTR_00015] [SWS_Eth_CONSTR_00016] [SWS_Eth_CONSTR_00017]
[SRS_Eth_00179]	Ethernet Switch Transmission Selection Algorithm	[SWS_Eth_00408]
[SRS_Eth_00180]	Ethernet Switch port scheduling of egress queues	[SWS_Eth_00408]
[SRS_Eth_00184]	Ethernet Driver scheduler algorithm	[SWS_Eth_00401] [SWS_Eth_00402] [SWS_Eth_00403] [SWS_Eth_00404] [SWS_Eth_00405] [SWS_Eth_00406] [SWS_Eth_00407] [SWS_Eth_00409] [SWS_Eth_00410] [SWS_Eth_00411] [SWS_Eth_00412] [SWS_Eth_CONSTR_00013] [SWS_Eth_CONSTR_00014] [SWS_Eth_CONSTR_00015] [SWS_Eth_CONSTR_00016] [SWS_Eth_CONSTR_00017]





Requirement	Description	Satisfied by
[SRS_Eth_00185]	Ethernet Driver Frame Preemption	[SWS_Eth_CONSTR_00021] [SWS_Eth_CONSTR_00022]
[SRS_Eth_00188]	Ethernet Driver transmission requests with indirect data provision	[SWS_Eth_00413] [SWS_Eth_00414]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Ethernet BSW stack

As part of the AUTOSAR Layered Software Architecture according to Figure 7.1, the Ethernet BSW modules also form a layered software stack. Figure 7.1 depicts the basic structure of this Ethernet BSW stack. The Ethernet Interface module accesses several controllers using the Ethernet Driver layer, which can be made up of several Ethernet Drivers modules.

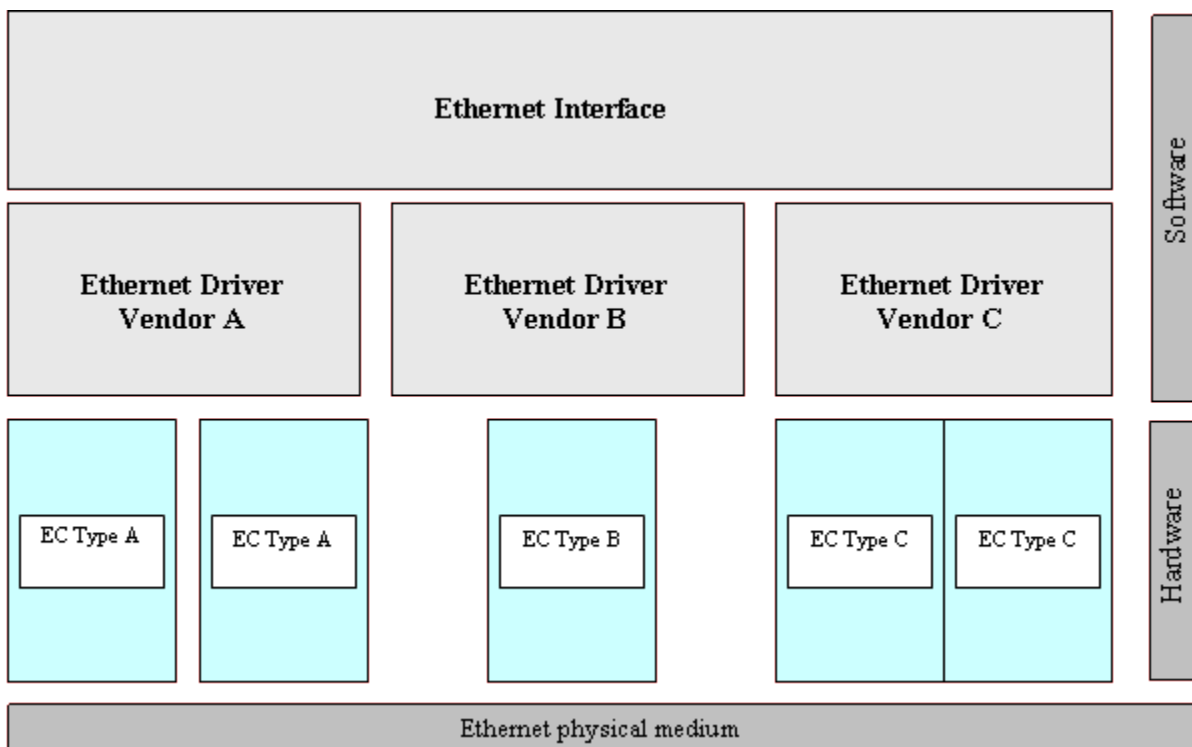


Figure 7.1: Basic Structure of the Ethernet BSW stack

7.1.1 Switch

Furthermore a Switch device might be connected to a dedicated controller index of an Ethernet Driver. This scenario leads to additional interaction between the Switch Driver and the Ethernet Driver using the driver API abstraction of the EthIf (Figure 7.2). The Ethernet Driver asks the Switch Driver for a special treatment to ensure that the current Ethernet frame could be managed in the Switch later on.

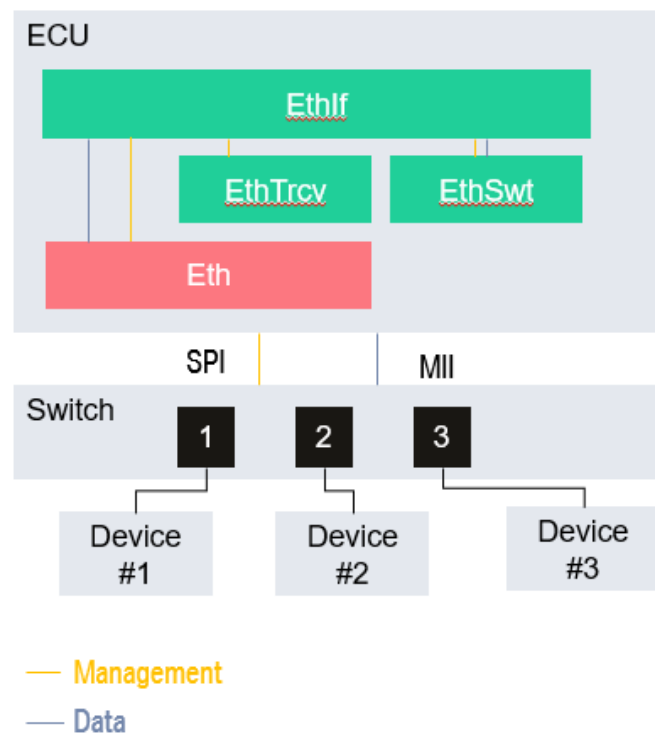


Figure 7.2: HW/SW basic structure including Switch device

7.1.2 External MAC

In case of MACPHY (external mac controller) the data and management are done via the SPI module (see [10] and [11]) (Figure 7.3).

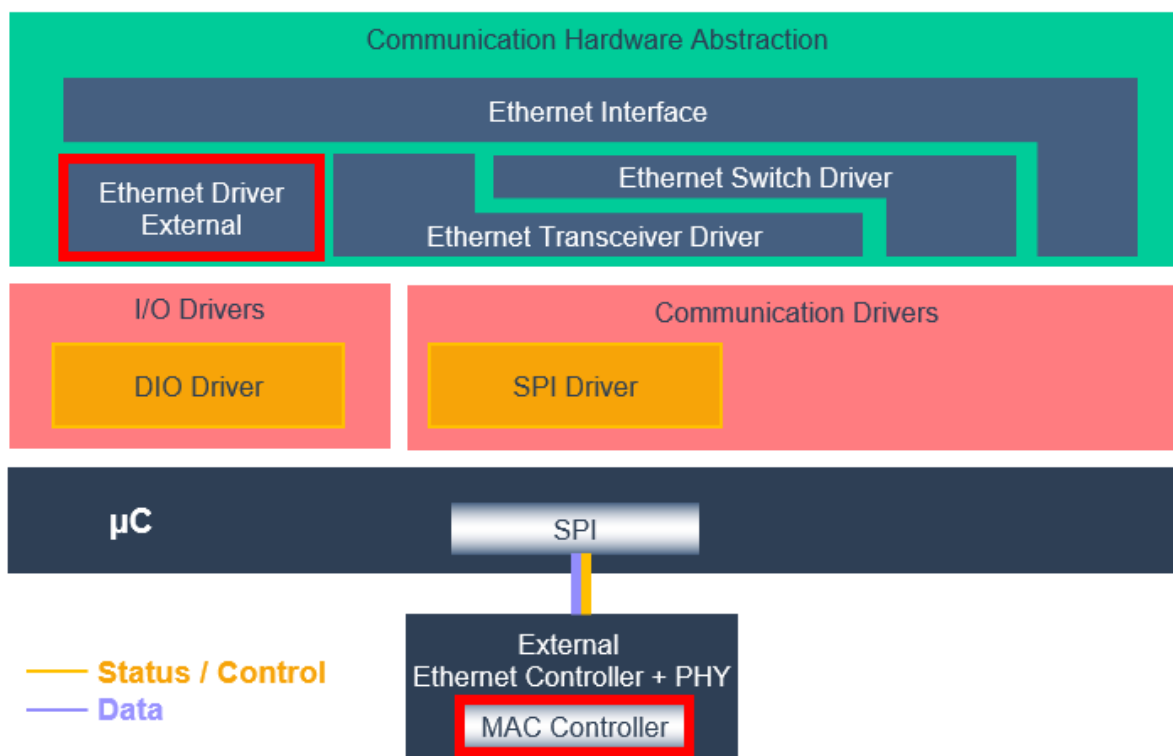


Figure 7.3: External MAC Controller

7.1.3 Indexing scheme

Users of the Ethernet Driver identify controller resources using an indexing scheme as depicted in Figure 7.4.

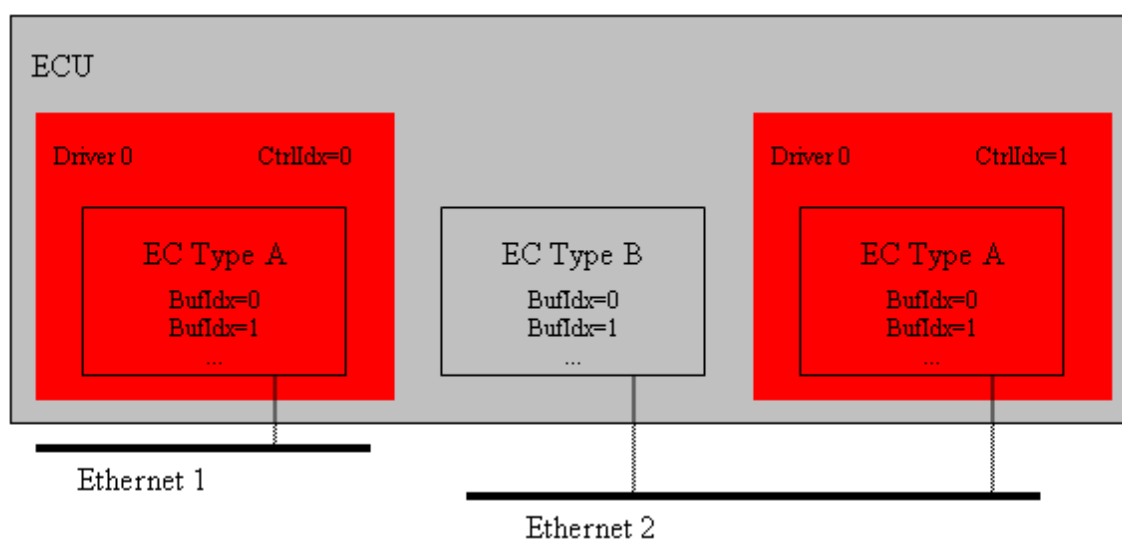


Figure 7.4: Ethernet Driver indexing scheme

[SWS_Eth_00003] [The Ethernet Driver is using a zero-based index to abstract the access for upper software layers. The parameter EthCtrlIdx [\[ECUC_Eth_00007\]](#) within configuration corresponds to parameter CtrlIdx used in the API.]

Note: A buffer index (BufIdx) is an internal handle for one transaction between EthIf and Eth. BufIdx identifies an Ethernet buffer processed by Ethernet Driver API functions. Each controller's buffers are identified by buffer indexes 0 to (n-1) where n is the number of buffers processed by the corresponding controller. Buffer indexes are valid within a tuple <CtrlIdx, BufIdx> only. A BufIdx uniquely identifies the buffer used for an Ethernet Driver.

7.1.4 Requirements

This chapter lists requirements that shall be fulfilled by Ethernet Driver module implementations.

The Ethernet Driver module environment comprises all modules which are calling interfaces of the Ethernet Driver module.

[SWS_Eth_00005] [The Ethernet Driver module shall support pre-compile time, link time and post-build time configuration.]

[SWS_Eth_00008] [In case development error detection is enabled for the Ethernet Driver module: The Ethernet Driver module shall check API parameters for validity and report detected errors to the DET.]

DET API functions are specified in *SWS Default Error Tracer* [\[12\]](#).

[SWS_Eth_00011] [None of the Ethernet Driver module header files shall define global variables.]

[SWS_Eth_00218] [The Ethernet Driver shall ensure that the base addresses of all reception and transmission buffers fulfill the memory alignment requirements for all AUTOSAR data types of the respective platform.]

[SWS_Eth_00216] [For transmissions the Ethernet Controller shall enable hardware capabilities for the calculation of protocol checksums (offloading) according to the following list:

- a) for IPv4 frames if EthCtrlEnableOffloadChecksumIPv4 is set to TRUE
- b) for ICMP frames if EthCtrlEnableOffloadChecksumICMP is set to TRUE
- c) for TCP frames if EthCtrlEnableOffloadChecksumTCP is set to TRUE
- d) for UDP frames if EthCtrlEnableOffloadChecksumUDP is set to TRUE.

In all other cases, the Ethernet Controller shall not manipulate the checksum fields.]

[SWS_Eth_00217] [For reception the Ethernet Controller shall enable hardware capabilities to discard frames with mismatching protocol checksums (offloading) according to the following list:

- a) for IPv4 frames if `EthCtrlEnableOffloadChecksumIPv4` is set to `TRUE`
- b) for ICMP frames if `EthCtrlEnableOffloadChecksumICMP` is set to `TRUE`
- c) for TCP frames if `EthCtrlEnableOffloadChecksumTCP` is set to `TRUE`
- d) for UDP frames if `EthCtrlEnableOffloadChecksumUDP` is set to `TRUE`.

In all other cases, the Ethernet Controller shall not consider the protocol checksum fields.]

[SWS_Eth_00247] [The Switch Driver management API's:

- `EthSwt_EthRxProcessFrame()`,
- `EthSwt_EthRxFinishedIndication()`,
- `EthSwt_EthTxPrepareFrame()`,
- `EthSwt_EthTxAdaptBufferLength()`,
- `EthSwt_EthTxProcessFrame()` and
- `EthSwt_EthTxFinishedIndication()`

shall be used using the driver API abstraction of the `EthIf` to inform the Switch Driver about a required special treatment for Switch management purpose (see document *AUTOSAR_SWS_EthernetInterface* [4]).]

7.1.5 Initialization

The Eth driver module is initialized via `Eth_Init()`, and de-initialized via `Eth_DeInit()`. Except for `Eth_GetVersionInfo`, `Eth_Init()` or any scheduled function (e.g. `Eth_MainFunction()`) the API functions of the Eth driver module may only be called after the module has been properly initialized.

[SWS_Eth_00393] Eth Initialization

Upstream requirements: [SRS_BSW_00350](#), [SRS_BSW_00386](#), [SRS_BSW_00450](#)

[If development error reporting is enabled via `EthDevErrorDetect`, the Eth driver module shall call `Det_ReportError` with the error code `Eth_E_UNINIT` when any API other than `Eth_Init()`, `Eth_GetVersionInfo` or any scheduled function (e.g. `Eth_MainFunction()`) is called in uninitialized state.]

7.1.6 Communication

7.1.6.1 Transmission

The Ethernet driver provides two approaches to handle transmission requests.

7.1.6.1.1 Indirect data provision

Transmission request with indirect data provision: splits the request for available egress queue resources and the transmission request in two API calls. The upper layer has to request for an available egress queue element of the corresponding `EthCtrlConfigEgressQueue` at the corresponding Ethernet controller. If the Ethernet driver is able to provide an egress queue element, then the requester (upper layer) can update this egress queue element with data. A second call from the upper layer would request to transmit the egress queue element:

1. An upper layer call `Eth_ProvideTxBuffer` to request an egress buffer at the Ethernet driver according the given priority. After return, the upper layer copies data to the provided egress buffer
2. An upper layer call `Eth_Transmit` to request the Ethernet driver to transmit the content of the egress buffer

[SWS_Eth_00413] Precondition checks for transmission request with indirect data provision

Status: DRAFT

Upstream requirements: [SRS_Eth_00188](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If `Eth_ProvideTxBuffer` has been called and the given `CtrlIdx` has an `EthCtrlConfigEgressQueue` configured, then the Ethernet driver shall perform the following precondition checks in the following order, otherwise return with `E_NOT_OK`:

1. If the given `Priority` matches the configured `EthCtrlConfigEgressQueueSortingType` of an `EthCtrlConfigEgressQueue` at the given `CtrlIdx`, then proceed. Otherwise report a runtime error `ETH_E_UNKNOWN_EGRESS_PRIORITY` and proceed.
2. If the Ethernet frame could be enqueued in a `EthCtrlConfigEgressQueue` at the given `CtrlIdx` where no `EthCtrlConfigEgressQueueSortingType` is configured (i.e. try to enqueue the Ethernet frame in a default `EthCtrlConfigEgressQueue` (see [\[SWS_Eth_CONSTR_00020\]](#))), then proceed. Otherwise return with `E_NOT_OK`.
3. If an element of the identified `EthCtrlConfigEgressQueue` is available, then proceed. Otherwise report a runtime error `ETH_E_EGRESS_QUEUE_OCCUPIED` and return with `E_NOT_OK`.

If all precondition checks passed successfully, then proceed with evaluation of the Ethernet frame.]

Specification for transmission can be found in [Chapter 8.4.23](#) and [Chapter 8.4.22](#)

7.1.6.1.2 Direct data provision

Transmission request with direct data provision: Performs the data and transmission request in one API call. The upper layer call `Eth_ImmediateTransmit` provides a list of headers as single linked list and the payload with payload length. All headers of the single linked list together with the payload form an entire Ethernet frame. Each element of the list contains a pointer to data, data length and a pointer to the next element. The Ethernet driver has to traverse from the head to the last element (tail) and copy data of each header to an egress queue element. After the last element has been reached, the payload is added to the egress queue element. If the data transfer is finished, the entire Ethernet frame resides in the egress queue element. The Ethernet driver triggers a transmission of the Ethernet frame to convey the data on the Ethernet network.

[SWS_Eth_00313] Precondition checks for transmission request with direct data provision

Status: DRAFT

Upstream requirements: [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If `Eth_ImmediateTransmit` has been called and the given `CtrlIdx` has an `EthCtrlConfigEgressQueue` configured, then the Ethernet driver shall perform the following precondition checks in the following order:

1. If the Ethernet frame, which is requested to be transmitted, matches the configured `EthCtrlConfigEgressQueueSortingType` of an `EthCtrlConfigEgressQueue` at the given `CtrlIdx`, then proceed. Otherwise report runtime error `ETH_E_NO_MATCHING_EGRESS_QUEUE_IDENTIFIED` and proceed.
2. If the Ethernet frame could be enqueued in a `EthCtrlConfigEgressQueue` at the given `CtrlIdx` where no `EthCtrlConfigEgressQueueSortingType` is configured (i.e. try to enqueue the Ethernet frame in a default `EthCtrlConfigEgressQueue` (see [\[SWS_Eth_CONSTR_00020\]](#))), then proceed. Otherwise return with `E_NOT_OK`.
3. If an element of the identified `EthCtrlConfigEgressQueue` is available, then proceed. Otherwise report an runtime error error code `ETH_E_EGRESS_QUEUE_OCCUPIED` and return with `E_NOT_OK`.

If all precondition checks passed successfully, then proceed with evaluation of the Ethernet frame.]

[SWS_Eth_00314]

Status: DRAFT

Upstream requirements: [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If `Eth_ImmediateTransmit` has been called, an element in the `EthCtrlConfigEgressQueue` is reserved and the Ethernet driver is requested to evaluate the

given Ethernet frame parts (according to [SWS_Eth_00313]), then the Ethernet driver shall evaluate the given single linked list given with `HeaderListPtr` and the payload `PayloadPtr` and payload length `PayloadLength` by considering the following steps:

1. Traverse the single linked list given with `HeaderListPtr` by starting with the first element `HeaderListPtr` and continue with next element of the single linked list given with `NextListElemPtr` until an element of the single linked list is reached where `NextListElemPtr` is set to `NUL_PTR`. Perform the following action at each element of the single linked list:
 - Store the the given data location (`DataPtr`) and the given data length (`DataLength`)
 - accumulate the `DataLength`)
2. calculate the overall length by considering accumulated `DataLength` of all single linked list elements and the length of payload given with `PayloadLength`

If the calculated Ethernet frame length is larger then the available egress queue element, then abort the evaluation and return with `E_NOT_OK`, or if `EthDevErrorDetect` is set to `TRUE`, Eth driver shall call `Det_ReportError` with the error code `ETH_EXCEED_EGRESS_QUEUE_ELEMENT`. Otherwise proceed with construction of the Ethernet frame.]

[SWS_Eth_00315]

Status: DRAFT

Upstream requirements: [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If `EthImmediateTransmit` has been called, an element in the `EthCtrlConfigEgressQueue` is reserved, the Ethernet driver is requested to construct the Ethernet frame (according to [SWS_Eth_00314]) and `EthCtrlEnableEgressHardwareSupportedDataTransfer` is set to `FALSE`, then the Ethernet driver shall consider the following construction steps:

- iterate over the stored list of header pointers (see [SWS_Eth_00314]) and perform for each header the following step:
 - Copy data from the given data location (`DataPtr`) with respect to the given data length (`DataLength`) to the next available position in `EthCtrlConfigEgressQueue` element in consecutive order without gaps and continue
- copy payload data from the given location `PayloadPtr` with respect to the given length (`PayloadLength`) to the next available position in `EthCtrlConfigEgressQueue` element in consecutive order without gaps
- trigger a transmission for content of this `EthCtrlConfigEgressQueue` element
- store the given `TxHandleId` with the used `EthCtrlConfigEgressQueue` element and the given `CtrlIdx`

]

[SWS_Eth_00316]*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If [Eth_ImmediateTransmit](#) has been called, an element in the [EthCtrlConfigEgressQueue](#) is reserved, the Ethernet driver is requested to construct the Ethernet frame (according to [\[SWS_Eth_00314\]](#)) and [EthCtrlEnableEgressHardwareSupportedDataTransfer](#) is set to `TRUE`, then the Ethernet driver shall consider the following construction steps:

- iterate over the stored list of header pointers (see [\[SWS_Eth_00314\]](#)) and perform for each header to the following steps:
 - if the given header length (`DataLength`) of a list element exceeds the configured [EthCtrlEgressHardwareSupportedDataTransferThreshold](#), then the Ethernet driver shall prepare a hardware supported transfer with respect to the given header length (`DataLength`) and header location (`DataPtr`), trigger the data transfer and reserve space according the given `DataLength` in the [EthCtrlConfigEgressQueue](#) element, store the data transfer session handle (by considering given `TxHandleId`, `CtrlIdx` and [EthCtrlConfigEgressQueue](#) element) and continue at next available position + `DataLength` + 1 of the [EthCtrlConfigEgressQueue](#) element
 - if the given length (`DataLength`) is equal or smaller than the configured [EthCtrlEgressHardwareSupportedDataTransferThreshold](#), then the Ethernet driver shall copy data from the given header location (`DataPtr`) with respect to the given header length (`DataLength`) to the next available position in [EthCtrlConfigEgressQueue](#) element in consecutive order and continue
- check the payload length given with (`PayloadLength`)
 - if the given payload length (`PayloadLength`) of a list element exceeds the configured [EthCtrlEgressHardwareSupportedDataTransferThreshold](#), then the Ethernet driver shall prepare a hardware supported transfer with respect to the given payload length (`PayloadLength`) and payload location (`PayloadPtr`), trigger the data transfer and reserve space according the given `PayloadLength` in the [EthCtrlConfigEgressQueue](#) element, store the data transfer session handle (by considering given `TxHandleId`, `CtrlIdx` and [EthCtrlConfigEgressQueue](#) element)
 - if the given payload length (`PayloadLength`) is equal or smaller than the configured [EthCtrlEgressHardwareSupportedDataTransferThreshold](#), then the Ethernet driver shall copy the payload from the given payload location (`PayloadPtr`) with respect to the given payload length (`PayloadLength`) to the next available position in [EthCtrlConfigEgressQueue](#) element in consecutive order

- store the given `TxHandleId` with the used `EthCtrlConfigEgressQueue` element and the given `CtrlIdx`

]

Note: The mapping of `TxHandleId` with the used `EthCtrlConfigEgressQueue` element and the given `CtrlIdx` are used to identify the provided `TxHandleId`, which is needed if confirmation of the transmission has to be indicated via `Eth_TxTransmission`.

All sessions for hardware supported data transfer which relate to the same `EthCtrlConfigEgressQueue` element need to be confirmed by hardware. Therefore the Ethernet driver needs to supervise the state of triggered hardware supported data transfer in relation to the affected `TxHandleId`, `CtrlIdx` and `EthCtrlConfigEgressQueue` element. After all data transfers which relate to the same `EthCtrlConfigEgressQueue` element have been finalized, the transmission for this `EthCtrlConfigEgressQueue` element can be triggered.

[SWS_Eth_00317]

Status: DRAFT

Upstream requirements: [SRS_Eth_00172](#), [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If `Eth_ImmediateTransmit` has been called, `EthCtrlEnableEgressHardwareSupportedDataTransfer` is set to `TRUE` and all data transfer sessions have confirmed successful transfer for a specific `EthCtrlConfigEgressQueue` element, then the Ethernet driver shall perform the following actions:

- remove all data transfer session handles which are associated with this `EthCtrlConfigEgressQueue` element
- trigger a transmission of the content of this `EthCtrlConfigEgressQueue` element

]

Please note: Mapping of `EthCtrlConfigEgressQueue` element and the given `CtrlIdx` to `TxHandleId` is needed for asynchronous check in the `EthIf_MainFunctionTx` or within an interrupt.

7.1.6.2 Transmission confirmation

[SWS_Eth_00243] [Ethernet SW Driver shall call `EthIf_TxConfirmation` with `Result` set to `E_OK` to indicate a successful transmission; either from the Interrupt routine (in interrupt mode) or from the `Eth_TxConfirmation` routine in polling mode (if the notification has been enabled).]

[SWS_Eth_00256] [Ethernet SW Driver shall call `EthIf_TxConfirmation` with `Result` set to `E_NOT_OK` if the transmission failed.]

The call to `EthIf_TxConfirmation` with `Result` set to `E_NOT_OK` shall allow the upper layer to implement a simple locking scheme. It can rely on the fact that every time `Eth_Transmit` is called, `EthIf_TxConfirmation` will be called afterwards.

7.1.6.2.1 Indirect data provision

A transmission requests with indirect data provision uses `Eth_ProvideTxBuffer` as first call to reserve an `EthCtrlConfigEgressQueue` element with a specific `Priority` at a dedicated Ethernet controller. The function returns a `BufIdxPtr`. The tuple of Ethernet controller and `BufIdxPtr` is used as unique identification of the `EthCtrlConfigEgressQueue` element. If a transmission of an Ethernet frame was successful, the Ethernet driver calls `EthIf_TxConfirmation` with `BufIdxPtr` and `CtrlIdx` that refers to the `EthCtrlConfigEgressQueue` element.

[SWS_Eth_00318]

Status: DRAFT

Upstream requirements: [SRS_Eth_00172](#), [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If `Eth_ProvideTxBuffer` was called and returned a `BufPtrIdx` for a specific `EthCtrlConfigEgressQueue` element at the given `CtrlIdx` and a subsequent `Eth_Transmit` request for a transmission for this `BufPtrIdx` at the same `CtrlIdx` and with `TxConfirmation` set to `TRUE` is performed, then the Ethernet driver shall call `EthIf_TxConfirmation` with a `BufPtrIdx` which refers to this `EthCtrlConfigEgressQueue` element.]

7.1.6.2.2 Direct data provision

[SWS_Eth_00321]

Status: DRAFT

Upstream requirements: [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If `Eth_ImmediateTransmit` was called and returned with `E_OK`, and the Ethernet driver detected the finalization of the transmission (either successful or not), then the Ethernet driver shall call `Eth_TxConfirmation` with `TxHandleId` provided in the previous call of `Eth_ImmediateTransmit` which refer to the same `EthCtrlConfigEgressQueue` element.]

Note: A call of `Eth_ImmediateTransmit` which return `E_OK` reserved a `EthCtrlConfigEgressQueue` element at the given `CtrlIdx` and map the given `TxHandleId` to this `EthCtrlConfigEgressQueue` element

[SWS_Eth_00322]*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If [Eth_ImmediateTransmit](#) has been called, [EthCtrlEnableEgressHardwareSupportedDataTransfer](#) is set to `TRUE` and the hardware report for at least one data transfer sessions of a specific [EthCtrlConfigEgressQueue](#) element unsuccessful transfer, then the Ethernet driver shall perform the following actions:

- remove all data transfer session handles from this [EthCtrlConfigEgressQueue](#) element
- call [EthIf_TxConfirmation](#) with `BufIdx` set to `TxHandleId` and result set to `E_NOT_OK`

]

7.1.6.3 Reception

An Ethernet controller receives frames in the configured [EthCtrlConfigIngressQueue](#). The arrival of an Ethernet frame at an [EthCtrlConfigIngressQueue](#) could signal a receive interrupt if interrupt mode is configured for the Ethernet controller or individually for this [EthCtrlConfigIngressQueue](#) (see [Chapter 7.1.7](#) for more details). Otherwise the [EthCtrlConfigIngressQueues](#) are polled. Independent from the handling, the Ethernet driver will call [EthIf_RxIndication](#) to indicate the reception of Ethernet frame.

[SWS_Eth_00244] [Ethernet SW Driver shall call [EthIf_RxIndication](#) to indicate a successful reception either from the Interrupt routine (in interrupt mode) or from the [Eth_Receive](#) routine in polling mode.]

[SWS_Eth_00153] [When calling the callback function [EthIf_RxIndication](#) broadcast frames shall be indicated to the Ethernet Interface (see [\[4\]](#)).]

[SWS_Eth_00323]*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[When calling the callback function [EthIf_RxIndication](#) and [EthGlobalTimeSupport](#) set to `TRUE`, then the Ethernet driver shall provide the ingress timestamp as tuple of type `TimeTupleType` with API parameter `IngressTimeTuplePtr`.]

[SWS_Eth_00324]*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[When calling the callback function [EthIf_RxIndication](#) and [EthGlobalTimeSupport](#) set to `FALSE`, then the Ethernet driver shall provide the ingress timestamp as

tuple of type `TimeTupleType` with API parameter `IngressTimeTuplePtr`, where the included `TimeStampQualType` is set to `INVALID`.]

[SWS_Eth_00327]

Status: DRAFT

Upstream requirements: [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[When calling the callback function `EthIf_RxIndication`, then the Ethernet driver shall provide an unique id as `RxHandleId` which is associated with the affected `EthCtrlConfigIngressQueue` element and the corresponding `CtrlIdx`.]

[SWS_Eth_00328]

Status: DRAFT

Upstream requirements: [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[When calling the callback function `EthIf_RxIndication`, then the Ethernet driver shall keep the affected `EthCtrlConfigIngressQueue` element locked, until `Eth_ReleaseRxBuffer` is called with `RxHandleId` associated with the affected `EthCtrlConfigIngressQueue` element.]

Note: `Eth_ReleaseRxBuffer` could be called by the upper layer in context of the `EthIf_RxIndication`.

[SWS_Eth_00329]

Status: DRAFT

Upstream requirements: [SRS_Eth_00173](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If `Eth_ReleaseRxBuffer` indicate to release the `EthCtrlConfigIngressQueue` element associated with the given `RxHandleId` and the unique `RxHandleId` is associated with a `EthCtrlConfigIngressQueue` element of the given `CtrlIdx`, then the Ethernet driver shall release the `EthCtrlConfigIngressQueue` element and the association with the unique `RxHandleId`. Otherwise the Ethernet driver shall ignore this call and return, or, if `EthDevErrorDetect` is set to `TRUE`, the Ethernet driver shall call `Det_ReportError` with the error code `ETH_E_RX_HANDLE_ID_NOT_ASSOCIATED`.]

7.1.6.4 Hardware supported data transfer

It is possible to configure a hardware supported data transfer (e.g. DMA), to transfer data from the upper layer to an `EthCtrlConfigEgressQueue` element, if hardware supports this feature. A hardware supported data transfer should preserve CPU load. It is assumed that the preparation for each data transfer increase the load on the CPU. If a preparation wastes the same amount of CPU resource as the data transfer itself (or in worst case wastes more), then the CPU performance is negatively impacted. The usage of hardware supported data transfer has to consider a proper tradeoff between either using CPU or hardware for data transfer. The Ethernet driver supports to configure a data length related threshold to balance between usage of CPU

and hardware supported data transfer. Usage and data length related threshold in bytes can be configured per Ethernet controller with `EthCtrlEnableEgressHardwareSupportedDataTransfer` and `EthCtrlEgressHardwareSupportedDataTransferThreshold`.

Note:

- Hardware supported data transfer could be triggered in context of the Ethernet driver, if `Eth_ImmediateTransmit` is used (direct data provision approach). If using approach for indirect data provision (`Eth_ProvideTxBuffer` in combination with `Eth_Transmit`), a hardware supported data transfer could be triggered in the context of the calling upper layer.
- Hardware supported data transfer for received data could be triggered by destination module. The Ethernet driver support this approach by providing `Eth_ReleaseRxBuffer`. The Ethernet driver keep the `EthCtrlConfigIngressQueue` element locked, until `Eth_ReleaseRxBuffer`. A destination module could trigger hardware supported data transfer and request afterwards to release the `EthCtrlConfigIngressQueue` element.

[SWS_Eth_00319]

Status: DRAFT

Upstream requirements: [SRS_Eth_00172](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If a specific Ethernet controller has `EthCtrlEnableEgressHardwareSupportedDataTransfer` set to `TRUE` and the length of data to be transferred exceeds the configured `EthCtrlEgressHardwareSupportedDataTransferThreshold`, then the Ethernet driver shall prepare and trigger a hardware supported data transfer for this Ethernet controller. Otherwise a CPU driven data transfer shall be performed (e.g. memcpy).]

[SWS_Eth_00320]

Status: DRAFT

Upstream requirements: [SRS_Eth_00172](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If a specific Ethernet controller has triggered a hardware supported data transfer and the according hardware reject the hardware supported data transfer, then the Ethernet driver shall release all resources needed for this data transfer and if `EthDevErrorDetect` is set to `TRUE`, then the Ethernet driver shall call `Det_ReportError` with error code `ETH_E_HW_SUPPORTED_DATA_TRANSFER_REJECTED`.]

7.1.7 Queue handling

The Ethernet driver provide the possibility to configure queues for transmission (`EthCtrlConfigIngressQueue`) and for reception (`EthCtrlConfigEgressQueue`) of Ethernet frames. A single Ethernet controller is represented as `EthCtrlConfig`. An `EthCtrlConfig` could have multiple queues configured. A queue exist of elements. One element hold one Ethernet frame. The size of an element is configured with `EthC-`

`trlConfigEgressQueueBufLenByte` in bytes. The total amount elements of one queue is configured with `EthCtrlConfigEgressQueueBufTotal`. Thus, the total size in bytes of one queue is calculated as `EthCtrlConfigEgressQueueBufLenByte` multiplied with `EthCtrlConfigEgressQueueBufTotal`. The following sub-chapters describe the specific properties of `EthCtrlConfigIngressQueues` and `EthCtrlConfigEgressQueues`.

7.1.7.1 Ingress queue

An `EthCtrlConfig` could have 1 or more `EthCtrlConfigIngressQueues` configured. For each `EthCtrlConfigIngressQueue` a `EthCtrlConfigIngressQueueSortingType` could be assigned. `EthCtrlConfigIngressQueueSortingType` represents a Ethernet frame attribute used as filter to identify received Ethernet frame. Ethernet frames attributes of the received Ethernet frame which match to the configured `EthCtrlConfigIngressQueueSortingType` of `EthCtrlConfigIngressQueue` are sorted in that `EthCtrlConfigIngressQueue`. The following sorting types are supported:

- Destination MAC address (`EthCtrlIngressQueueSortingMacDestinationAssignment`)
- VLAN-ID (`EthCtrlIngressQueueSortingVlanIdAssignment`)
- VLAN priority (`EthCtrlIngressQueueSortingVlanPriorityAssignment`)
- EtherType (`EthCtrlIngressQueueSortingEtherTypeAssignment`)

[SWS_Eth_00331]

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[The configured `EthCtrlConfigIngressQueueSortingType` of an `EthCtrlConfigIngressQueue` shall be applied as filter on an Ethernet frame to identify a match. If a match is identified, then this Ethernet frame shall be enqueued in the affected `EthCtrlConfigIngressQueue`.]

[SWS_Eth_00332]

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If an Ethernet frame could not be identified as match to be enqueued in any configured `EthCtrlConfigIngressQueue` at the same Ethernet controller, then the Ethernet frame shall be dropped, and, if `EthDevErrorDetect` is set to `TRUE`, the Ethernet driver shall call `Det_ReportError` with the error code `ETH_E_NO_MATCHING_INGRESS_QUEUE_IDENTIFIED`.]

Example: If a `EthCtrlConfigIngressQueue` has `EthCtrlIngressQueueSortingVlanIdAssignment` set to 0x0FF (12bit value), then all receiving Ethernet frames, where VLAN-ID is set to 0x0FF are enqueued in this `EthCtrlConfigIngressQueue`

If multiple queues configured at the same `EthCtrlConfigIngressQueue` with different `EthCtrlConfigIngressQueueSortingTypes`, then the Ethernet controller need an prioritization in which order the sorting type should be applied to identify a match. Therefore a sorting priority has to be configured `EthCtrlConfigIngressQueueSortingPriority`. If no match is found for an receiving Ethernet frame, the Ethernet frame will be dropped.

Example

Configuration:

- `EthCtrlConfigIngressQueue A` has `EthCtrlIngressQueueSortingVlanIdAssignment` set to 0x0FF (12bit value)
- `EthCtrlConfigIngressQueue B` has `EthCtrlIngressQueueSortingEtherTypeAssignment` set to 0x22F0 (AVTP EtherType)
- `SortingPriorityEtherTypeAssignment` has priority 0
- `SortingPriorityVlanIdAssignment` has priority 1

Expected runtime behavior:

- An Ethernet frame with EtherType set to 0x22F0 is sorted in `EthCtrlConfigIngressQueue A`
- An Ethernet frame with EtherType set to 0x8100 and VLAN-ID set 0x0FF is sorted in `EthCtrlConfigIngressQueue B`
- An Ethernet frame with EtherType set to 0x8100 and VLAN-ID set 0x001 is dropped

[SWS_Eth_CONSTR_00005]

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If an `EthCtrlConfigIngress` of the same Ethernet controller have at least two `EthCtrlConfigIngressQueues` with different `EthCtrlConfigIngressQueueSortingTypes` configured, then a `EthCtrlConfigIngressQueueSortingPriority` shall be configured where the configured `EthCtrlConfigIngressQueueSortingType` are prioritized.]

[SWS_Eth_CONSTR_00006]

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[An Ethernet Controller shall have at most one `EthCtrlConfigIngressQueue` with the same value of `EthCtrlConfigIngressQueueSortingType` configured]

Note: Multiple `EthCtrlConfigIngressQueue` with the same value of `EthCtrlConfigIngressQueueSortingType` (e.g. two egress queues with sorting type `EtherType` configured to `0x22F0`) are invalid.

[SWS_Eth_00325]

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If an `EthCtrlConfig` have multiple `EthCtrlConfigIngressQueues` with different `EthCtrlConfigIngressQueueSortingTypes` configured, then the `EthCtrlConfigIngressQueueSortingType` with the highest priority `EthCtrlConfigIngressQueueSortingPriority` shall be applied to identify a match for this Ethernet frame. If no match could be identified, proceed in descending order with the next sorting `EthCtrlConfigIngressQueueSortingType`.]

[SWS_Eth_CONSTR_00007]

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[An `EthCtrlConfigIngressQueue` with no `EthCtrlConfigIngressQueueSortingType` configured, shall always have the lowest `EthCtrlConfigIngressQueueSortingPriority`.]

[SWS_Eth_CONSTR_00008]

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[An `EthCtrlConfig` shall have at most one `EthCtrlConfigIngressQueue` with no `EthCtrlConfigIngressQueueSortingTypes` configured.]

Note: A `EthCtrlConfigIngressQueue` with no `EthCtrlConfigIngressQueueSortingType` configured, could be used as default ingress queue where all Ethernet frames are added which could not be sorted in other ingress queues.

The Ethernet driver provide the possibility to configure the enqueueing behavior if an Ethernet controller is identified as matching Ethernet frame and all elements of the affected `EthCtrlConfigIngressQueue` are occupied. Either the Ethernet controller discard the Ethernet frame or the eldest available Ethernet frame in this `EthCtrlConfigIngressQueue`, which is not processed for reception, is overwritten. For some use cases it may be beneficial to allow overwriting of existing Ethernet frames (e.g. audio streaming).

[SWS_Eth_00334]

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If an Ethernet frame is identified to match an `EthCtrlConfigIngressQueueSortingType` of an `EthCtrlConfigIngressQueue` at an particular Ethernet controller, all elements of this `EthCtrlConfigIngressQueue` are occupied and `EthCtrlConfigIngressQueueOverwriteEnabled` of this `EthCtrlConfigIngressQueue` is

set to `FALSE`, then this Ethernet frame shall be discarded and a runtime error with error code `ETH_E_INGRESS_QUEUE_OCCUPIED` shall be reported.]

[SWS_Eth_00335]

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If an Ethernet frame is identified to match an `EthCtrlConfigIngressQueueSortingType` of an `EthCtrlConfigIngressQueue` at an particular Ethernet controller, all elements of this `EthCtrlConfigIngressQueue` are occupied and `EthCtrlConfigIngressQueueOverwriteEnabled` of this `EthCtrlConfigIngressQueue` is set to `TRUE`, then this Ethernet frame shall be enqueued by overwriting the `EthCtrlConfigIngressQueue` element where the eldest Ethernet frame reside which is not locked for reception.]

7.1.7.1.1 Ingress queue handler

An Ethernet controller receive an Ethernet frame, try to find a matching ingress queue and if an matching queue is found, enqueue this Ethernet frame in the according Ethernet ingress queue. An enqueueing of an Ethernet frame could be signaled as interrupt or the upper layer of the Ethernet driver is polling the ingress queues. Independent on the approach either "interrupt driven" or "polling", the communication stack need to dequeue the received Ethernet frames from the configures ingress queues. Therefore a so-called "ingress queue handler" is needed. An ingress queue handler is implementation specific. The Ethernet driver provide the possibility to configure polling and interrupt driven approaches, and to define an entry where to implement the ingress queue handler.

The following points summarize the possibility how `EthCtrlConfigIngressQueues` could be processed:

- Interrupt driven approach by setting `EthCtrlEnableRxInterrupt` to `TRUE`: enqueueing of an Ethernet frame at any `EthCtrlConfigIngressQueue` of the same Ethernet controller, signal an receive interrupt. A ingress queue handler is executed in the context of the ISR.
- Interrupt and polling driven approach by setting `EthCtrlEnableRxInterrupt` to `FALSE` and for specific `EthCtrlConfigIngressQueues`, `EthCtrlEnableIngressQueueInterrupt` to `TRUE`: enqueueing of an Ethernet frame at specific `EthCtrlConfigIngressQueues` signal an receive interrupt. An ingress queue handler is executed in the context of the ISR. The remaining `EthCtrlConfigIngressQueues` are polled in the context of the `EthIf_RxMainFunction`. An ingress queue handler is executed in the context of the `EthIf_RxMainFunction`
- Polling specific and polling driven approach by setting `EthCtrlEnableRxInterrupt` to `FALSE` and for specific `EthCtrlConfigIngressQueues` configure a `EthCtrlConfigIngressQueueHandlerFunction`: Ethernet frame at

specific `EthCtrlConfigIngressQueues` are polled in the configured `EthCtrlConfigIngressQueueHandlerFunction`. An ingress queue handler is executed in each configured `EthCtrlConfigIngressQueueHandlerFunction`. The `EthCtrlConfigIngressQueueHandlerFunction` may be scheduled by a CDD according to an external hardware unit (e.g. media clock). The remaining `EthCtrlConfigIngressQueues` are polled in the context of the `EthIf_RxMainFunction`. An ingress queue handler is executed in the context of the `EthIf_RxMainFunction`

- Polling specific and polling driven approach by setting `EthCtrlEnableRxInterrupt` to `FALSE` and for specific `EthCtrlConfigIngressQueues` configure at an `EthIfPhysController` multiple `EthIfPhysCtrlRxMainFunctionIngressProcessing` which could reference multiple `EthCtrlConfigIngressQueues`. An ingress queue handler is executed in each configured `EthIf_MainFunctionRx_<IngressQueueProcessing ShortName>`. The remaining `EthCtrlConfigIngressQueues` are polled in the context of the `EthIf_RxMainFunction`. An ingress queue handler is executed in the context of the `EthIf_RxMainFunction`

[SWS_Eth_00333]

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[An `EthCtrlConfigIngressQueue` with `EthCtrlConfigIngressQueueHandlerFunction` configured, shall be processed in the context of the generated ingress queue handler function.]

[SWS_Eth_00336]

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[An `EthCtrlConfigIngressQueue` with `EthCtrlEnableIngressQueueInterrupt` set to `TRUE`, shall be processed in the context of the signaled interrupt service routine.]

Ingress queues, which are polled by the upper layer (e.g. `EthIf`), call `Eth_Receive` to enqueue Ethernet frames.

[SWS_Eth_00337]

Status: DRAFT

[A call of `Eth_Receive` shall read the next frame from the receive buffers. The function passes the received frame to the Ethernet interface using the callback function `EthIf_RxIndication` and indicates if there are more frames in the receive buffers.]

7.1.7.2 Egress queue

An `EthCtrlConfig` could have 1 or more `EthCtrlConfigEgressQueues` configured. For each `EthCtrlConfigEgressQueue` a `EthCtrlConfigEgressQueueSortingType` could be assigned. `EthCtrlConfigEgressQueueSortingType` contains `EthCtrlConfigEgressQueueSortingEntry` which represents a field of the sorting filter to transmit an Ethernet frame. Ethernet frames attributes of the transmitted Ethernet frame which match to the configured `EthCtrlConfigEgressQueueSortingEntry` of `EthCtrlConfigEgressQueue` are sorted in that `EthCtrlConfigEgressQueue`. The following attributes are defining the sorting type :

- Sorting Key (`EthCtrlConfigEgressQueueSortingKey`)
- Sorting Mask (`EthCtrlConfigEgressQueueSortingMask`)
- Sorting Size (`EthCtrlConfigEgressQueueSortingSize`)
- Sorting Offset (`EthCtrlConfigEgressQueueSortingOffset`)

Note: `EthCtrlConfigEgressQueueSortingType` is optional for at most one `EthCtrlConfigEgressQueue` at the same `EthCtrlConfig` (see [SWS_Eth_CONSTR_00020]).

[SWS_Eth_00415] Apply `EthCtrlConfigEgressQueueSortingEntry` of an `EthCtrlConfigEgressQueue` on an Ethernet frame

Status: DRAFT

Upstream requirements: SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386

[The configured `EthCtrlConfigEgressQueueSortingEntry` of an `EthCtrlConfigEgressQueue` shall be applied as filter on an Ethernet frame to identify a match. If a match is identified, then this Ethernet frame shall be enqueued in the affected `EthCtrlConfigEgressQueue`.]

[SWS_Eth_00416] Error handling if an Ethernet frame could not be identified as match for an `EthCtrlConfigEgressQueue`

Status: DRAFT

Upstream requirements: SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386

[If an Ethernet frame could not be identified as match to be enqueued in any configured `EthCtrlConfigEgressQueue` at the same Ethernet controller, then the Ethernet frame shall be dropped and a runtime error with error code `ETH_E_NO_MATCHING_EGRESS_QUEUE_IDENTIFIED` shall be reported.]

[SWS_Eth_CONSTR_00018] Prioritization of `EthCtrlConfigEgressQueues` to identify a match for enqueueing an Ethernet frame

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If an `EthCtrlConfigEgress` of the same Ethernet controller have at least two `EthCtrlConfigEgressQueues` with different `EthCtrlConfigEgressQueueSortingTypes` configured, then it will be prioritized as per `EthCtrlConfigEgressQueueSortingEntry`, a lower index being the higher priority.]

[SWS_Eth_00417] Identification of an `EthCtrlConfigEgressQueue` match

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[If an `EthCtrlConfig` have multiple `EthCtrlConfigEgressQueues` configured, then the `EthCtrlConfigEgressQueueSortingType` with the highest priority i.e. lowest index of `EthCtrlConfigEgressQueueSortingEntry` shall be applied to identify a match for this Ethernet frame. If no match could be identified, proceed in descending order with the next sorting `EthCtrlConfigEgressQueueSortingType`.]

[SWS_Eth_CONSTR_00019] `EthCtrlConfigEgressQueue` with no `EthCtrlConfigEgressQueueSortingType` configured

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[An `EthCtrlConfigEgressQueue` with no `EthCtrlConfigEgressQueueSortingType` configured, shall always have the lowest priority i.e. highest index of `EthCtrlConfigEgressQueueSortingEntry`.]

[SWS_Eth_CONSTR_00020] `EthCtrlConfigEgressQueue` with no `EthCtrlConfigEgressQueueSortingType` per `EthCtrlConfig`

Status: DRAFT

Upstream requirements: [SRS_Eth_00171](#), [SRS_BSW_00350](#), [SRS_BSW_00386](#)

[An `EthCtrlConfig` shall have at most one `EthCtrlConfigEgressQueue` with no `EthCtrlConfigEgressQueueSortingTypes` configured.]

Note: A `EthCtrlConfigEgressQueue` with no `EthCtrlConfigEgressQueueSortingType` configured, could be used as default egress queue where all Ethernet frames are added which could not be sorted in other egress queues.

7.1.7.2.1 Egress - shapers and transmission selection

Ethernet frames are enqueued in egress queues according to their priority assignment. A Ethernet frame stay in the egress queue as long as the so-called `EthCtrlConfigScheduler` select an Ethernet frame for transmission. Each egress queue (see `EthCtrlConfigEgressQueue`) has to configure the algorithm to select the Ethernet

frames for transmission. Therefore each egress queue has an mandatory sub container `EthCtrlConfigEgressQueueTransmissionSelection`. `EthCtrlConfigEgressQueueTransmissionSelection` defines the selection algorithm via `EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm` (e.g. credit based shaper, asynchronous traffic shaper ... a.s.o.). Each `EthCtrlConfigEgressQueue` is connected to an egress scheduler. The egress scheduler has to schedule all connected egress queues. Each egress scheduler has an mandatory sub container `EthCtrlConfigScheduler` which defines the scheduler algorithm via `EthCtrlConfigSchedulerAlgorithm` (e.g. strict priority). Multiple egress schedulers at the same `EthCtrl` could be configured and connected in a cascaded manner. Thus, the output of an egress scheduler is used as an input for the sub sequential egress scheduler. [Figure 7.5](#) shows examples for an egress port structure.

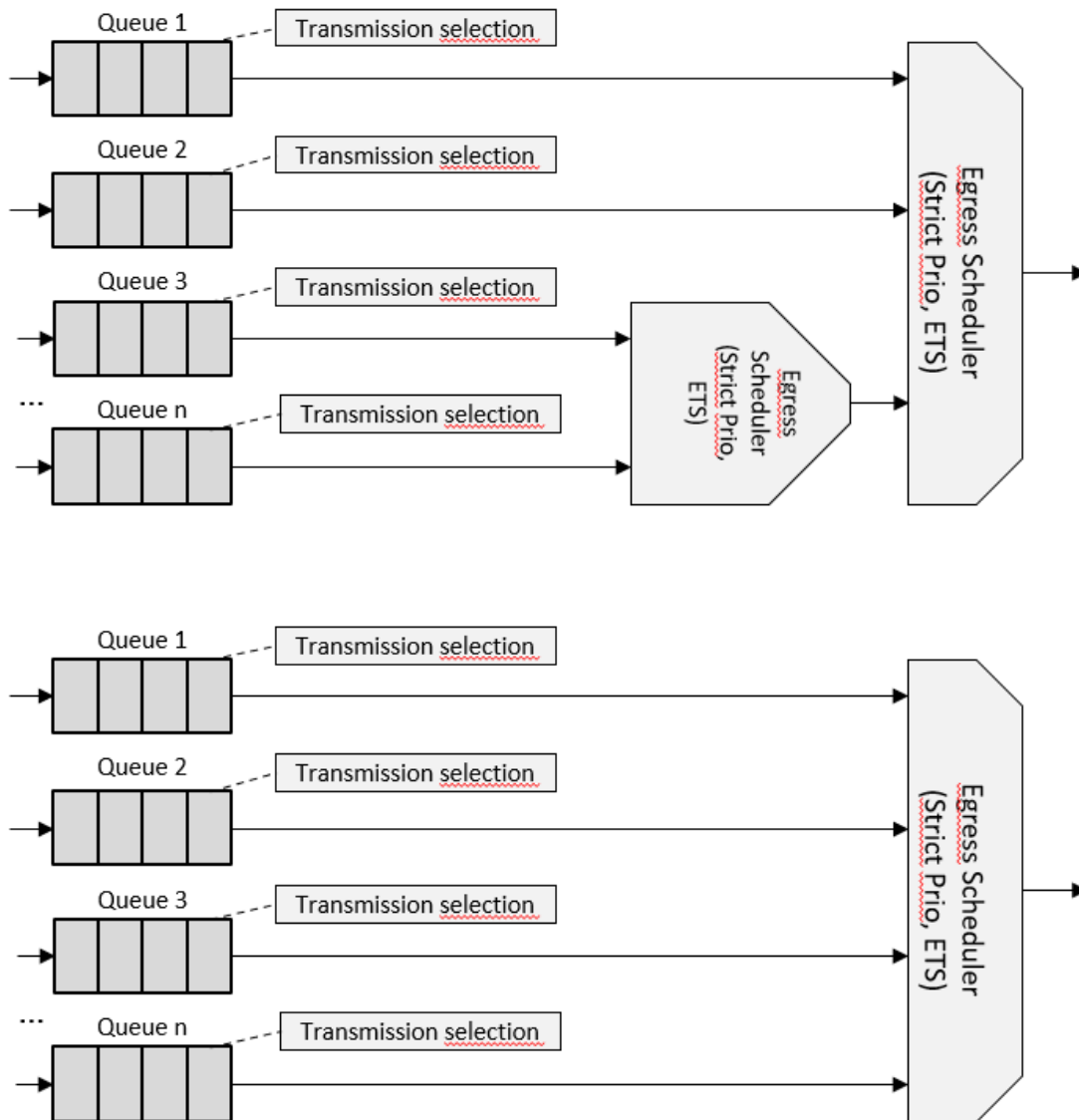


Figure 7.5: Examples for an egress structure at an EthCtrl

The egress scheduler algorithm schedule its input (either an egress queue or an egress scheduler) by considering the according properties (e.g. traffic class assignment). Once the egress scheduler algorithm has decided which of its input should be handled, the egress scheduler select an Ethernet frame from the according egress queue based on the configured transmission selection algorithm:

- If the transmission selection algorithm is configured as credit based shaper, then the according egress queue is handled as FIFO. The egress queue has credit, which is constantly increased according to the configured bandwidth and decreased for each transmission of Ethernet frame from this egress queue.
- If the transmission selection algorithm is configured as asynchronous traffic shaper, then the according egress queue is handled as queue. Each Ethernet frame of the queue has an assigned eligibility time. According the eligibility time a Ethernet frame is selected from this egress queue. The Ethernet frames are not handled according the arrival in this egress queue, but according the assigned eligibility time which has been added.
- If the transmission selection algorithm is configured as unshaped, then the according egress queue is handled as FIFO.
- If the transmission selection algorithm is configured as enhanced traffic shaping, then the according egress queue is handled as queue.

Note: The parameterization of the egress scheduler in combination with the egress queue at an EthCtrl influences the egress latency for transmission of Ethernet frames via this EthCtrl.

The configuration of the egress schedulers is done with the container `EthCtrlConfigScheduler` and its sub-container `EthCtrlConfigSchedulerPredecessor` with multiplicity 1 to *. Egress scheduler connect its predecessors with the predecessor references `EthCtrlConfigSchedulerPredecessorRef`. An egress scheduler could either have an further egress scheduler or a egress queue as predecessor.

Egress queues are considered as neighboring egress queues if the are referenced by the same `EthCtrlConfigScheduler` via `EthCtrlConfigSchedulerPredecessorRef`. The composition of an egress queue(s) and its direct connected `EthCtrlConfigScheduler` form an Ethernet frame processing unit, where its output is used as input to the connected successor. The very last successor at an `EthCtrlConfigEgress` is always an `EthCtrlConfigScheduler` referenced via `EthCtrlConfigEgressLastSchedulerRef`.

Please note, the configured egress structure is an configuration model and does not reflect the hardware implementation at an egress of an Ethernet controller.

[SWS_Eth_CONSTR_00013] Definition of neighboring egress queues*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[Egress queues shall be considered as neighboring egress queues if they are referenced by the same [EthCtrlConfigScheduler](#) via [EthCtrlConfigScheduler-PredecessorRef](#).]

[SWS_Eth_00401] Ethernet frame handling according the configured transmission selection algorithm*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[If an Ethernet frame is added to an [EthCtrlConfigEgressQueue](#), then the Ethernet controller shall handle this Ethernet frame according the configured transmission selection algorithm ([EthCtrlConfigEgressQueueTransmissionSelection](#)) of this [EthCtrlConfigEgressQueue](#) and with respect to the configured egress structure ([EthCtrlConfigScheduler](#), [EthCtrlConfigSchedulerPredecessor](#)) of the corresponding Ethernet controller egress ([EthCtrlConfigEgress](#))]

7.1.7.2.2 Details on egress scheduler

As mentioned before [EthCtrlConfigScheduler](#) select Ethernet frames which are offered to be transmitted by the [EthCtrlConfigEgressQueue](#) based on the configured [EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm](#). The [EthCtrlConfigScheduler](#) examine each of its incoming predecessor [EthCtrlConfigEgressQueue](#), starting with the [EthCtrlConfigEgressQueue](#) where the highest priority is assigned and proceed in descending order. This scheduling process highly depends on the configuration of the [EthCtrlConfigSchedulerAlgorithm](#), the [EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm](#) of each relevant egress queue and the egress structure.

The configuration supports the following [EthCtrlConfigSchedulerAlgorithms](#):

- [ETH_SCHEDULER_STRICT_PRIORITY](#): The egress scheduler always selects a relevant egress queue with the highest assigned priority, that offers an emission opportunity to dequeue an Ethernet frame. After each dequeued Ethernet frame the scheduling algorithm checks for current available offers of an egress queue with a higher assigned priority before proceeding. If no other egress queue with a higher assigned priority offers an emission opportunity the scheduling algorithm proceeds by either dequeuing further Ethernet frames of the current processed egress queue or by scheduling the next egress queue in descending order.
- [ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER](#): The egress scheduler starts with the [EthCtrlConfigEgressQueue](#) where the highest priority is assigned and proceed in descending order. If reaching the last relevant egress queue, the scheduling algorithm continues with the egress queue where the highest priority

is assigned. This round robin scheduling strictly keeps the order of the scheduled relevant egress queues.

[SWS_Eth_00402] Scheduling with strict priority

Status: DRAFT

Upstream requirements: [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[If an [EthCtrlConfigSchedulerAlgorithm](#) is configured with [ETH_SCHEDULER_STRICT_PRIORITY](#), then the egress scheduler shall always select a relevant egress queue with the highest assigned priority, that offers an emission opportunity to dequeue an Ethernet frame.]

[SWS_Eth_00403] Scheduling with enhanced traffic shaping

Status: DRAFT

Upstream requirements: [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[If an [EthCtrlConfigSchedulerAlgorithm](#) is configured with [ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER](#), then the egress scheduler shall start with the egress queue where the highest priority is assigned and proceed in descending order. If reaching the last relevant egress queue, the scheduling algorithm shall continue with the egress queue where the highest priority is assigned.]

The [EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm](#) and the according configuration defines the amount of Ethernet frames which are dequeued, if the egress scheduler select an relevant egress queue.

[SWS_Eth_00404] Dequeueing Ethernet frames with strict priority scheduling

Status: DRAFT

Upstream requirements: [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[If an [EthCtrlConfigSchedulerAlgorithm](#) is configured with [ETH_SCHEDULER_STRICT_PRIORITY](#) and a relevant egress queue is selected that offers an emission opportunity, then the egress scheduler shall dequeue Ethernet frames from this egress queue until either of the following conditions is valid:

- an egress queue with a higher priority offers an emission opportunity
- the emission offer opportunity of this egress queue is suspended
- no further Ethernet frames are available in this egress queue

]

[SWS_Eth_00405] Emission opportunity suspension with strict priority scheduling

Status: DRAFT

Upstream requirements: [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[If an [EthCtrlConfigSchedulerAlgorithm](#) is configured with [ETH_SCHEDULER_STRICT_PRIORITY](#) and the connected egress queue is dequeued

by the `EthCtrlConfigScheduler`, then a connected egress queue shall suspend its emission opportunity in dependency to the configured `EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm` according the following conditions:

- If set to `ETH_TRANSMISSION_SELECTION_UNSHAPED`, then the emission opportunity is suspended if no Ethernet frame resides in the egress queue
- If set to `ETH_TRANSMISSION_SELECTION_CBS`, then the emission opportunity is suspended if the credit of this egress queue is lower than 0
- If set to `ETH_TRANSMISSION_SELECTION_ATS`, then the emission opportunity is suspended if all Ethernet frames are dequeued which are eligible for transmission
- If set to `ETH_TRANSMISSION_SELECTION_ETS`, then the emission opportunity is suspended under either the following conditions:
 - if the configured limit (either in weight or in percent of the available bandwidth see (`EthCtrlConfigEgressQueueTransmissionSelectionETS`)) exceeds and at least one neighboring egress queue configured to `ETH_TRANSMISSION_SELECTION_ETS` resume its emission opportunity
 - no further Ethernet frames resides in this egress queue

]

Please note: It is recommended to use `ETH_TRANSMISSION_SELECTION_UNSHAPED` for egress queues, where the application design of the ECU ensure limited amount of Ethernet frames. Otherwise Ethernet frames in egress queues assigned to lower priorities may be confronted with high transmission delay.

[SWS_Eth_00406] Emission opportunity suspension with enhanced traffic shaping

Status: DRAFT

Upstream requirements: [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[If an `EthCtrlConfigSchedulerAlgorithm` is configured with `ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER` and the connected egress queue is dequeued by the `EthCtrlConfigScheduler`, then a connected egress queue shall suspend its emission opportunity in dependency to the configured `EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm` according the following conditions:

- If set to `ETH_TRANSMISSION_SELECTION_ETS`, then the emission opportunity is suspended under either the following conditions:
 - if the configured limit (either in weight or in percent of the available bandwidth see (`EthCtrlConfigEgressQueueTransmissionSelectionETS`)) exceeds and at least one neighboring egress queue configured to `ETH_TRANSMISSION_SELECTION_ETS` resume its emission opportunity

- no further Ethernet frames resides in this egress queue

]

An `EthCtrlConfigScheduler` where the `EthCtrlConfigSchedulerAlgorithm` is configured to `ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER` could only handle egress queues where the `EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm` are configured to `ETH_TRANSMISSION_SELECTION_ETS`. An egress queue configured with `ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER` as its `EthCtrlConfigSchedulerAlgorithm` need to have at least one neighboring egress queue with the same `EthCtrlConfigSchedulerAlgorithm`. An `EthCtrlConfigScheduler` where the `EthCtrlConfigSchedulerAlgorithm` is configured to `ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER` shape the traffic of all direct connected egress queues to get an fair bandwidth distribution in traffic congestion scenarios where at least two egress queues resume the emission opportunity.

[SWS_Eth_CONSTR_00014] Egress configuration constraint for scheduling with enhanced traffic shaping

Status: DRAFT

Upstream requirements: [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[A configuration where an `EthCtrlConfigScheduler` has set the `EthCtrlConfigSchedulerAlgorithm` to `ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER` shall support to have egress queues with `EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm` set to `ETH_TRANSMISSION_SELECTION_ETS`. All other `EthCtrlConfigEgressQueueTransmissionSelectionAlgorithms` are not supported in combination with `EthCtrlConfigScheduler` set to `ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER`.]

[SWS_Eth_CONSTR_00015] Enhanced traffic shaping require at least to egress queues

Status: DRAFT

Upstream requirements: [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[A configuration where an `EthCtrlConfigScheduler` has set the `EthCtrlConfigSchedulerAlgorithm` to `ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER` shall have at least two egress queues with `EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm` set to `ETH_TRANSMISSION_SELECTION_ETS` as direct connected predecessors.]

The combination of `EthCtrlConfigScheduler` set the `EthCtrlConfigSchedulerAlgorithm` to `ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER` and egress queues with `EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm` set to `ETH_TRANSMISSION_SELECTION_ETS` realize round-robin network traffic handling. The available bandwidth for transmission of enqueued Ethernet frames of the configured egress queues, are configured per egress queue via `EthCtrlConfigEgressQueueTransmissionSelectionETS`. The configuration supports the following options:

- configuration of available bandwidth as weights of Ethernet frames via `EthCtrlETSTConfigAvailableBandwidthInWeightValue`
- configuration of available bandwidth in percent via `EthCtrlETSTConfigAvailableBandwidthInPercent`

Independent which configuration variant for `EthCtrlConfigEgressQueueTransmissionSelectionETS` is used, all egress queues that are scheduled by the same `EthCtrlConfigScheduler` with `EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm` set to `ETH_TRANSMISSION_SELECTION_ETS` should use the same variant of `EthCtrlConfigEgressQueueTransmissionSelectionETS`

[SWS_Eth_CONSTR_00016] Neighboring egress queues need the same variant of availability bandwidth configuration

Status: DRAFT

Upstream requirements: [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[If an `EthCtrlConfigScheduler` is configured with `EthCtrlConfigSchedulerAlgorithm` set to `ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER`, then all egress queues which are configured for this `EthCtrlConfigScheduler` shall exclusively use the same configuration of `EthCtrlConfigEgressQueueTransmissionSelectionETS`:

- `EthCtrlETSTConfigAvailableBandwidthInWeightValue` XOR
- `EthCtrlETSTConfigAvailableBandwidthInPercent`

]

The configuration for the ETS traffic shaping allows the following variants:

- `EthCtrlETSTConfigAvailableBandwidthInWeightValue`: the available bandwidth is configured in weights, where the weights represents the amount of Ethernet frames
- `EthCtrlETSTConfigAvailableBandwidthInPercent`: the available bandwidth per egress queue is configured in percent

Both configuration variants are based on congestion scenario where all neighboring egress queues consume their bandwidth.

The configuration of the available bandwidth in weights as Ethernet frames need to be considered in relation to the emission of all neighboring egress queues. The sum of all configured weights as Ethernet frames across all neighboring egress queues reflect one emission portion of Ethernet frames. If an emission portion of Ethernet frames were processed by an `EthCtrlConfigScheduler`, then the amount of configured weights as Ethernet frames per neighboring egress queue should be enclosed in the emission portion of processed Ethernet frames. Or in other words, the configured available bandwidth in weights as Ethernet frames of each neighboring egress queue should be processed, if an emission portion of Ethernet frames were processed by the `EthCtrlConfigScheduler`.

The configuration of the available bandwidth in percent need to be considered in relation to a measurement interval. This interval defines the time slot which is used to calculate the expected emission of each egress at the same [EthCtrlConfigScheduler](#).

Note: For both configuration variants count, the order of Ethernet frames, either within one emission portion or within the measurement interval, depends on the implemented scheduler algorithm (e.g. weighted round robin, deficit round robin) and is not defined / configurable by the Eth driver module.

[SWS_Eth_00407] Determination of egress queue emission with available bandwidth configured in weights as amount of Ethernet frames

Status: DRAFT

Upstream requirements: [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[If the available bandwidth of neighboring egress queues configured with [EthCtrlETSTConfigAvailableBandwidthInWeightValue](#) and all egress queues consume their available bandwidth, then the emission of all neighboring egress queues shall be determined by the following considerations:

The emission of one egress queue in Ethernet frames:

$$em_{queue\ n}[Ethernet\ frames] = weight_{queue\ n}[Ethernet\ frames] \quad (7.1)$$

em_{queue n}: emission of egress queue n in unit of Ethernet frames

weight_{queue n}: [EthCtrlETSTConfigAvailableBandwidthInWeightValue](#) configured for queue n in unit of Ethernet frames

One emission portion is equal to the configured emission of all neighboring egress queues in Ethernet frames:

$$em_{Ethernet\ frames} = \sum_{n=1}^N em_{queue}(n) \quad (7.2)$$

N: count of neighboring egress queues

em_{neighboring queues}: emission in unit of Ethernet frames of all neighboring egress queues

em_{queue n}: emission of egress queue n in unit of Ethernet frames

]

[SWS_Eth_00408] Amount of Ethernet frames within one emission portion if available bandwidth in weights is configured as amount of Ethernet frames*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00121](#), [SRS_Eth_00179](#), [SRS_Eth_00180](#)

[If an [EthCtrlConfigSchedulerAlgorithm](#) is configured with [ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER](#) and the [EthCtrlConfigEgressQueueTransmissionSelectionETS](#) is set to [EthCtrlETSConfigAvailableBandwidthInWeightValue](#) and all neighboring egress queues offers an emission opportunity during the procession of one emission portion, then the emission portion shall enclose the amount of Ethernet frames of each neighboring egress queue configured via [EthCtrlETSConfigAvailableBandwidthInWeightValue](#).]

[SWS_Eth_00409] Tolerance of egress queue emission within the defined measurement interval if available bandwidth is configured in percent is used*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[If an [EthCtrlConfigSchedulerAlgorithm](#) is configured with [ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER](#) and the [EthCtrlConfigEgressQueueTransmissionSelectionETS](#) of all neighboring egress queues is set to [EthCtrlETSConfigAvailableBandwidthInPercent](#) and all neighboring egress queues offers an emission opportunity during a defined measurement interval, then the emission of all egress queues during this defined measurement interval shall reflect the configured bandwidth limitation of neighboring egress queues (either in weights or in percent of the available bandwidth see ([EthCtrlConfigEgressQueueTransmissionSelectionETS](#))) with a tolerance of 10 % (see [13, IEEE Std 802.1Q] chapter "ETS algorithm").]

The definition of a measurement interval need to consider the line rate ([EthCtrlMacLayerSpeed](#)) of the according [EthCtrlConfig](#). The [EthCtrlMacLayerSpeed](#) defines the bit time. The defined measurement interval divided by bit time defines the amount of bits which is expected for the emission of all neighboring egress queues.

[SWS_Eth_00410] Definition of bit time*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[The time consumed to transmit a bit (bit time) shall be calculated according the following equation:

$$bit_{time}[\frac{s}{bit}] = \frac{1}{line_{rate}[\frac{Bit}{s}]} \quad (7.3)$$

line_{rate}[Bit/s]: [EthCtrlMacLayerSpeed](#)]

[SWS_Eth_00411] Determination of egress queue emission with available bandwidth configured in percent

Status: DRAFT

Upstream requirements: [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[If the available bandwidth of neighboring egress queues are configured with [EthCtrlETSTConfigAvailableBandwidthInPercent](#) and all egress queues consume their available bandwidth, then the emission per egress queue within the defined measurement interval of the according [EthCtrlConfigScheduler](#) shall be determined by the following calculations:

The emission of all neighboring egress queues in bits:

$$em_{\text{neighboring queues}}(T) = \frac{T[s]}{bit_{\text{time}}[\frac{s}{bit}]} \quad (7.4)$$

T: measurement interval in seconds

em_{neighboring queues}: emission in bits of all neighboring egress queues per defined measurement interval

The emission of one egress queue in bits:

$$em_{\text{queue } n}(T)[bit] : em_{\text{neighboring queues}}(T)[bit] * \frac{bw_{\text{queue } n}}{100} \quad (7.5)$$

em_{queue n}: emission of egress queue n in bits during the defined measurement interval

bw_{queue n}: bandwidth of queue n in percent configured via [EthCtrlETSTConfigAvailableBandwidthInPercent](#)]

Note: If the available bandwidth of neighboring egress port queues is configured with [EthCtrlETSTConfigAvailableBandwidthInPercent](#), then the total number of bits that are consumed on the medium by the transmission of the according Ethernet frames need to be considered for determining an emission, i.e. including all required framing bits like preamble, start frame delimiter (SFD), frame check sequence (FCS) and minimum inter-packet gap (IPG).

[SWS_Eth_CONSTR_00017] Constraint for configuration of available bandwidth in percent

Status: DRAFT

Upstream requirements: [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[If the available bandwidth of neighboring egress queues configured with [EthCtrlETSTConfigAvailableBandwidthInPercent](#), then the sum of the configured available bandwidth of all neighboring egress shall result in 100 %.]

The available bandwidth of the neighboring egress queues need to be shared on runtime, if egress queues have bandwidth left over and their emission opportunity was resumed. A round-robin traffic shaping configured with [EthCtrlConfigScheduler](#)

set the `EthCtrlConfigSchedulerAlgorithm` to `ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER` should always try to utilize the complete available bandwidth.

[SWS_Eth_00412] Utilization of all neighboring egress queues with a resumed emission opportunity to approximantely 100 %

Status: DRAFT

Upstream requirements: [SRS_Eth_00177](#), [SRS_Eth_00184](#)

[Neighboring egress queues with `EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm` set to `ETH_TRANSMISSION_SELECTION_ETS` shall equally share remaining available bandwidth on runtime. The sum of utilized bandwidth of all neighboring egress queues with a resumed emission opportunity should result

- in the amount of Ethernet frames covered by one emission portion if available bandwidth is configured in weights as amount of Ethernet frames
- in approximately 100 % if available bandwidth is configured in percent

]

The available bandwidth per neighboring egress queues could deviate on runtime from the configured available bandwidth (either in weights or in percent of the available bandwidth see (`EthCtrlConfigEgressQueueTransmissionSelectionETS`)). The configured bandwidth represent the expected utilized bandwidth if all neighboring egress queues consume their configured bandwidth within the defined measurement interval, i.e. all neighboring egress queues resume the emission opportunity.

7.1.8 Support of frame preemption

The the Eth driver module supports the configuration of frame preemption per `EthCtrlConfig` (see [13, IEEE Std 802.1Q-2022]), if the Ethernet switch hardware support the functionality. By default it is assumed that Ethernet controller hardware is not supposed to perform or not even capable of frame preemption, thus `EthCtrlFramePreemptionEnable` is set to FALSE for the respective `EthCtrlConfig`.

[SWS_Eth_CONSTR_00021] Ethernet controller hardware support for frame preemption

Status: DRAFT

Upstream requirements: [SRS_Eth_00185](#)

[If and only if an Ethernet controller hardware supports frame preemption, then it shall be allowed to set `EthCtrlFramePreemptionEnable` to TRUE for the resepective `EthCtrlConfig`.]

[SWS_Eth_CONSTR_00022] Frame preemption enabling on ingress and egress per `EthCtrlConfig`*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00185](#)

[If `EthCtrlFramePreemptionEnable` is set to TRUE for an `EthCtrl` (`EthCtrlConfig`), then frame preemption handling shall be enabled on ingress and egress for this `EthCtrl`. Otherwise frame preemption shall be disabled.]

Frame preemption is working on a per link basis. Thus, both ends of the link need to support the frame preemption, otherwise it does not work properly. In dynamic networks the frame preemption capability between peers negotiated via dedicated layer 2 protocols (e.g. LLDP). However, such protocols are not supported by AUTOSAR. Automotive networks are in addition rather statically designed, rendering those protocols unnecessary, as a proper communication network design is ensured by the system description (see [14]), such that a system-wide consistent support of frame preemption can be ensured by the configuration and the data model.

7.1.9 HW Clock Handling

If HW Timestamping support is enabled (`EthGlobalTimeSupport` is set to TRUE), it is expected, that the Ethernet Controller supports a HW clock to perform HW timestamping for Timesync frames (EtherType = 0x88F7) ingressed and egressed on the controller port (refer to [Chapter 7.1.9.1](#)).

In addition, if supported by the Ethernet Controller, the Ethernet Driver may support an adjustable PTP HW clock (PHC), i.e., a clock that is adjustable in rate and offset. (refer to chapter [Chapter 7.1.9.2](#)).

Finally, if a PHC is supported, the Ethernet Driver may also support the generation of a Pulse-Per-Second (PPS) signal (refer to chapter [Chapter 7.1.9.3](#)).

7.1.9.1 HW Timestamping

If the Ethernet Controller supports HW timestamping (refer to `EthGlobalTimeSupport`), the Ethernet Driver module will provide the following APIs to the upper layer to enable HW timestamping:

- `Eth_EnableEgressTimeStamp` to enable timestamping for a frame
- `Eth_GetEgressTimeStamp` to read the egress timestamp of a transmitted frame.
- `Eth_GetCurrentTimeTuple` to read the current value of the timestamping HW clock and, if supported, the current value of the PTP HW clock (PHC)

7.1.9.2 Adjustable PTP HW Clock (PHC)

If the Ethernet Controller supports an adjustable PTP HW Clock (refer to [EthPhcSupport](#)), the Ethernet Driver allows the upper layer to read and set the PHC using the following APIs:

- [Eth_GetPhcTime](#) to read the current value of the PHC
- [Eth_SetPhcTime](#) to set the current value of the PHC
- [Eth_SetPhcCorrection](#) to apply a given rate and offset value as correction for the PHC

[Eth_SetPhcTime](#) is used to set an absolute value of a PHC. [Eth_SetPhcCorrection](#) is used to apply rate and offset correction to an PHC. [Eth_SetPhcTime](#) is typically called if the upper layer detect a jump of the synchronized time (e.g. after first reception of a time sync message from a global time provider). Afterwards the PHC is adjusted with rate deviation and offset correction values which are calculated by the upper layer as deviation from a global time provider. The upper layer is responsible to call [Eth_SetPhcTime](#) and [Eth_SetPhcCorrection](#) in a sensible way.

[SWS_Eth_CONSTR_00010]

Status: DRAFT

Upstream requirements: [SRS_Eth_00167](#)

[If [EthGlobalTimeSupport](#) is set to FALSE, then [EthPhcSupport](#) shall be set to FALSE]

[SWS_Eth_00373]

Status: DRAFT

Upstream requirements: [SRS_Eth_00167](#)

[If [Eth_SetPhcTime](#) or [Eth_SetPhcCorrection](#) is called and the given [EthClkUnitIdx](#) address an [EthClkUnit](#) where all referenced [EthCtrlClks](#) have [EthCtrlClkAdjustmentEnable](#) set to FALSE, then the Ethernet driver shall return with E_NOT_OK, or, if development error detection is enabled ([EthDevErrorDetect](#) set to TRUE), the Ethernet driver shall report development error [ETH_E_CLOCK_ADJUSTMENT_FAILED](#).]

[SWS_Eth_00374]

Status: DRAFT

Upstream requirements: [SRS_Eth_00167](#)

[If [Eth_SetPhcTime](#) is called and the given [EthClkUnitIdx](#) address an [EthClkUnit](#) where a referenced [EthCtrlClk](#) has [EthCtrlClkAdjustmentEnable](#) set to TRUE, then the Ethernet driver shall apply the timestamp value given with [timestampPtr](#) to this [EthCtrlClk](#).]

[SWS_Eth_00375]

Status: DRAFT

Upstream requirements: [SRS_Eth_00167](#)

[If [Eth_SetPhcCorrection](#) is called and the given [EthClkUnitIdx](#) address an [EthClkUnit](#) where a referenced [EthCtrlClk](#) has [EthCtrlClkAdjustmentEnable](#) set to TRUE, then the Ethernet driver shall apply the value for rate deviation given with [rateDeviation](#) and the value for offset correction given with [offset](#) to this [EthCtrlClk](#).]

[SWS_Eth_CONSTR_00011]

Status: DRAFT

Upstream requirements: [SRS_Eth_00167](#)

[Two different [EthCtrlClks](#) which are referenced by the same [EthClkUnit](#) via [EthClkUnitTimePhcRef](#) and [EthClkUnitTimeStampingRef](#) shall allow one of the following configurations, all other constellations shall be rejected as invalid:

- both [EthCtrlClks](#) have [EthCtrlClkAdjustmentEnable](#) set to FALSE
- [EthCtrlClk](#) referenced via [EthClkUnitTimePhcRef](#) shall have [EthCtrlClkAdjustmentEnable](#) set to TRUE and [EthCtrlClk](#) referenced via [EthClkUnitTimeStampingRef](#) shall have [EthCtrlClkAdjustmentEnable](#) set to FALSE.

]

7.1.9.2.1 Cross-Timestamping of PTP HW Clock and Timestamping Clock

If a PHC is supported, the upper layer time synchronization protocol that makes use of it needs to correlate the PHC value to the timestamping clock value, i.e., it needs to do a crosstimestamping of the two clocks.

[SWS_Eth_00339]

Status: DRAFT

Upstream requirements: [SRS_Eth_00167](#)

[If [EthClkUnitCrossTimestampingSupport](#) is set to [HW_XTIMESTAMPING](#), then the Ethernet Driver shall trigger the cross-timestamping in HW in the context of [Eth_GetCurrentTimeTuple](#) of the given [EthClkUnit](#) and read

- the cross-timestamped value of the PTP HW clock which is referenced via [EthClkUnitTimePhcRef](#)
- and the cross-timestamped value of the timestamping HW clock which is referenced via [EthClkUnitTimeStampingRef](#).

and return the values as [TimeTupleType](#) addressed via out paramter [currentTimeTuplePtr](#) of [Eth_GetCurrentTimeTuple](#) by

- setting the `disciplinedClockValue` of `TimeTupleType` to the cross-timestamped value of the PTP HW clock
- and setting the `timestampClockValue` of `TimeTupleType` to the cross-timestamped value of the timestamping HW clock

]

Note: HW supported cross-timestamping is a very HW dependend feature, which is not further detailed in this document.

[SWS_Eth_00340]

Status: DRAFT

Upstream requirements: [SRS_Eth_00167](#)

[If `EthClkUnitCrossTimestampingSupport` is set to `SW_XTIMESTAMPING`, then the Ethernet Driver shall perform two consecutive read operations of the given `EthClkUnit` in the context of `Eth_GetCurrentTimeTuple` for reading

- the value of the PTP HW clock which is referenced via `EthClkUnit-TimePhcRef`
- the value of the timestamping HW clock which is referenced via `EthClkUnit-TimeStampingRef`

and return the values as `TimeTupleType` addressed via out parameter `currentTimeTuplePtr` of `Eth_GetCurrentTimeTuple` by

- setting the `disciplinedClockValue` of `TimeTupleType` to the cross-timestamped value of the PTP HW clock
- and setting the `timestampClockValue` of `TimeTupleType` to the cross-timestamped value of the timestamping HW clock

]

[SWS_Eth_00341]

Status: DRAFT

Upstream requirements: [SRS_Eth_00167](#)

[If `EthClkUnitCrossTimestampingSupport` is set to `NO_XTIMESTAMPING`, then the Ethernet Driver shall read the value of the timestamping HW clock, which is referenced via `EthClkUnitTimeStampingRef` by the given `EthClkUnit`, in context of `Eth_GetCurrentTimeTuple` and return the value as `TimeTupleType` addressed via out parameter `currentTimeTuplePtr` of `Eth_GetCurrentTimeTuple`, where `disciplinedClockValue` and `timestampClockValue` of `TimeTupleType` are set to same value read from the timestamping HW clock]

7.1.9.3 Generation of a Pulse-Per-Second (PPS) Signal

A Pulse-Per-Second signal allows to compare the phase of a HW clock to a reference clock. Refer to [15, FO_EXP_TimeSensitiveNetworkFeatures] for more details. It is assumed that the PPS signal generation as configured by the [EthCtrlPulsePerSecondConfig](#) is derived automatically from the PHC and driven in hardware.

[SWS_Eth_CONSTR_00012]

Status: DRAFT

Upstream requirements: [SRS_Eth_00168](#)

[A [EthCtrlPulsePerSecondConfig](#) configuration shall be rejected as invalid, if the affected Ethernet controller hardware do not support PPS signal generation.]

[SWS_Eth_00342]

Status: DRAFT

Upstream requirements: [SRS_Eth_00168](#)

[If [EthCtrlPulsePerSecondConfig](#) is configured and the affected Ethernet controller support PPS signal generation and PPS signal output property configuration ([EthCtrlPulsePerSecondDutyCycle](#), [EthCtrlPulsePerSecondFrequency](#) and [EthCtrlPulsePerSecondStartEnum](#)), then the Ethernet Driver shall use the configuration [EthCtrlPulsePerSecondConfig](#) to configure the PHC (referenced by [EthCtrlPulsePerSecondClockRef](#) such that it generates

- a square wave PPS signal
- with a duty cycle of [EthCtrlPulsePerSecondDutyCycle](#)
- and a frequency of [EthCtrlPulsePerSecondFrequency](#)

]

[SWS_Eth_00377]

Status: DRAFT

Upstream requirements: [SRS_Eth_00168](#)

[If [EthCtrlPulsePerSecondConfig](#) is configured and the affected Ethernet controller is limited to PPS signal generation and has no capability to configure the PPS signal output properties ([EthCtrlPulsePerSecondDutyCycle](#), [EthCtrlPulsePerSecondFrequency](#) and [EthCtrlPulsePerSecondStartEnum](#)), then the Ethernet Driver shall consider only those PPS signal output configuration properties which are supported by hardware.]

Note: If an Ethernet controller hardware is limited to generate a PPS signal without having capability to configure the PPS signal output properties (e.g. frequency, duty cycle), then it should still be possible to use this PPS signal generation. It is recommended to use configured PPS signal output properties for hardware configuration only. It is not recommended to cover missing hardware capabilities for PPS signal output property configuration in software, since this could impact accuracy of the PPS signal generation.

[SWS_Eth_00376]*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00168](#)

[If [EthCtrlPulsePerSecondConfig](#) has [EthCtrlPulsePerSecondStartEnum](#) set to `RISING_EDGE` and the Ethernet controller hardware support configure PPS signal output properties, then the periode of the square wave PPS signal shall start with a rising edge. Otherwise the square wave PPS signal shall start with a falling edge.]

Note: The HW will only start/stop generation of the PPS signal, if explicitly requested by [Eth_SetPpsSignalMode](#)

[SWS_Eth_00343]*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00168](#)

[If [Eth_SetPpsSignalMode](#) is called with [signalMode](#) set to `TRUE` and [EthCtrlPulsePerSecondConfig](#) is configured for the given [CtrlIdx](#), then the Ethernet Driver shall start the PPS signal generation in hardware.]

[SWS_Eth_00378]*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00168](#)

[If [Eth_SetPpsSignalMode](#) is called with [signalMode](#) set to `FALSE` and [EthCtrlPulsePerSecondConfig](#) is configured for the given [CtrlIdx](#), then the Ethernet Driver shall stop the PPS signal generation in hardware.]

[SWS_Eth_00379]*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00168](#)

[If [Eth_SetPpsSignalMode](#) is called and [EthCtrlPulsePerSecondConfig](#) is configured for the given [CtrlIdx](#) and the affected hardware has already reached the requested [signalMode](#) mode, then the Ethernet Driver shall ignore the mode request and return with `E_OK`.]

[SWS_Eth_00344]*Status:* DRAFT*Upstream requirements:* [SRS_Eth_00168](#)

[If [Eth_SetPpsSignalMode](#) is called and [EthCtrlPulsePerSecondConfig](#) is NOT configured for the given [CtrlIdx](#), then the Ethernet Driver shall return with `E_NOT_OK`.]

7.1.10 MACsec support

[SWS_Eth_CONSTR_00023] Availability of `EthCtrlMacSecConfig` is mandatory if `EthCtrlMacSecEnabled` is set to `TRUE`

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `EthCtrlMacSecEnabled` is set to `TRUE` for an Ethernet controller configuration (see `EthCtrlConfig`), then the `EthCtrlMacSecConfig` shall be available. Otherwise the configuration shall be rejected as invalid)]

[SWS_Eth_00418] Initialize Ethernet controller with MACsec support, where MACsec functionality is enabled

Upstream requirements: [FO_RS_MACsec_00004](#)

[At initialization of an Ethernet controller, the Eth driver shall initialize each Ethernet controller with the corresponding `EthCtrlConfig` configuration where `EthCtrlMacSecEnabled` is available and set to `TRUE`, and consider the following configuration parameter if available:

- initialize the Ethernet controller with all configured `EthCtrlMacSecBypassEtherTypes` to bypass MACsec handling for all Ethernet frames carrying the configured `EtherType`
- initialize the Ethernet controller with all configured `EthCtrlMacSecBypassVlans` to bypass MACsec handling for all Ethernet frames carrying the configured VLAN-ID
- initialize the Ethernet controller with all configured `EthCtrlMacSecBypassDestinationMacAddresses` to bypass MACsec handling for all Ethernet frames carrying the configured destination MAC address

]

[SWS_Eth_00419] Handling if `Eth_MacSecInitRxSc` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is available and set to `TRUE`

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecInitRxSc` is called and `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`, then the Eth driver shall configure a receive secure channel with the secure channel id given with `RxSci` at the corresponding Ethernet controller addressed with `CtrlIdx` and return with `E_OK`.]

[SWS_Eth_00420] Handling if `Eth_MacSecInitRxSc` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `FALSE` or the configuration parameter is not available

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecInitRxSc` is called and `EthCtrlMacSecEnabled` is set to `FALSE` or the configuration parameter is not available of the `EthCtrlConfig` identified with the given `CtrlIdx`, then the Eth driver shall return with `E_NOT_OK`.]

[SWS_Eth_00421] Handling if `Eth_MacSecUpdateSecY` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is available and set to `TRUE`

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecUpdateSecY` is called and `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`, then the Eth driver shall consider to update the Ethernet controller that corresponds to the given `CtrlIdx` with the MAC security entity configuration provided via `MACSecCfgPtr` and consider to establish a transmit secure channel with the transmission secure channel id given with `TxSci` and return with `E_OK`.]

[SWS_Eth_00422] Handling if attempts to update an Ethernet controller with a MAC security entity configuration and the establishment of a transmission secure channel has been finalized

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecUpdateSecY` was called and `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx` and the update process for the corresponding Ethernet controller with the MAC security entity configuration and the establishment of a transmit secure channel has been successfully finalized, then the Eth driver shall notify the EthIf by calling `EthIf_EthMacSecUpdateSecYNotification` with `Result` set to `E_OK`. Otherwise the Eth driver shall call `EthIf_EthMacSecUpdateSecYNotification` with `Result` set to `E_NOT_OK`.]

Note: A retry to update an Ethernet controller with an MAC security entity configuration and to establish a transmit secure channel is in responsibility of the north bound layer.

[SWS_Eth_00423] Handling if `Eth_MacSecUpdateSecY` is called and update of a MAC security entity configuration and the establishment of a transmission secure channel was already successfully finalized

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecUpdateSecY` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`

- the Ethernet controller was already successfully updated with a MAC security entity configuration given with `MACSecCfgPtr`
- the establishment of a transmission secure channel with the given `TxSci` was already successfully finalized

then the Eth driver shall consider to update the Ethernet controller that corresponds to the given `CtrlIdx` with the MAC security entity configuration provided via `MACSecCfgPtr` and return with `E_OK`.]

Note: According to the given MACsec configuration to update an Ethernet controller, it could be possible to disable MACsec handling completely at the given Ethernet controller and to release the established transmission secure channel.

[SWS_Eth_00424] Handling if `Eth_MacSecUpdateSecY` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `FALSE` or the configuration parameter is not available

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecUpdateSecY` is called and the `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `FALSE` or the configuration parameter is not available, then the Eth driver shall return with `E_NOT_OK`.]

[SWS_Eth_00425] Handling if `Eth_MacSecResetRxSc` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is available and set to `TRUE`

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecResetRxSc` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- a receive secure channel with the given `RxSci` exists at this Ethernet controller

then the Eth driver shall reset the receive secure channel given with `RxSci` at this Ethernet controller to default values and return with `E_OK`. Otherwise the Eth driver shall return with `E_NOT_OK`.]

[SWS_Eth_00426] Handling if `Eth_MacSecResetRxSc` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `FALSE` or the configuration parameter is not available

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecResetRxSc` is called and the `EthCtrlMacSecEnabled` is available and set to `FALSE` of the `EthCtrlConfig` identified with the given `CtrlIdx` or the configuration parameter is not available, then the Eth driver shall return with `E_NOT_OK`.]

[SWS_Eth_00427] Handling if `Eth_MacSecAddRxSa` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is available and set to `TRUE`

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecAddRxSa` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- a reception secure association can be added to the given association number (see [An](#)), i.e. the given association number is not occupied

then the Eth driver shall perform the following actions. Otherwise return with `E_NOT_OK`:

- create a reception secure association at the Ethernet controller that corresponds to the given `CtrlIdx` by considering the given association number (see [An](#)), the lowest acceptable packet number (see [LowestPn](#)), the short secure channel identifier (see [Ssci](#)) and the secured association key (available via [KeysPtr](#) and addressed with [SakKeyPtr](#))
- activate the created reception secure association, if given API parameter `Active` is set to `TRUE`, otherwise deactivate the created reception secure association
- return with `E_OK`.

]

[SWS_Eth_00428] Handling if `Eth_MacSecAddRxSa` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is available and set to `FALSE` or the configuration parameter is not available

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecAddRxSa` is called and `EthCtrlMacSecEnabled` is available and set to `FALSE` of the `EthCtrlConfig` identified with the given `CtrlIdx` or the configuration parameter is not available, then the Ethernet switch driver shall return with `E_NOT_OK`.]

[SWS_Eth_00429] Handling if attempts to enable / disable the reception secure association at an Ethernet controller has been finalized

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecAddRxSa` was called for an `EthCtrlConfig` identified with the given `CtrlIdx` and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is available and set to `TRUE` and the attempts to enable or disable the reception secure association at the corresponding Ethernet controller has been successfully finalized, then the Eth driver shall notify the EthIf by calling `EthIf_EthMacSecAddRxSaNotification` with `Result` set to `E_OK`. Otherwise the Eth driver shall call `EthIf_EthMacSecAddRxSaNotification` with `Result` set to `E_NOT_OK`.]

[SWS_Eth_00430] Handling if `Eth_MacSecUpdateRxSa` is called with `Active` is set to `TRUE`, the secured association is disabled and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `TRUE`

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecUpdateRxSa` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- the given API parameter `Active` is set to `TRUE`
- the reception secured association given with `An` exists and is disabled

then the Eth driver shall update the reception secure association at the Ethernet controller that corresponds to the given `CtrlIdx` by considering the lowest acceptable packet number (see [LowestPn](#)) and enable the reception secure association at this Ethernet controller and return with `E_OK`. Otherwise, if the addressed reception secured association does not exist, return with `E_NOT_OK`.]

[SWS_Eth_00431] Handling if `Eth_MacSecUpdateRxSa` is called with `Active` is set to `TRUE`, the secured association is enabled and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `TRUE`

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecUpdateRxSa` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- the given API parameter `Active` is set to `TRUE`
- the reception secured association given with `An` exists and has already been enabled

then the Eth driver shall update the reception secure association at the Ethernet controller that corresponds to the given to the given `CtrlIdx` by considering the lowest acceptable packet number (see [LowestPn](#)) and return with `E_OK`. Otherwise, if the addressed reception secured association does not exist, return with `E_NOT_OK`.]

[SWS_Eth_00432] Handling if `Eth_MacSecUpdateRxSa` is called with `Active` is set to `FALSE`, the secured association is enabled and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `TRUE`

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecUpdateRxSa` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`

- the given API parameter `Active` is set to `FALSE`
- the reception secured association given with `An` exists and is enabled

then the Eth driver shall disable the reception secure association at the Ethernet controller that corresponds to the given `CtrlIdx` by considering the lowest acceptable packet number (see `LowestPn`) and return with `E_OK`. Otherwise, if the addressed reception secured association does not exist, return with `E_NOT_OK`.]

[SWS_Eth_00433] Handling if `Eth_MacSecUpdateRxSa` is called with `Active` is set to `FALSE`, the secured association is disabled and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `TRUE`

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecUpdateRxSa` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- the given API parameter `Active` is set to `FALSE`
- the reception secured association given with `An` exists and is disabled

then the Eth driver shall ignore the API call and return with `E_OK`. Otherwise, if the addressed reception secured association does not exist, return with `E_NOT_OK`.]

[SWS_Eth_00434] Handling if `Eth_MacSecUpdateRxSa` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `FALSE` or the configuration parameter is not available

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecUpdateRxSa` is called and `EthCtrlMacSecEnabled` is available and set to `FALSE` of the `EthCtrlConfig` identified with the given `CtrlIdx` or the configuration parameter is not available, then the Ethernet driver shall return with `E_NOT_OK`.]

[SWS_Eth_00435] Handling if `Eth_MacSecDeleteRxSa` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is available and set to `TRUE`

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecDeleteRxSa` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- a reception secure association with the given `An` exists at this Ethernet controller

then the Eth driver shall remove the reception secure association given with `An` at the Ethernet controller given with `CtrlIdx` and return with `E_OK`. Otherwise, if addressed secured association is not available, the Eth driver shall return with `E_NOT_OK`.]

[SWS_Eth_00436] Handling if `Eth_MacSecAddTxSa` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is available and set to `TRUE`

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecAddTxSa` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- a transmission secure association can be added to the given association number (see [An](#)), i.e. the given association number is not occupied

then the Eth driver shall perform the following actions. Otherwise return with `E_NOT_OK`:

- create a transmission secure association at the Ethernet controller that corresponds to the given `CtrlIdx` by considering the given association number (see [An](#)), the next accepted packet number (see [NextPn](#)), the short secure channel identifier (see [Ssci](#)) and the secured association key (available via [KeysPtr](#) and addressed with [SakKeyPtr](#))
- activate the created transmission secure association, if given API parameter `Active` is set to `TRUE`, otherwise deactivate the created transmission secure association
- return with `E_OK`.

]

[SWS_Eth_00437] Handling if `Eth_MacSecAddTxSa` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is available and set to `FALSE` or the configuration parameter is not available

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecAddTxSa` is called and `EthCtrlMacSecEnabled` is available and set to `FALSE` of the `EthCtrlConfig` identified with the given `CtrlIdx` or the configuration parameter is not available, then the Eth driver shall return with `E_NOT_OK`.]

[SWS_Eth_00438] Handling if attempts to add a transmission secure association at an Ethernet controller and to enable or disable this secure association has been finalized

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If `Eth_MacSecAddTxSa` was called for an `EthCtrlConfig` identified with the given `CtrlIdx` and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is available and set to `TRUE` and the process to enable or disable the transmission secure association at the corresponding Ethernet controller has been successfully finalized,

then the Eth driver shall notify the EthIf by calling `EthIf_EthMacSecAddTxSaNotification` with `Result` set to `E_OK`. Otherwise the Eth driver shall call `EthIf_EthMacSecAddTxSaNotification` with `Result` set to `E_NOT_OK`.]

[SWS_Eth_00439] Handling if `Eth_MacSecUpdateTxSa` is called with `Active` is set to `TRUE`, the secured association is disabled and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `TRUE`

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecUpdateTxSa` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- the given API parameter `Active` is set to `TRUE`
- the transmission secured association given with `An` exists and is disabled

then the Eth driver shall perform the following actions:

- update the transmission secure association at the Ethernet controller that corresponds to the given `CtrlIdx` by considering the next packet number (see `NextPn`)
- enable the transmission secure association given with `An` at this Ethernet controller
- if another transmission secured association at this Ethernet controller is running, then disable this secured association
- return with `E_OK`.

Otherwise, if the addressed transmission secured association given with `An` does not exist, return with `E_NOT_OK`.]

[SWS_Eth_00440] Handling if `Eth_MacSecUpdateTxSa` is called with `Active` is set to `TRUE`, the secured association is enabled and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `TRUE`

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecUpdateTxSa` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- the given API parameter `Active` is set to `TRUE`
- the transmission secured association given with `An` exists and has already been enabled

then the Eth driver shall update the transmission secure association at the Ethernet controller that corresponds to the given `CtrlIdx` by considering the next packet number (see `NextPn`) and return with `E_OK`. Otherwise, if the addressed transmission secured association does not exist, return with `E_NOT_OK`.]

[SWS_Eth_00441] Handling if `Eth_MacSecUpdateTxSa` is called with `Active` is set to `FALSE`, the secured association is enabled and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `TRUE`

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecUpdateTxSa` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- the given API parameter `Active` is set to `FALSE`
- the transmission secured association given with `An` exists and is enabled

then the Eth driver shall disable the transmission secure association at the Ethernet controller that corresponds to the given `CtrlIdx` by considering the next packet number (see `NextPn`) and return with `E_OK`. Otherwise, if the addressed transmission secured association does not exist, return with `E_NOT_OK`.]

[SWS_Eth_00442] Handling if `Eth_MacSecUpdateTxSa` is called with `Active` is set to `FALSE`, the secured association is disabled and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `TRUE`

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecUpdateTxSa` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- the given API parameter `Active` is set to `FALSE`
- the transmission secured association given with `An` exists and is disabled

then the Eth driver shall ignore the API call and return with `E_OK`. Otherwise, if the addressed transmission secured association does not exist, return with `E_NOT_OK`.]

[SWS_Eth_00443] Handling if `Eth_MacSecUpdateTxSa` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `FALSE` or the configuration parameter is not available

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecUpdateTxSa` is called and `EthCtrlMacSecEnabled` is available and set to `FALSE` of the `EthCtrlConfig` identified with the given `CtrlIdx` or the

configuration parameter is not available, then the Ethernet driver shall return with E_NOT_OK.]

[SWS_Eth_00444] Handling if `Eth_MacSecDeleteTxSa` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is available and set to `TRUE`

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecDeleteTxSa` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- a transmission secure association with the given `An` exists at this Ethernet controller

then the Eth driver shall remove the transmission secure association given with `An` at the Ethernet controller given with `CtrlIdx` and return with `E_OK`. Otherwise, if addressed secured association is not available, the Eth driver shall return with `E_NOT_OK`.]

[SWS_Eth_00445] Handling if `Eth_MacSecGetTxSaNextPn` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is available and set to `TRUE`

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecGetTxSaNextPn` is called and the following conditions are valid:

- `EthCtrlMacSecEnabled` is available and set to `TRUE` of the `EthCtrlConfig` identified with the given `CtrlIdx`
- the transmission secured association given with `An` exists

then the Eth driver shall return the next package number from the MACsec entity (SecY) at the Ethernet controller that corresponds to the given `CtrlIdx` via the out parameter `NextPnPtr` and return with `E_OK`. Otherwise, if the addressed transmission secured association does not exist, return with `E_NOT_OK`.]

[SWS_Eth_00446] Handling if `Eth_MacSecGetTxSaNextPn` is called and `EthCtrlMacSecEnabled` of the addressed `EthCtrlConfig` is set to `FALSE` or the configuration parameter is not available

Status: DRAFT

Upstream requirements: `FO_RS_MACsec_00004`

[If `Eth_MacSecGetTxSaNextPn` is called and `EthCtrlMacSecEnabled` is available and set to `FALSE` of the `EthCtrlConfig` identified with the given `CtrlIdx` or the configuration parameter is not available, then the Ethernet driver shall return with `E_NOT_OK`.]

[SWS_Eth_00447] Handling if attempts to collect MACsec statistics at an Ethernet controller has been finalized

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If [Eth_MacSecGetMacSecStatistics](#) was called for an Ethernet controller where the corresponding [EthCtrlConfig](#) has [EthCtrlMacSecEnabled](#) set to `TRUE` and the collection of the MACsec statistics for this Ethernet controller has been successfully finalized, then the Eth driver shall perform the following actions:

- set all statistics value which are not available (e.g. not provided by the hardware) to the maximum value of the used data type (e.g. for uint64 set value to 0xFFFFFFFFFFFFFFFF)
- notify the EthIf by calling [EthIf_EthMacSecGetMacSecStatisticsNotification](#) with `Result` set to `E_OK`

Otherwise the Eth driver shall call [EthIf_EthMacSecGetMacSecStatisticsNotification](#) with `Result` set to `E_NOT_OK`.]

Note: AUTOSAR expects that hardware devices are used that comply to [16, IEEE 802.1AE-2018] and able to provide the statistics which are defined with `Mka_Stats_SecYType`. Even though in some cases may not all statistic values are available, therefore the Eth driver should provide the maximum value for not provided statistic values.

[SWS_Eth_00448] Handling if [Eth_MacSecSetControlledPortEnabled](#) is called and [EthCtrlMacSecEnabled](#) of the addressed [EthCtrlConfig](#) is set to `FALSE` or the configuration parameter is not available

Status: DRAFT

Upstream requirements: [FO_RS_MACsec_00004](#)

[If [Eth_MacSecSetControlledPortEnabled](#) is called and [EthCtrlMacSecEnabled](#) is available and set to `FALSE` of the [EthCtrlConfig](#) identified with the given [CtrlIdx](#) or the configuration parameter is not available, then the Eth driver shall return with `E_NOT_OK`.]

7.1.11 Configuration description

[SWS_Eth_00125] [The MCG shall read the ECU configuration description of the Ethernet Driver module(s). Ethernet Driver related configuration data is contained in the Ethernet Driver module configuration description.]

[SWS_Eth_00126] [The MCG shall ensure the consistency of the generated configuration data.]

Note: For more details regarding the initialization please refer to section [8.4.1](#).

An assignment of those configuration classes to configuration parameters can be found in chapter 10.

A detailed description of all Ethernet Driver related configuration parameters can be found in chapter 10 of this document.

7.2 Error Classification

Chapter [3, General Specification of Basic Software Modules] 7.2 “Error Handling” describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.2.1 Development Errors

[SWS_Eth_00016] Definition of development errors in module Eth [

Type of error	Related error code	Error value
Invalid controller index	ETH_E_INV_CTRL_IDX	0x01
Eth module or controller was not initialized	ETH_E_UNINIT	0x02
Invalid pointer in parameter list	ETH_E_PARAM_POINTER	0x03
Invalid parameter	ETH_E_INV_PARAM	0x04
Invalid mode	ETH_E_INV_MODE	0x05
Invalid clock unit index	ETH_E_INV_CLKUNIT_IDX	0x06
Clock adjustment in absolut value or rate/offset correction failed	ETH_E_CLOCK_ADJUSTMENT_FAILED	0x07
The size of the Ethernet frame exceed the available egress queue element size	ETH_E_EXCEED_EGRESS_QUEUE_ELEMENT	0x09
A requested hardware supported data transfer was rejected by hardware	ETH_E_HW_SUPPORTED_DATA_TRANSFER_REJECTED	0x0A
A rx handle id is not associated with an ingress queue element.	ETH_E_RX_HANDLE_ID_NOT_ASSOCIATED	0x0B

]

7.2.2 Runtime Errors

[SWS_Eth_91014] Definition of runtime errors in module Eth [

Type of error	Related error code	Error value
All egress queue elements are occupied	ETH_E_EGRESS_QUEUE_OCCUPIED	0x01
All ingress queues elements are occupied	ETH_E_INRESS_QUEUE_OCCUPIED	0x02
Failure or incorrect communication with the Ethernet Controller	ETH_E_COMMUNICATION	0x06





Type of error	Related error code	Error value
No egress queue for requested priority available	ETH_E_UNKNOWN_EGRESS_PRIORITY	0x08
A received Ethernet frame could not be enqueued in any ingress queue	ETH_E_NO_MATCHING_INGRESS_QUEUE_IDENTIFIED	0x0C

]

7.2.3 Production Errors

There are no production errors.

7.2.4 Extended Production Errors

Extended production errors are handled as events of the Diagnostic Event Manager. The event IDs are defined in the following tables, while the actual values are assigned externally by the configuration of the Diagnostic Event Manager, and are included in the module via Dem.h.

[SWS_Eth_00173] Ethernet Controller Access Failure.

Upstream requirements: [SRS_BSW_00458](#)

[

Diagnostic Event (Error Name)	ETH_E_ACCESS	
Description	Monitors the access to the Ethernet Controller.	
Failed condition	When access to the Ethernet Controller fails the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.	
Passed condition	When access to the Ethernet Controller succeeds the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.	

]

[SWS_Eth_00174] [

Error Name:	ETH_E_RX_FRAMES_LOST	
Short Description:	Ethernet Frames Lost.	
Long Description:	Monitors the loss of Ethernet frames during reception.	
Detection Criteria:	Fail	When lost frames are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
Secondary Parameters:	None.	





Time Required:	None.
Monitor Frequency	None.

[SWS_Eth_00219] CRC Failure.

Upstream requirements: [SRS_BSW_00458](#)

Diagnostic Event (Error Name)	ETH_E_CRC
Description	Monitors invalid Ethernet frames during reception.
Failed condition	When invalid frames are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
Passed condition	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.

[SWS_Eth_00220] Frame Size Underflow.

Upstream requirements: [SRS_BSW_00458](#)

Diagnostic Event (Error Name)	ETH_E_UNDERSIZEFRAME
Description	Monitors undersize Ethernet frames during reception.
Failed condition	When invalid frames are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
Passed condition	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.

[SWS_Eth_00221] Frame Size Overflow.

Upstream requirements: [SRS_BSW_00458](#)

Diagnostic Event (Error Name)	ETH_E_OVERSIZEFRAME
Description	Monitors oversize Ethernet frames during reception.
Failed condition	When invalid frames are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
Passed condition	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.

[SWS_Eth_00222] Frame Alignment Error.

Upstream requirements: [SRS_BSW_00458](#)

[

Diagnostic Event (Error Name)	ETH_E_ALIGNMENT
Description	Monitors oversize Ethernet frames during reception.
Failed condition	When invalid frames are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
Passed condition	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.

]

[SWS_Eth_00223] Single Frame Collision.

Upstream requirements: [SRS_BSW_00458](#)

[

Diagnostic Event (Error Name)	ETH_E_SINGLECOLLISION
Description	Monitors Ethernet single frame collision.
Failed condition	When frame collisions are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
Passed condition	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.

]

[SWS_Eth_00224] Multiple Frame Collision.

Upstream requirements: [SRS_BSW_00458](#)

[

Diagnostic Event (Error Name)	ETH_E_MULTIPLECOLLISION
Description	Monitors Ethernet multiple frame collision.
Failed condition	When frame collisions are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
Passed condition	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.

]

[SWS_Eth_00225] Late Frame Collision.

Upstream requirements: [SRS_BSW_00458](#)

[

Diagnostic Event (Error Name)	ETH_E_LATECOLLISION
Description	Monitors Ethernet late frame collision.
Failed condition	When frame collisions are detected the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.

▽



<i>Passed condition</i>	When Ethernet Controller is successfully initialized the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
--------------------------------	---

」

8 API specification

8.1 API Parameters Checking

[SWS_Eth_00394] CtrlIdx Parameter [The Eth driver APIs which has the parameter `CtrlIdx` shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETH_E_INV_CTRL_IDX` when the development error detection is enabled.]

[SWS_Eth_00395] ClkUnitIdx Parameter [The Eth driver APIs which has the parameter `ClkUnitIdx` shall check the parameter `ClkUnitIdx` for being valid. If the check fails, the function shall raise the development error `ETH_E_INV_CLKUNIT_IDX` when the development error detection is enabled.]

[SWS_Eth_00396] Pointer Parameter [The Eth driver APIs which has any pointer parameter shall check the pointer parameter for being valid. If the check fails, the function shall raise the development error `ETH_E_PARAM_POINTER` when the development error detection is enabled.]

8.2 Imported types

This chapter lists all types included from the following modules:

[SWS_Eth_00026] Definition of imported datatypes of module Eth

Upstream requirements: [SRS_Eth_00127](#)

[

Module	Header File	Imported Type
Comtype	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	ListElemStructType (draft)
	ComStack_Types.h	TimeStampQualType (draft)
	ComStack_Types.h	TimeStampType (draft)
	ComStack_Types.h	TimeTupleType (draft)
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Icu	Icu.h	Icu_ChannelType
Mka	Mka.h	Mka_ConfidentialityOffsetType (draft)
	Mka.h	Mka_MacSecConfigType (draft)
	Mka.h	Mka_SakKeyPtrType (draft)
	Mka.h	Mka_Stats_Rx_ScType (draft)
	Mka.h	Mka_Stats_Rx_SecYType (draft)
	Mka.h	Mka_Stats_SecYType (draft)

▽



Module	Header File	Imported Type
	Mka.h	Mka_Stats_Tx_ScType (draft)
	Mka.h	Mka_Stats_Tx_SecYType (draft)
	Mka.h	Mka_ValidateFramesType (draft)
Spi	Spi.h	Spi_ChannelType
	Spi.h	Spi_DataBufferType
	Spi.h	Spi_NumberOfDataType
	Spi.h	Spi_SequenceType
	Spi.h	Spi_StatusType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

8.3 Type definitions

8.3.1 Eth_ConfigType

[SWS_Eth_00156] Definition of datatype Eth_ConfigType [

Name	Eth_ConfigType
Kind	Structure
Description	Implementation specific structure of the post build configuration
Available via	Eth.h

8.3.2 Eth_ModeType

[SWS_Eth_91011] Definition of datatype Eth_ModeType [

Name	Eth_ModeType		
Kind	Enumeration		
Range	ETH_MODE_DOWN	0x00	disable the Ethernet Rx/Tx communication and set its corresponding hardware to a low-power sleep mode and initiate a sleep process, if the Ethernet hardware provides such a feature. E.g. request a sleep on data line for OA TC10 compatible Ethernet hardware
	ETH_MODE_ACTIVE	0x01	enable the Ethernet Rx/Tx communication and set its corresponding hardware to a power-on mode





	ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST	0x02	enable the Ethernet Rx/Tx communication, set its corresponding Ethernet hardware to a power-on mode and request an wake-up on the network, if the Ethernet hardware provides a wake-up feature. E.g. wake-up on data line for OA TC10 compatible Ethernet hardware
	ETH_MODE_ACTIVE_TX_OFFLINE	0x03	disable the Tx communication path. Please note, this is only used in EthIf to support silent communication (see COMM_SILENT_COMMUNICATION). In silent communication all transmission requests are rejected
Description	This is an generic type and used in the layers of the Ethernet communication stack (e.g. EthIf, Eth, EthSwt, EthTrcv) to enable and disable, respectively, the Ethernet communication channel and set the corresponding hardware (e.g. Ethernet controller, Ethernet Switch port, Ethernet transceiver) to a low-power sleep and power on mode, respectively. The type also supports to transfer a wake-up request from the services layer (ComM) to the communication drivers (EthTrcv). This could be used e.g. for Ethernet hardware that has the capability to wake-up and sleep on data line (see OA TC10)		
Available via	Eth_GeneralTypes.h		

8.3.3 Eth_StateType

[SWS_Eth_00159] Definition of datatype Eth_StateType

Name	Eth_StateType		
Kind	Enumeration		
Range	ETH_STATE_UNINIT	0x00	Driver is not yet configured
	ETH_STATE_INIT	0x01	Driver is configured
Description	Status supervision used for Development Error Detection. The state shall be available for debugging.		
Available via	Eth_GeneralTypes.h		

8.3.4 Eth_FrameType

[SWS_Eth_00160] Definition of datatype Eth_FrameType

Name	Eth_FrameType		
Kind	Type		
Derived from	uint16		
Description	This type defines the Ethernet frame type used in the Ethernet frame header		
Available via	Eth_GeneralTypes.h		

8.3.5 Eth_DataType

[SWS_Eth_00161] Definition of datatype Eth_DataType [

Name	Eth_DataType	
Kind	Type	
Derived from	Basetype	Variation
	uint16	8 or 16 bit CPU
	uint32	32 bit CPU
	uint8	8, 16 or 32 bit CPU
Description	This type defines the Ethernet data type used for data transmission. Its definition depends on the used CPU.	
Available via	Eth_GeneralTypes.h	

]

8.3.6 Eth_BufIdxType

[SWS_Eth_00175] Definition of datatype Eth_BufIdxType [

Name	Eth_BufIdxType		
Kind	Type		
Derived from	uint32		
Description	Ethernet buffer identifier type.		
Available via	Eth_GeneralTypes.h		

]

8.3.7 Eth_RxStatusType

[SWS_Eth_00162] Definition of datatype Eth_RxStatusType [

Name	Eth_RxStatusType		
Kind	Enumeration		
Range	ETH_RECEIVED	0x00	Ethernet frame has been received, no further frames available
	ETH_NOT_RECEIVED	0x01	Ethernet frame has not been received, no further frames available
	ETH_RECEIVED_MORE_DATA_AVAILABLE	0x02	Ethernet frame has been received, more frames are available
Description	Used as out parameter in Eth_Receive() indicates whether a frame has been received and if so, whether more frames are available or frames got lost.		
Available via	Eth_GeneralTypes.h		

]

8.3.8 Eth_FilterActionType

[SWS_Eth_00163] Definition of datatype Eth_FilterActionType [

Name	Eth_FilterActionType		
Kind	Enumeration		
Range	ETH_ADD_TO_FILTER	0x00	add the MAC address to the filter, meaning allow reception
	ETH_REMOVE_FROM_FILTER	0x01	remove the MAC address from the filter, meaning reception is blocked in the lower layer
Description	The Enumeration Type Eth_FilterActionType describes the action to be taken for the MAC address given in *PhysAddrPtr.		
Available via	Eth_GeneralTypes.h		

]

8.3.9 Eth_MacVlanType

[SWS_Eth_91001] Definition of datatype Eth_MacVlanType

Upstream requirements: [SRS_Eth_00121](#), [SRS_Eth_00072](#)

[

Name	Eth_MacVlanType	
Kind	Structure	
Elements	MacAddr	
	Type	Array of uint8
	Size	6
	Comment	Specifies the MAC address [0..255,0..255,0..255,0..255,0..255,0..255]
	VlanId	
	Type	uint16
	Comment	Specifies the VLAN address 0..65535
	SwitchPort	
	Type	uint32
	Comment	Specifies the ports of the switch as bit mask (0x00000001->Port0, 0x80000001->Port31+Port0)
Description	This type is used to read out addresses from the address resolution logic (ARL) table of the switch. typedef struct { uint8 MacAddr[6U]; uint16 VlanId; uint32 SwitchPort; } Eth_MacVlanType; In case of Macaddr contains a Multicast Address MacVlanType.SwitchPort shall be handled as Bitmask, each bit represents a Switch Port, Bit 0 represents EthSwichtPortIdx = 0 , Bit 1 represents EthSwichtPortIdx = 1 and so on. In case of Macaddr contains not a Multicast Address MacVlanType.SwitchPort shall be handled as a value representing the EthSwitchPortIdx.	
Available via	Eth_GeneralTypes.h	

]

8.3.10 Eth_CounterType

[SWS_Eth_91007] Definition of datatype Eth_CounterType [

Name	Eth_CounterType	
Kind	Structure	
Elements	DropPktBufOverrun	
	Type	uint32
	Comment	dropped packets due to buffer overrun
	DropPktCrc	
	Type	uint32
	Comment	dropped packets due to CRC errors
	UndersizePkt	
	Type	uint32
	Comment	number of undersize packets which were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. (see IETF RFC 1757)
	OversizePkt	
	Type	uint32
	Comment	number of oversize packets which are longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed. (see IETF RFC 1757)
	AlgnmtErr	
	Type	uint32
	Comment	number of alignment errors, i.e. packets which are received and are not an integral number of octets in length and do not pass the CRC.
	SqeTestErr	
	Type	uint32
	Comment	SQE test error according to IETF RFC1643 dot3StatsSQETestErrors
	DiscInbdPkt	
	Type	uint32
	Comment	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space. (see IETF RFC 2233 ifInDiscards)
	ErrInbdPkt	
	Type	uint32
	Comment	total number of erroneous inbound packets
	DiscOtbdPkt	
	Type	uint32
	Comment	The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. (see IETF RFC 2233 ifOutDiscards)
	ErrOtbdPkt	
	Type	uint32
	Comment	total number of erroneous outbound packets





	SnglCollPkt	
	Type	uint32
	Comment	Single collision frames: A count of successfully transmitted frames on a particular interface for which transmission is inhibited by exactly one collision. (see IETF RFC1643 dot3StatsSingleCollisionFrames)
	MultCollPkt	
	Type	uint32
	Comment	Multiple collision frames: A count of successfully transmitted frames on a particular interface for which transmission is inhibited by more than one collision. (see IETF RFC1643 dot3StatsMultipleCollisionFrames)
	DfrdPkt	
	Type	uint32
	Comment	Number of deferred transmission: A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy. (see IETF RFC1643 dot3StatsDeferred Transmissions)
	LatCollPkt	
	Type	uint32
	Comment	Number of late collisions: The number of times that a collision is detected on a particular interface later than 512 bit-times into the transmission of a packet. (see IETF RFC1643 dot3StatsLateCollisions)
	HwDepCtr0	
	Type	uint32
	Comment	hardware dependent counter value
	HwDepCtr1	
	Type	uint32
	Comment	hardware dependent counter value
	HwDepCtr2	
	Type	uint32
	Comment	hardware dependent counter value
	HwDepCtr3	
	Type	uint32
	Comment	hardware dependent counter value
	Description	Statistic counter for diagnostics.
	Available via	Eth_GeneralTypes.h

]

8.3.11 Eth_RxStatsType

[SWS_Eth_91002] Definition of datatype Eth_RxStatsType

Upstream requirements: [SRS_Eth_00127](#)

Name	Eth_RxStatsType	
Kind	Structure	
Elements	RxStatsDropEvents	
	Type	uint32
	Comment	The total number of events in which packets were dropped by the probe due to lack of resources. Also described in IETF RFC 2819 MIB etherStatsDropEvents.
	RxStatsOctets	
	Type	uint32
	Comment	The total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including FCS octets). Also described in IETF RFC 2819 MIB etherStatsOctets.
	RxStatsPkts	
	Type	uint32
	Comment	The total number of packets (including bad packets, broadcast packets, and multicast packets) received. Also described in IETF RFC 2819 MIB etherStatsPkts
	RxStatsBroadcastPkts	
	Type	uint32
	Comment	The total number of good packets received that were directed to the broadcast address. Also described in IETF RFC 2819 MIB etherStatsBroadcastPkts
	RxStatsMulticastPkts	
	Type	uint32
	Comment	The total number of good packets received that were directed to a multicast address. Also described in IETF RFC 2819 MIB etherStatsMulticastPkts.
	RxStatsCrcAlignErrors	
	Type	uint32
	Comment	The total number of packets received that had a length of between 64 and 1518 octets that had either a bad Frame Check Sequence (FCS) with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error). Also described in IETF RFC 2819 MIB etherStatsCRCAlignErrors
	RxStatsUndersizePkts	
	Type	uint32
	Comment	The total number of packets received that were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. Also described in IETF RFC 2819 MIB etherStatsUndersizePkts.
	RxStatsOversizePkts	
	Type	uint32





	Comment	The total number of packets received that were longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed. Also described in IETF RFC 2819 MIB etherStatsOversizePkts
	RxStatsFragments	
	Type	uint32
	Comment	The total number of packets received that were less than 64 octets in length (excluding framing bits but including FCS octets) and had either a bad Frame Check Sequence (FCS) with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error). Also described in IETF RFC 2819 MIB etherStatsFragments.
	RxStatsJabbers	
	Type	uint32
	Comment	The total number of packets received that were longer than 1518 octets, and had either a bad Frame Check Sequence (FCS) with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error). Also described in IETF RFC 2819 MIB etherStatsJabbers.
	RxStatsCollisions	
	Type	uint32
	Comment	The best estimate of the total number of collisions on this Ethernet segment. Also described in IETF RFC 2819 MIB etherStatsCollisions
	RxStatsPkts64Octets	
	Type	uint32
	Comment	The total number of packets (including bad packets) received that were 64 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts64Octets
	RxStatsPkts65to127Octets	
	Type	uint32
	Comment	The total number of packets (including bad packets) received that were between 65 and 127 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts65to127Octets
	RxStatsPkts128to255Octets	
	Type	uint32
	Comment	The total number of packets (including bad packets) received that were between 128 and 255 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts128to255Octets
	RxStatsPkts256to511Octets	
	Type	uint32
	Comment	The total number of packets (including bad packets) received that were between 256 and 511 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts256to511Octets
	RxStatsPkts512to1023Octets	
	Type	uint32
	Comment	The total number of packets (including bad packets) received that were between 512 and 1023 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts512to1023Octets
	RxStatsPkts1024to1518Octets	
	Type	uint32





	Comment	The total number of packets (including bad packets) received that were between 1024 and 1518 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts1024to1518Octets
	RxUnicastFrames	
	Type	uint32
	Comment	The number of subnetwork-unicast packets delivered to a higher-layer protocol. Also described in IETF RFC1213 MIB ifInUcastPkts
Description	Statistic counter for diagnostics.	
Available via	Eth_GeneralTypes.h	

8.3.12 Eth_TxStatsType

[SWS_Eth_91003] Definition of datatype Eth_TxStatsType

Upstream requirements: [SRS_Eth_00127](#)

Name	Eth_TxStatsType	
Kind	Structure	
Elements	TxNumberOfOctets	
	Type	uint32
	Comment	The total number of octets transmitted out of the interface, including framing characters. Also described in IETF RFC1213 MIB ifOutOctets.
	TxNUcastPkts	
	Type	uint32
	Comment	The total number of packets that higher-level protocols requested be transmitted to a non-unicast (i.e., a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent. Also described in IETF RFC1213 MIB ifOutNUcastPkts
	TxUniCastPkts	
	Type	uint32
	Comment	The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent. Also described in IETF RFC1213 MIB ifOutUcastPkts.
Description	Statistic counter for diagnostics.	
Available via	Eth_GeneralTypes.h	

8.3.13 Eth_TxErrorCounterValuesType

[SWS_Eth_91004] Definition of datatype Eth_TxErrorCounterValuesType

Upstream requirements: [SRS_Eth_00127](#)

Name	Eth_TxErrorCounterValuesType	
Kind	Structure	
Elements	TxDroppedNoErrorPkts	
	Type	uint32
	Comment	The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. Also described in IETF RFC1213 MIB ifOutDiscards
	TxDroppedErrorPkts	
	Type	uint32
	Comment	transmitted because of errors. Also described in IETF RFC1213 MIB ifOutErrors
	TxDeferredTrans	
	Type	uint32
	Comment	A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy. The count represented by an instance of this object does not include frames involved in collisions. Also described in IETF RFC1643 MIB dot3StatsDeferredTransmissions
	TxSingleCollision	
	Type	uint32
	Comment	A count of successfully transmitted frames on a particular interface for which transmission is inhibited by exactly one collision. A frame that is counted by an instance of this object is also counted by the corresponding instance of either the TxUniCastPkts and TxNUcastPkts and is not counted by the corresponding instance of the TxMultipleCollision object. Also described in IETF RFC1643 MIB dot3StatsSingleCollisionFrames
	TxMultipleCollision	
	Type	uint32
	Comment	A count of successfully transmitted frames on a particular interface for which transmission is inhibited by more than one collision. A frame that is counted by an instance of this object is also counted by the corresponding instance of either the TxUniCastPkts and TxNUcastPkts and is not counted by the corresponding instance of the TxSingleCollision object. Also described in IETF RFC1643 MIB dot3StatsMultipleCollisionFrames.
	TxLateCollision	
	Type	uint32





	Comment	The number of times that a collision is detected on a particular interface later than 512 bit-times into the transmission of a packet. Five hundred and twelve bit-times corresponds to 51.2 microseconds on a 10 Mbit/s system. A (late) collision included in a count represented by an instance of this object is also considered as a (generic) collision for purposes of other collision-related statistics. Also described in IETF RFC1643 MIB dot3StatsLateCollisions
	TxExcessiveCollision	
	Type	uint32
	Comment	A count of frames for which transmission on a particular interface fails due to excessive collisions. Also described in IETF RFC1643 MIB dot3 StatsExcessiveCollisions
Description	Statistic counters for diagnostics.	
Available via	Eth_GeneralTypes.h	

8.3.14 Eth_SpiStatusType

[SWS_Eth_91013] Definition of datatype Eth_SpiStatusType

Status: DRAFT

Upstream requirements: [SRS_Eth_00147](#), [SRS_Eth_00120](#)

Name	Eth_SpiStatusType (draft)	
Kind	Structure	
Elements	SpiStatusRegister	
	Type	uint32
	Comment	Bit mapped status defined by OA TC6 to notify following information: (Pos : description) 0x00: Transmit_Protocol_Error, 0x01: Transmit_Buffer_Overflow_Error, 0x02: Transmit_Buffer_Underflow_Error, 0x03: Receive_Buffer_Overflow_Error, 0x04: Loss_Framing_error, 0x05: Header_Error, 0x06: Reset_Complete, 0x07: PHY_Interrupt, 0x08: Transmit_Timestamp_Capture_Available_A, 0x09: Transmit_Timestamp_Capture_Available_B, 0x0A: Transmit_Timestamp_Capture_Available_C, 0x0B: Transmit_Frame_Check_Sequence_Error, 0x0C: Control_Data_Protection_Error, 0x0D - 0xFF: Reserved.
	Sync	
	Type	boolean
	Comment	Synchronization configuration as defined in the OA TC6. TRUE: MACPHY has been reset and is not configured. FALSE: MACPHY is configured.
	BufferStatusTxCredit	
	Type	uint8





	Comment	Contains the number of consecutive transmitted data chunks of Ethernet frame the SPI host can write without overflowing the MAC.
	BufferStatusRxCredit	
	Type	uint8
	Comment	Contains the number of additional received data chunks of Ethernet frame currently available for the SPI host to read.
Description	Returns the Spi status, errors and configuration state. Tags: atp.Status=draft	
Available via	Eth_GeneralTypes.h	

]

8.3.15 Eth_StreamStatisticCounterType

[SWS_Eth_91027] Definition of datatype Eth_StreamStatisticCounterType [

Name	Eth_StreamStatisticCounterType	
Kind	Structure	
Elements	BucketIdx	
	Type	uint8
	Comment	Bucket Index
	CounterValue	
	Type	uint32
	Comment	Bucket counter value
Description	Type for holding the bucket counter values for a stream.	
Available via	Eth_GeneralTypes.h	

]

8.4 Function definitions

This is a list of functions provided for upper layer modules.

8.4.1 Eth_Init

[SWS_Eth_00027] Definition of API function Eth_Init [

Service Name	Eth_Init
Syntax	<pre>void Eth_Init (const Eth_ConfigType* CfgPtr)</pre>
Service ID [hex]	0x01
Sync/Async	Synchronous
Reentrancy	Non Reentrant





Parameters (in)	CfgPtr	Points to the implementation specific structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initializes the Ethernet Driver	
Available via	Eth.h	

]

[SWS_Eth_00028] [The function shall store the access to the configuration structure for subsequent API calls.]

[SWS_Eth_00275] [The function shall for all configured Ethernet controllers in the current EthConfigSet:

- Disable Rx/Tx communication of all Ethernet controllers
- Clear pending Ethernet interrupts
- Configure all controller configuration parameters (e.g. interrupts, frame length, frame filter, ...)
- Configure all transmit / receive resources (e.g. buffer initialization)
- delete all pending transmit and receive requests.

]

Note: The implementation has to ensure that the control capabilities (e.g. MDIO) provided by an Ethernet controller which are used by other driver modules (e.g. Ethernet switch driver) are always available independent of the requested mode (ETH_MODE_DOWN or ETH_MODE_ACTIVE). Therefore the Ethernet driver may initialize the control capabilities within Eth_Init.

[SWS_Eth_00350]

Status: DRAFT

Upstream requirements: [SRS_BSW_00406](#)

[If the config parameter [EthPhcSupport](#) is set to TRUE, then the Ethernet driver shall check for all configured Ethernet controllers if [EthCtrlClks](#) are configured. If [EthCtrlClks](#) are configured, then the Ethernet driver shall initialize the Ethernet controller hardware clocks, set the initialization value to zero and start the hardware clock.]

[SWS_Eth_00029] [The function shall change the state of the component from ETH_STATE_UNINIT to ETH_STATE_INIT.]

[SWS_Eth_00039] [The function shall check the access to the Ethernet controller. If the check fails, the function shall raise the production error ETH_E_ACCESS.]

[SWS_Eth_00031] [[Eth_Init\(\)](#) shall be called during initialization.]

8.4.2 Eth_SetControllerMode

[SWS_Eth_91009] Definition of API function Eth_SetControllerMode [

Service Name	Eth_SetControllerMode	
Syntax	<pre>Std_ReturnType Eth_SetControllerMode (uint8 CtrlIdx, Eth_ModeType CtrlMode)</pre>	
Service ID [hex]	0x03	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
	CtrlMode	ETH_MODE_DOWN: Disable Rx/Tx communication of the controller ETH_MODE_ACTIVE: Enable Rx/Tx communication of the controller
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
Description	Enables / Disables Rx/Tx communication of the indexed controller	
Available via	Eth.h	

]

[SWS_Eth_00276] [The function shall put the controller in the specified mode given in the parameter 'CtrlMode':

- Upon mode ETH_MODE_DOWN the driver shall:
 - Disable Tx/Rx communication of the Ethernet controller
 - Reset all transmit and receive buffers (i.e. ignore all pending transmission and reception requests)
- Upon mode ETH_MODE_ACTIVE:
 - Enable all transmit and receive buffers
 - Activate Rx/Tx communication of the Ethernet controller

]

[SWS_Eth_00301] [If development error detection is enabled: the function shall check the parameter CtrlMode. If the given mode is other than ETH_MODE_ACTIVE or ETH_MODE_DOWN, the function shall raise the development error ETH_E_INV_MODE.]

[SWS_Eth_00168] [The function shall check the access to the Ethernet controller. If the check fails, the function shall raise the production error ETH_E_ACCESS and return E_NOT_OK.]

[SWS_Eth_00045] [Eth_Init() shall be called before Eth_SetControllerMode().]

8.4.3 Eth_GetControllerMode

[SWS_Eth_91010] Definition of API function Eth_GetControllerMode [

Service Name	Eth_GetControllerMode	
Syntax	<pre>Std_ReturnType Eth_GetControllerMode (uint8 CtrlIdx, Eth_ModeType* CtrlModePtr)</pre>	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
Parameters (inout)	None	
Parameters (out)	CtrlModePtr	ETH_MODE_DOWN: the Rx/Tx communication of the controller is disabled ETH_MODE_ACTIVE: the Rx/Tx communication of the controller is enabled
Return value	Std_ReturnType	E_OK: success E_NOT_OK: controller mode could not be obtained
Description	Obtains the communication state of the indexed controller	
Available via	Eth.h	

]

[SWS_Eth_00277] [The function shall read the current Rx/Tx communication state of the indexed controller.]

[SWS_Eth_00051] [Eth_Init() shall be called before Eth_GetControllerMode().]

8.4.4 Eth_GetPhysAddr

[SWS_Eth_00052] Definition of API function Eth_GetPhysAddr [

Service Name	Eth_GetPhysAddr	
Syntax	<pre>void Eth_GetPhysAddr (uint8 CtrlIdx, uint8* PhysAddrPtr)</pre>	
Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
Parameters (inout)	None	
Parameters (out)	PhysAddrPtr	Physical source address (MAC address) in network byte order.
Return value	void	None
Description	Obtains the physical source address used by the indexed controller	
Available via	Eth.h	

]

[SWS_Eth_00053] [The function shall read the source address used by the indexed controller.]

[SWS_Eth_00057] [*Eth_Init()* shall be called before *Eth_GetPhysAddr()*.]

8.4.5 Eth_SetPhysAddr

[SWS_Eth_00151] Definition of API function **Eth_SetPhysAddr** [

Service Name	Eth_SetPhysAddr	
Syntax	<pre>void Eth_SetPhysAddr (uint8 CtrlIdx, const uint8* PhysAddrPtr)</pre>	
Service ID [hex]	0x13	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver.
	PhysAddrPtr	Pointer to memory containing the physical source address (MAC address) in network byte order.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Sets the physical source address used by the indexed controller	
Available via	Eth.h	

]

[SWS_Eth_00139] [The function shall update the source address used by the indexed controller.]

[SWS_Eth_00143] [*Eth_Init()* shall be called before *Eth_SetPhysAddr()*.]

8.4.6 Eth_UpdatePhysAddrFilter

[SWS_Eth_00152] Definition of API function **Eth_UpdatePhysAddrFilter** [

Service Name	Eth_UpdatePhysAddrFilter	
Syntax	<pre>Std_ReturnType Eth_UpdatePhysAddrFilter (uint8 CtrlIdx, const uint8* PhysAddrPtr, uint16 VlanId, Eth_FilterActionType Action)</pre>	
Service ID [hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same CtrlIdx, reentrant for different	





Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
	PhysAddrPtr	Pointer to memory containing the physical destination address (MAC address) in network byte order. This is the multicast destination address of the layer 2 packet.
	VlanId	Identifier of virtual LAN
	Action	Add or remove the address from the controllers filter.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: filter was successfully changed
		E_NOT_OK: filter could not be changed
Description	Update the physical source address to/from the indexed controller filter. If the controller is not capable to do the filtering, the software has to do this.	
Available via	Eth.h	

]

[SWS_Eth_00150] [The function shall update the physical address and VlanId receive filter of the indexed controller. If VlanId is set to 0xFFFF, the Vlan filtering is not used.]

[SWS_Eth_00245] [The Ethernet driver module shall receive a frame when the destination Address match the PhyAddrPtr and VlanId passed here. (e.g matching can be done via hash table or simple pattern matching)]

Note: Underlying HW mechanism can be used if available. Otherwise the Ethernet driver needs to do this by software.

[SWS_Eth_00246] [If the matching is positive, the upper layer shall be notified by calling RxIndication() callback.

If the matching is negative, the frame shall be discarded.]

[SWS_Eth_00167] [*Eth_Init()* shall be called before *Eth_UpdatePhysAddrFilter()*.]

[SWS_Eth_00144] [If the physical address (MAC address) is set to FF:FF:FF:FF:FF:FF, this shall completely open the filter for the given VlanId.]

[SWS_Eth_00146] [If this API is used and the hardware does not support filtering, promiscuous mode shall be enabled during initialization.]

[SWS_Eth_00147] [If the physical address (MAC address) is set to 00:00:00:00:00:00, this shall reduce the filter to the controllers unique unicast MAC address for the given VlanId and end promiscuous mode if it was turned on.]

8.4.7 Eth_WriteMii

[SWS_Eth_00058] Definition of API function Eth_WriteMii [

Service Name	Eth_WriteMii	
Syntax	<pre>Std_ReturnType Eth_WriteMii (uint8 CtrlIdx, uint8 TrcvIdx, uint8 RegIdx, uint16 RegVal)</pre>	
Service ID [hex]	0xa0	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs, non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Index of the controller.
	TrcvIdx	Index of the transceiver on the SMI.
	RegIdx	Index of the transceiver register on the SMI.
	RegVal	Value to be written into the indexed register.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Service accepted
		E_NOT_OK: Service denied
Description	Writes a transceiver register.	
Available via	Eth.h	

]

[SWS_Eth_00286]

Status: DRAFT

[The function shall check the communication with the Ethernet Controller. If the check fails, the function shall report the runtime error code ETH_E_COMMUNICATION and return E_NOT_OK.]

[SWS_Eth_00278]

Status: DRAFT

Upstream requirements: [SRS_Eth_00148](#)

[The function shall write the specified transceiver register through the MII according to Clause 22 [17] for the indexed controller.]

[SWS_Eth_00273]

Upstream requirements: [SRS_Eth_00148](#)

[If Clause 45 registers need to be written via this access mechanism, the API shall use the register 13 and 14 to access them as explicitly specified by the annex 22D [17].]

[SWS_Eth_00287]

Status: DRAFT

Upstream requirements: [SRS_Eth_00147](#), [SRS_Eth_00146](#)

[If EthCtrlEnableSpiInterface is TRUE, the function shall process the write request as described in the TC6 [11].]

[SWS_Eth_00390] Sync write

Status: DRAFT

Upstream requirements: [SRS_Eth_00148](#), [SRS_Eth_00147](#), [SRS_Eth_00146](#)

「The function shall write the MDIO synchronously and return E_OK when the access finished.」

[SWS_Eth_00062] 「The function shall be pre compile time configurable On/Off by the configuration parameter: EthCtrlEnableMii [[ECUC_Eth_00012](#)].」

[SWS_Eth_00063] 「Eth_Init() shall be called before Eth_WriteMii().」

8.4.8 Eth_ReadMii

[SWS_Eth_00064] Definition of API function Eth_ReadMii 「

Service Name	Eth_ReadMii	
Syntax	<pre>Std_ReturnType Eth_ReadMii (uint8 CtrlIdx, uint8 TrcvIdx, uint8 RegIdx, uint16* RegValPtr)</pre>	
Service ID [hex]	0xa1	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs, non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Index of the controller.
	TrcvIdx	Index of the transceiver on the SMI.
	RegIdx	Index of the transceiver register on the SMI.
Parameters (inout)	None	
Parameters (out)	RegValPtr	Filled with the register content of the indexed register.
Return value	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
Description	Reads a transceiver register.	
Available via	Eth.h	

」

[SWS_Eth_00289]

Status: DRAFT

Upstream requirements: [SRS_Eth_00148](#), [SRS_Eth_00146](#)

「The function shall check the communication with the Ethernet Controller. If the check fails, the function shall report the runtime error code ETH_E_COMMUNICATION and return E_NOT_OK.」

[SWS_Eth_00279]

Status: DRAFT

Upstream requirements: [SRS_Eth_00148](#), [SRS_Eth_00146](#)

「The function shall read the specified transceiver register through the MII according to Clause 22 [[17](#)] for the indexed controller.」

[SWS_Eth_00274]

Upstream requirements: [SRS_Eth_00148](#)

[If Clause 45 registers need to be read via this access mechanism, the API shall use the register 13 and 14 to access them as explicitly specified by the annex 22D [\[17\]](#).]

[SWS_Eth_00290]

Status: DRAFT

Upstream requirements: [SRS_Eth_00148](#), [SRS_Eth_00146](#), [SRS_Eth_00147](#)

[If EthCtrlEnableSpilInterface is TRUE, the function shall process the read request as described in the TC6 [\[11\]](#).]

[SWS_Eth_00391] Sync read

Status: DRAFT

Upstream requirements: [SRS_Eth_00148](#), [SRS_Eth_00147](#), [SRS_Eth_00146](#)

[The function shall read the MDIO synchronously and return E_OK when the access finished.]

[SWS_Eth_00069] [The function shall be pre compile time configurable On/Off by the configuration parameter: EthCtrlEnableMii [\[ECUC_Eth_00012\]](#).]

[SWS_Eth_00070] [Eth_Init() shall be called before Eth_ReadMii().]

8.4.9 Eth_GetCounterValues

[SWS_Eth_00226] Definition of API function Eth_GetCounterValues

Upstream requirements: [SRS_Eth_00127](#)

[

Service Name	Eth_GetCounterValues	
Syntax	<pre>Std_ReturnType Eth_GetCounterValues (uint8 CtrlIdx, Eth_CounterType* CounterPtr)</pre>	
Service ID [hex]	0x14	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
Parameters (inout)	None	
Parameters (out)	CounterPtr	counter values according to IETF RFC 1757, RFC 1643 and RFC 2233.
Return value	Std_ReturnType	E_OK: success E_NOT_OK: counter values read failure
Description	Reads a list with drop counter values of the corresponding controller. The meaning of these values is described at Eth_CounterType.	
Available via	Eth.h	

]

[SWS_Eth_00227] [The function shall read a list of values from the indexed controller.]

[SWS_Eth_00231] [The function `Eth_GetCounterValues` shall be pre compile time configurable On/Off by the configuration parameter: `EthGetCounterValuesApi` [ECUC_Eth_00035].]

[SWS_Eth_00232] [`Eth_Init()` shall be called before `Eth_GetCounterValues()`.]

8.4.10 Eth_GetRxStats

[SWS_Eth_00233] Definition of API function `Eth_GetRxStats`

Upstream requirements: [SRS_Eth_00127](#)

[

Service Name	Eth_GetRxStats	
Syntax	<pre>Std_ReturnType Eth_GetRxStats (uint8 CtrlIdx, Eth_RxStatsType* RxStats)</pre>	
Service ID [hex]	0x15	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
Parameters (inout)	None	
Parameters (out)	RxStats	List of values according to IETF RFC 2819 (Remote Network Monitoring Management Information Base)
Return value	Std_ReturnType	E_OK: success E_NOT_OK: drop counter could not be obtained
Description	Returns the following list according to IETF RFC2819, where the maximal possible value shall denote an invalid value, e.g. if this counter is not available: 1. etherStatsDropEvents 2. etherStatsOctets 3. etherStatsPkts 4. etherStatsBroadcastPkts 5. etherStatsMulticastPkts 6. etherStatsCrcAlignErrors 7. etherStatsUndersizePkts 8. etherStatsOversizePkts 9. etherStatsFragments 10. etherStatsJabbers 11. etherStatsCollisions 12. etherStatsPkts64Octets 13. etherStatsPkts65to127Octets 14. etherStatsPkts128to255Octets 15. etherStatsPkts256to511Octets 16. etherStatsPkts512to1023Octets 17. etherStatsPkts1024to1518Octets	
Available via	Eth.h	

]

[SWS_Eth_00234] [The function shall read a list of values from the indexed controller according to [18](#).]

[SWS_Eth_00238] [The function `Eth_GetRxStats` shall be pre compile time configurable On/Off by the configuration parameter: `EthGetRxStatsApi`.]

8.4.11 Eth_GetTxStats

[SWS_Eth_91005] Definition of API function Eth_GetTxStats

Upstream requirements: [SRS_Eth_00127](#)

[

Service Name	Eth_GetTxStats	
Syntax	<pre>Std_ReturnType Eth_GetTxStats (uint8 CtrlIdx, Eth_TxStatsType* TxStats)</pre>	
Service ID [hex]	0x1c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
Parameters (inout)	None	
Parameters (out)	TxStats	List of values to read statistic values for transmission.
Return value	Std_ReturnType	E_OK: success, E_NOTOK: Tx-statistics could not be obtained
Description	Returns the list of Transmission Statistics out of IETF RFC1213 defined with Eth_TxStatsType, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.	
Available via	Eth.h	

]

[SWS_Eth_00251]

Upstream requirements: [SRS_Eth_00053](#)

[The function Eth_GetTxStats shall be pre compile time configurable On/Off by the configuration parameter: EthGetTxStatsApi [[ECUC_Eth_00060](#)].]

8.4.12 Eth_GetTxErrorCounterValues

[SWS_Eth_91006] Definition of API function Eth_GetTxErrorCounterValues

Upstream requirements: [SRS_Eth_00127](#)

[

Service Name	Eth_GetTxErrorCounterValues	
Syntax	<pre>Std_ReturnType Eth_GetTxErrorCounterValues (uint8 CtrlIdx, Eth_TxErrorCounterValuesType* TxErrorCounterValues)</pre>	
Service ID [hex]	0x1d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
Parameters (inout)	None	





Parameters (out)	TxErroCounterValues	List of values to read statistic error counter values for transmission.
Return value	Std_ReturnType	E_OK: success, E_NOTOK: Tx-statistics could not be obtained
Description	Returns the list of Transmission Error Counters out of IETF RFC1213 and RFC1643 defined with Eth_TxErroCounterValuesType, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.	
Available via	Eth.h	

]

[SWS_Eth_00255]

Upstream requirements: [SRS_Eth_00053](#)

[The function Eth_GetTxErrorCounterValues shall be pre compile time configurable On/Off by the configuration parameter: EthGetTxErrorCounterValuesApi [[ECUC_Eth_00061](#)].]

8.4.13 Eth_GetSpiStatus

[SWS_Eth_91012] Definition of API function Eth_GetSpiStatus

Status: DRAFT

Upstream requirements: [SRS_Eth_00147](#), [SRS_Eth_00120](#)

[

Service Name	Eth_GetSpiStatus (draft)	
Syntax	Std_ReturnType Eth_GetSpiStatus (uint8 CtrlIdx, Eth_SpiStatusType* SpiStatusType)	
Service ID [hex]	0x1E	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Ethernet Driver
Parameters (inout)	None	
Parameters (out)	SpiStatusType	MACPHY status
Return value	Std_ReturnType	E_OK: success, E_NOT_OK: Status could not be obtained
Description	Returns the status defined by OA TC6 to identify if an error can occurred at the SPI interface. Tags: atp.Status=draft	
Available via	Eth.h	

]

[SWS_Eth_00295]

Status: DRAFT

Upstream requirements: [SRS_Eth_00146](#), [SRS_Eth_00147](#)

[The function Eth_GetSpiStatus shall be pre compile time configurable On/Off by the configuration parameter: EthCtrlEnableSpiInterface[[ECUC_Eth_00073](#)].]

8.4.14 Eth_GetCurrentTimeTuple

[SWS_Eth_91017] Definition of API function Eth_GetCurrentTimeTuple

Status: DRAFT

Upstream requirements: [SRS_Eth_00175](#)

Service Name	Eth_GetCurrentTimeTuple (draft)	
Syntax	Std_ReturnType Eth_GetCurrentTimeTuple (uint8 CtrlIdx, uint8 ClkUnitIdx, TimeTupleType* currentTimeTuplePtr)	
Service ID [hex]	0x21	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of Ethernet Controller within the context of the Ethernet driver which owns the clock unit
	ClkUnitIdx	Index of the Clock Unit within the context of the Ethernet driver to provide the time tuple
Parameters (inout)	None	
Parameters (out)	currentTimeTuplePtr	Current time tuple with the • value of the clock used for timestamping • value of adjustable PHC
Return value	Std_ReturnType	E_OK: PHC successfully set E_NOT_OK: PHC could not be set
Description	Reads the time tuple of the current time of the timestamp clock and the current time of the PHC in an atomic operation. If no PHC is supported, the PHC value will be a copy of the timestamp clock value. Tags: atp.Status=draft	
Available via	Eth.h	

[SWS_Eth_00349]

Status: DRAFT

Upstream requirements: [SRS_BSW_00171](#)

[The function shall be pre compile time configurable On/Off by the configuration parameter: [EthPhcSupport](#).]

In case the Com-Stack is distributed across several partitions, the Ethernet stack could reside in a different partition than the StbM module calling Eth_GetCurrentTimeTuple (via EthIf_GetCurrentTimeTuple) API, means the call of Eth_GetCurrentTimeTuple could happen in another partition.

[SWS_Eth_00351]

Status: DRAFT

Upstream requirements: [SRS_BSW_00459](#)

[The Eth module shall apply appropriate mechanisms to allow calls of Eth_GetCurrentTimeTuple API from other partitions than its main function, e.g. by providing an Eth satellite.]

8.4.15 Eth_SetPhcTime

[SWS_Eth_91018] Definition of API function Eth_SetPhcTime

Status: DRAFT

Upstream requirements: [SRS_Eth_00167](#)

[

Service Name	Eth_SetPhcTime (draft)	
Syntax	<pre>Std_ReturnType Eth_SetPhcTime (uint8 CtrlIdx, uint8 ClkUnitIdx, const TimeStampType* timeStampPtr)</pre>	
Service ID [hex]	0x22	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of Ethernet Controller within the context of the Ethernet driver which owns the clock unit
	ClkUnitIdx	Index off the Clock Unit within the context of the Ethernet driver which is addressed to be adjusted
Parameters (inout)	None	
Parameters (out)	timeStampPtr	Time value, by which the PHC is requested to be updated.
Return value	Std_ReturnType	E_OK: PHC successfully set E_NOT_OK: PHC could not be set
Description	Sets the absolute time of the PHC. Tags: atp.Status=draft	
Available via	Eth.h	

]

[SWS_Eth_00355]

Status: DRAFT

Upstream requirements: [SRS_BSW_00171](#)

[The function shall be pre compile time configurable On/Off by the configuration parameter: [EthPhcSupport](#).]

In case the Com-Stack is distributed across several partitions, the Ethernet stack could reside in a different partition than the StbM module calling Eth_SetPhcTime (via EthIf_SetPhcTime) API, means the call of Eth_SetPhcTime could happen in another partition.

[SWS_Eth_00357]

Status: DRAFT

Upstream requirements: [SRS_BSW_00459](#)

[The Eth module shall apply appropriate mechanisms to allow calls of Eth_SetPhcTime API from other partitions than its main function, e.g. by providing an Eth satellite.]

8.4.16 Eth_SetPhcCorrection

[SWS_Eth_91019] Definition of API function Eth_SetPhcCorrection

Status: DRAFT

Upstream requirements: [SRS_Eth_00167](#)

Service Name	Eth_SetPhcCorrection (draft)	
Syntax	Std_ReturnType Eth_SetPhcCorrection (uint8 CtrlIdx, uint8 ClkUnitIdx, sint32 rateDeviation, sint32 offset)	
Service ID [hex]	0x23	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of Ethernet Controller within the context of the Ethernet Interface which owns the clock unit
	ClkUnitIdx	Index of the Clock Unit within the context of the Ethernet Interface to provide the time tuple
	rateDeviation	Rate deviation (resolution: 2 ⁻³²), by which the PHC is requested to be corrected
	offset	Time offset, by which the PHC is requested to be updated. Unit: Nanoseconds.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: PHC successfully set E_NOT_OK:PHC could not be set
Description	Sets PHC parameters to adapt rate and offset of the PHC. Tags: atp.Status=draft	
Available via	Eth.h	

[SWS_Eth_00372]

Status: DRAFT

Upstream requirements: [SRS_BSW_00171](#)

[The function shall be pre compile time configurable On/Off by the configuration parameter: [EthPhcSupport](#).]

In case the Com-Stack is distributed across several partitions, the Ethernet stack could reside in a different partition than the StbM module calling Eth_SetPhcCorrection (via EthIf_SetPhcCorrection) API, means the call of Eth_SetPhcCorrection could happen in another partition.

[SWS_Eth_00387]

Status: DRAFT

Upstream requirements: [SRS_BSW_00459](#)

[The Eth module shall apply appropriate mechanisms to allow calls of Eth_SetPhcCorrection API from other partitions than its main function, e.g. by providing an Eth satellite.]

8.4.17 Eth_GetPhcTime

[SWS_Eth_91020] Definition of API function Eth_GetPhcTime

Status: DRAFT

Upstream requirements: [SRS_Eth_00175](#)

[

Service Name	Eth_GetPhcTime (draft)	
Syntax	<pre>Std_ReturnType Eth_GetPhcTime (uint8 CtrlIdx, uint8 ClkUnitIdx, TimeStampQualType* timeQualPtr, TimeStampType* timeStampPtr)</pre>	
Service ID [hex]	0x24	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	CtrlIdx	Index of Ethernet Controller within the context of the Ethernet driver which owns the clock unit
	ClkUnitIdx	Index of the Clock Unit within the context of the Ethernet driver to provide the time tuple
	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: PHC value successfully retrieved E_NOT_OK: PHC value could not be retrieved
	Returns the current time value out of the HW registers of the PHC. Tags: atp.Status=draft	
Available via	Eth.h	

]

[SWS_Eth_00363]

Status: DRAFT

Upstream requirements: [SRS_BSW_00171](#)

[The function shall be pre compile time configurable On/Off by the configuration parameter: [EthPhcSupport](#).]

[SWS_Eth_00364]

Status: DRAFT

Upstream requirements: [SRS_BSW_00101](#)

[*Eth_Init()* shall be called before *Eth_GetPhcTime()*.]

In case the Com-Stack is distributed across several partitions, the Ethernet stack could reside in a different partition than the StbM module calling *Eth_GetPhcTime* (via *EthIf_GetPhcTime*) API, means the call of *Eth_GetPhcTime* could happen in another partition.

[SWS_Eth_00365]

Status: DRAFT

Upstream requirements: [SRS_BSW_00459](#)

[The Eth module shall apply appropriate mechanisms to allow calls of Eth_GetPhcTime API from other partitions than its main function, e.g. by providing an Eth satellite.]

8.4.18 Eth_SetPpsSignalMode

[SWS_Eth_91021] Definition of API function Eth_SetPpsSignalMode

Status: DRAFT

Upstream requirements: [SRS_Eth_00176](#)

Service Name	Eth_SetPpsSignalMode (draft)	
Syntax	<pre>Std_ReturnType Eth_SetPpsSignalMode (uint8 CtrlIdx, uint8 ClkUnitIdx, boolean signalMode)</pre>	
Service ID [hex]	0x25	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of Ethernet Controller within the context of the Ethernet driver which owns the clock unit
	ClkUnitIdx	Index off the Clock Unit within the context of the Ethernet driver to drive the PPS signal generation
	signalMode	TRUE: PPS signal generation is enabled FALSE: PPS signal generation is disabled
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: PHC successfully set E_NOT_OK: PHC could not be set
Description	Enables/disables the generation of a PPS signal Tags: atp.Status=draft	
Available via	Eth.h	

[SWS_Eth_00368]

Status: DRAFT

Upstream requirements: [SRS_BSW_00171](#)

[The function shall be pre compile time configurable On/Off by the configuration parameter: [EthPhcSupport](#).]

8.4.19 Eth_EnableEgressTimeStamp

[SWS_Eth_00186] Definition of API function Eth_EnableEgressTimeStamp [

Service Name	Eth_EnableEgressTimeStamp	
Syntax	<pre>void Eth_EnableEgressTimeStamp (uint8 CtrlIdx, Eth_BufIdxType BufIdx)</pre>	
Service ID [hex]	0x17	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the addresses controller.
	BufIdx	Index of the message buffer, where Application expects egress time stamping
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Activates egress time stamping on a dedicated message object. Some HW does store once the egress time stamp marker and some HW needs it always before transmission. There will be no "disable" functionality, due to the fact, that the message type is always "time stamped" by network design.	
Available via	Eth.h	

]

[SWS_Eth_00211] [The function shall be pre compile time configurable On/Off by the configuration parameter: EthGlobalTimeSupport [ECUC_Eth_00037].]

[SWS_Eth_00189] [Eth_Init() shall be called before Eth_EnableEgressTimeStamp().]

8.4.20 Eth_GetEgressTimeStamp

[SWS_Eth_00190] Definition of API function Eth_GetEgressTimeStamp [

Service Name	Eth_GetEgressTimeStamp	
Syntax	<pre>Std_ReturnType Eth_GetEgressTimeStamp (uint8 CtrlIdx, Eth_BufIdxType BufIdx, TimeStampQualType* timeQualPtr, TimeStampType* timeStampPtr)</pre>	
Service ID [hex]	0x18	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the addresses controller.
	BufIdx	Index of the message buffer, where Application expects egress time stamping
Parameters (inout)	None	
Parameters (out)	timeQualPtr	quality of HW time stamp, e.g. based on current drift





	timeStampPtr	current time stamp
Return value	Std_ReturnType	E_OK: success E_NOT_OK: failed to read time stamp.
Description	Reads back the egress time stamp on a dedicated message object. It must be called within the TxConfirmation() function.	
Available via	Eth.h	

]

[SWS_Eth_00212] [The function shall be pre compile time configurable On/Off by the configuration parameter: EthGlobalTimeSupport [ECUC_Eth_00037].]

[SWS_Eth_00194] [Eth_Init() shall be called before Eth_GetEgressTimeStamp().]

8.4.21 Eth_GetIngressTimeStamp

[SWS_Eth_00195] Definition of API function Eth_GetIngressTimeStamp

Status: OBSOLETE

[

Service Name	Eth_GetIngressTimeStamp (obsolete)	
Syntax	<pre>Std_ReturnType Eth_GetIngressTimeStamp (uint8 CtrlIdx, const Eth_DataType* DataPtr, TimeStampQualType* timeQualPtr, TimeStampType* timeStampPtr)</pre>	
Service ID [hex]	0x19	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the addresses controller.
	DataPtr	Pointer to the message buffer, where Application expects ingress time stamping
Parameters (inout)	None	
Parameters (out)	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Return value	Std_ReturnType	E_OK: success E_NOT_OK: failed to read time stamp.
Description	Reads back the ingress time stamp on a dedicated message object. It must be called within the RxIndication() function. Tags: atp.Status=obsolete	
Available via	Eth.h	

]

[SWS_Eth_00213]

Status: OBSOLETE

[The function shall be pre compile time configurable On/Off by the configuration parameter: EthGlobalTimeSupport [ECUC_Eth_00037].]

[SWS_Eth_00199]

Status: OBSOLETE

[Eth_Init() shall be called before Eth_GetIngressTimeStamp().]

8.4.22 Eth_ProvideTxBuffer

[SWS_Eth_00077] Definition of API function Eth_ProvideTxBuffer [

Service Name	Eth_ProvideTxBuffer	
Syntax	<pre> BufReq_ReturnType Eth_ProvideTxBuffer (uint8 CtrlIdx, uint8 Priority, Eth_BufIdxType* BufIdxPtr, uint8** BufPtr, uint16* LenBytePtr) </pre>	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
	Priority	Frame priority for transmit buffer queue selection
Parameters (inout)	LenBytePtr	In: desired length in bytes, out: granted length in bytes
Parameters (out)	BufIdxPtr	Index to the granted buffer resource. To be used for subsequent requests
	BufPtr	Pointer to the granted buffer
Return value	BufReq_ReturnType	BUFREQ_OK: success BUFREQ_E_NOT_OK: request not accepted. BUFREQ_E_BUSY: all buffers in use BUFREQ_E_OVFL: requested buffer too large
Description	Provides access to a transmit buffer of the queue related to the specified priority	
Available via	Eth.h	

]

[SWS_Eth_00078] [The function shall provide a transmit buffer resource. The Ethernet Driver shall lock the buffer until it receives a subsequent call of Eth_Transmit service with the buffer index returned in the BufIdxPtr parameter.]

[SWS_Eth_00414] Value range of the returned buffer index

Status: DRAFT

Upstream requirements: [SRS_Eth_00188](#)

[The returned buffer index value of type [Eth_BufIdxType](#) shall be greater then $2^{16}-1$. The value range for the buffer index shall be:

- 0x00 01 00 00 ... 0xFF FF FF FF: valid
- 0x00 00 00 00 ... 0x00 00 FF FF: reserved for TxHandleId of [Eth_ImmediateTransmit](#)

]

Note: Constraining the buffer index is needed, since `TxHandleId` of `Eth_ImmediateTransmit` used for direct data provision (used as PDU-ID) and `BufIdxPtr` of `Eth_ProvideTxBuffer` used for indirect data provision could overlap. EthIf need an unambiguous id (non-overlapping value range) that corresponds to a transmission request, to identify the affected transmission request for transmission confirmation via `EthIf_TxConfirmation`.

[SWS_Eth_00280] [All locked transmit buffers shall be released if the Rx/Tx communication of the indexed controller is disabled via `Eth_SetControllerMode`.]

[SWS_Eth_00079] [If a buffer requested with `Eth_ProvideTxBuffer` that is larger than the available buffer length, the buffer shall not be locked but return the available length and `BUFREQ_E_OVFL`.]

[SWS_Eth_00080] [If all available buffers are in use the component shall return `BUFREQ_E_BUSY`.]

[SWS_Eth_00086] [`Eth_Init()` shall be called before `Eth_ProvideTxBuffer()`.]

8.4.23 Eth_Transmit

[SWS_Eth_00087] Definition of API function Eth_Transmit [

Service Name	Eth_Transmit	
Syntax	<pre>Std_ReturnType Eth_Transmit (uint8 CtrlIdx, Eth_BufIdxType BufIdx, Eth_FrameType FrameType , boolean TxConfirmation, uint16 LenByte, const uint8* PhysAddrPtr)</pre>	
Service ID [hex]	0x0A	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different buffer indexes and Ctrl indexes	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
	BufIdx	Index of the buffer resource
	FrameType	Ethernet frame type
	TxConfirmation	Activates transmission confirmation
	LenByte	Data length in byte
	PhysAddrPtr	Physical target address (MAC address) in network byte order
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: transmission failed
Description	Triggers transmission of a previously filled transmit buffer	
Available via	Eth.h	

]

[SWS_Eth_00088] [The function shall build the Ethernet header with the given physical target address (MAC address) and trigger the transmission of a previously filled transmit buffer.]

After transmission, the driver needs to release the allocated buffer. It is up to the implementation when the actual buffer release shall occur, e.g. within the context of the `Eth_TxConfirmation`, the `Eth_MainFunction`, or during the next `Eth_ProvideTxBuffer`.

[SWS_Eth_00281] [All pending transmit buffers shall be released if the Rx/Tx communication of the indexed controller is disabled via `Eth_SetControllerMode`.]

[SWS_Eth_00092] [If development error detection is enabled: the function shall check the parameter `BufIdx` for being valid. If the check fails, the function shall raise the development error `ETH_E_INV_PARAM`.]

[SWS_Eth_00129] [`Eth_Transmit` shall check the controller mode for being active (`ETH_MODE_ACTIVE`). If the check fails, the function shall directly return `E_NOT_OK`.]

[SWS_Eth_00094] [`Eth_ProvideTxBuffer()` shall be called before `Eth_Transmit`.]

8.4.24 Eth_Receive

[SWS_Eth_00095] Definition of API function `Eth_Receive` [

Service Name	Eth_Receive	
Syntax	<pre>void Eth_Receive (uint8 CtrlIdx, uint8 QueueIdx, Eth_RxStatusType* RxStatusPtr)</pre>	
Service ID [hex]	0x0B	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different queues. Non Reentrant for the same queue.	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
	QueueIdx	Specifies the related queue
Parameters (inout)	None	
Parameters (out)	RxStatusPtr	Indicates whether a frame has been received and if so, whether more frames are available for the related queue.
Return value	None	
Description	Receive a frame from the related queue.	
Available via	Eth.h	

]

[SWS_Eth_00132] [`Eth_Receive()` shall check the controller mode for being active (`ETH_MODE_ACTIVE`). If the check fails, the function shall return directly.]

[SWS_Eth_00099] [`Eth_Init()` shall be called before `Eth_Receive()`.]

8.4.25 Eth_ImmediateTransmit

[SWS_Eth_91022] Definition of API function Eth_ImmediateTransmit

Status: DRAFT

Upstream requirements: [SRS_Eth_00173](#)

Service Name	Eth_ImmediateTransmit (draft)	
Syntax	Std_ReturnType Eth_ImmediateTransmit (uint8 CtrlIdx, Eth_BufIdxType TxHandleId, uint8 Priority, ListElemStructType* HeaderListPtr, uint8* PayloadPtr, uint16 PayloadLength)	
Service ID [hex]	0x26	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different Tx handle ids and Ctrl indexes	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
	TxHandleId	Unique transmit handle id provided by the Ethernet Interface, to identify the transmission request per physical Ethernet controller
	Priority	Ethernet frame VLAN-priority
	HeaderListPtr	Pointer to first Ethernet frame header of a single linked list.
	PayloadPtr	Pointer to the payload of the Ethernet frame
	PayloadLength	Length of the payload
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Transmit request has been accepted. E_NOT_OK:Transmit request has been rejected.
Description	Request transmission of an Ethernet frame, where each upper layer a header part as element of a single linked list. All headers together with the payload form an entire Ethernet frame Tags: atp.Status=draft	
Available via	EthIf.h	

8.4.26 Eth_ReleaseRxBuffer

[SWS_Eth_91023] Definition of API function Eth_ReleaseRxBuffer

Status: DRAFT

Upstream requirements: [SRS_Eth_00172](#)

[		
Service Name	Eth_ReleaseRxBuffer (draft)	
Syntax	<pre>void Eth_ReleaseRxBuffer (uint8 CtrlIdx, Eth_BufIdxType RxHandleId)</pre>	





Service ID [hex]	0x27	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different Rx handle ids and Ctrl indexes	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
	RxHandleId	Unique receive handle id provided by the Ethernet Driver in a previous call of EthIf_RxIndication, to identify the ingress queue element per physical Ethernet controller
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indication from the upper layer to release the reception buffer (ingress queue element) of the given physical Ethernet controller. Tags: atp.Status=draft	
Available via	EthIf.h	

]

8.4.27 Eth_TxConfirmation

[SWS_Eth_00100] Definition of API function Eth_TxConfirmation [

Service Name	Eth_TxConfirmation	
Syntax	<pre>void Eth_TxConfirmation (uint8 CtrlIdx)</pre>	
Service ID [hex]	0x0C	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Driver
Parameters (inout)	None	
Parameters (out)	None	
Return value	void	None
Description	Triggers frame transmission confirmation	
Available via	Eth.h	

]

[SWS_Eth_00101] [The function shall check all filled transmit buffers for successful transmission. The function issues transmit confirmation for each transmitted frame using the callback function EthIf_TxConfirmation if requested by the previous call of Eth_Transmit service.]

[SWS_Eth_00102] [If transmission confirmation was enabled by a previous call to Eth_Transmit function the function shall release the buffer resource.]

[SWS_Eth_00134] [Eth_TxConfirmation shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function shall return directly.]

[SWS_Eth_00105] [Eth_Init() shall be called before Eth_TxConfirmation.]

8.4.28 Eth_GetVersionInfo

[SWS_Eth_00106] Definition of API function Eth_GetVersionInfo [

Service Name	Eth_GetVersionInfo	
Syntax	<pre>void Eth_GetVersionInfo (Std_VersionInfoType* VersionInfoPtr)</pre>	
Service ID [hex]	0x0D	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	VersionInfoPtr	Version information of this module
Return value	void	None
Description	Returns the version information of this module	
Available via	Eth.h	

]

8.4.29 Eth_ReadMmd

[SWS_Eth_91025] Definition of API function Eth_ReadMmd [

Service Name	Eth_ReadMmd	
Syntax	<pre>Std_ReturnType Eth_ReadMmd (uint8 CtrlIdx, uint8 TrcvIdx, uint8 Mmd, uint16 RegIdx, uint16* RegValPtr)</pre>	
Service ID [hex]	0xa3	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the controller.
	TrcvIdx	Index of the transceiver on the SMI.
	Mmd	MDIO Manageable Device.
	RegIdx	Index of the transceiver register on the SMI.
Parameters (inout)	None	
Parameters (out)	RegValPtr	Filled with the register content of the indexed register
Return value	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
Description	Reads a transceiver register using Clause45 access if supported by hardware or implements a Clause45 access using Clause 22 operations	
Available via	Eth.h	

]

[SWS_Eth_00397] Read Mmd

Status: DRAFT

[The function shall read the specified transceiver register through the MII of the indexed controller.]

[SWS_Eth_00398] Read Mmd

Status: DRAFT

[The function shall be pre compile time configurable On/Off by the configuration parameter: EthCtrlEnableMmd.]

8.4.30 Eth_WriteMmd

[SWS_Eth_91026] Definition of API function Eth_WriteMmd [

Service Name	Eth_WriteMmd	
Syntax	<pre>Std_ReturnType Eth_WriteMmd (uint8 CtrlIdx, uint8 TrcvIdx, uint8 Mmd, uint16 RegIdx, uint16 RegVal)</pre>	
Service ID [hex]	0xa2	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the controller.
	TrcvIdx	Index of the transceiver on the SMI.
	Mmd	MDIO Manageable Device.
	RegIdx	Index of the transceiver register on the SMI.
	RegVal	Value to be written to the given address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
Description	Writes a transceiver register using Clause 45 access or implements a Clause45 access using Clause 22 operations	
Available via	Eth.h	

]

[SWS_Eth_00399] Write Mmd

Status: DRAFT

[The function shall write the specified transceiver register through the MII of the indexed controller.]

[SWS_Eth_00400] Write Mmd

Status: DRAFT

[The function shall be pre compile time configurable On/Off by the configuration parameter: EthCtrlEnableMmd.]

8.4.31 Eth_MacSecUpdateSecY

[SWS_Eth_91028] Definition of API function Eth_MacSecUpdateSecY

Status: DRAFT

[

Service Name	Eth_MacSecUpdateSecY (draft)	
Syntax	<pre>Std_ReturnType Eth_MacSecUpdateSecY (uint8 CtrlIdx, const Mka_MacSecConfigType* MACSecCfgPtr, uint64 TxSci)</pre>	
Service ID [hex]	0x2a	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of Ethernet Driver
	MACSecCfgPtr	Pointer to the structure to configure a MACsec Entity (SecY)
	TxSci	Secure Channel Identifier for the MACsec's Transmission Secure channel.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has been rejected
	Requests the Ethernet controller to update the SecY/PAC at this Ethernet controller with the provided parameters. A Transmission Secure Channel with the provided SCI shall be configured during the first call. A pointer to a MACsec Basic Parameters Configuration file shall be provided to create the Secure Channel. Tags: atp.Status=draft	
Available via	Eth.h	

]

8.4.32 Eth_MacSecInitRxSc

[SWS_Eth_91029] Definition of API function Eth_MacSecInitRxSc [

Service Name	Eth_MacSecInitRxSc	
Syntax	<pre>Std_ReturnType Eth_MacSecInitRxSc (uint8 CtrlIdx, uint64 RxSci)</pre>	
Service ID [hex]	0x2b	





Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of Ethernet Driver
	RxSci	Secure Channel Identifier for the MACsec's Reception Secure channel.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has been rejected
Description	Requests the Ethernet Controller Driver to configure a Reception Secure Channel for the given Reception Secure Channel Identifier.	
Available via	Eth.h	

8.4.33 Eth_MacSecResetRxSc

[SWS_Eth_91030] Definition of API function Eth_MacSecResetRxSc

Status: DRAFT

Service Name	Eth_MacSecResetRxSc (draft)	
Syntax	<pre>Std_ReturnType Eth_MacSecResetRxSc (uint8 CtrlIdx, uint64 RxSci)</pre>	
Service ID [hex]	0x2c	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of Ethernet Driver
	RxSci	Secure Channel Identifier for the MACsec's Reception Secure channel.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has been rejected
Description	Requests the Ethernet Controller Driver to reset to MACsec default values of the Reception Secure Channel for the given Reception Secure Channel Identifier. Tags: atp.Status=draft	
Available via	Eth.h	

8.4.34 Eth_MacSecAddTxSa

[SWS_Eth_91031] Definition of API function Eth_MacSecAddTxSa

Status: DRAFT

Service Name	Eth_MacSecAddTxSa (draft)	
Syntax	<pre>Std_ReturnType Eth_MacSecAddTxSa (uint8 CtrlIdx, uint8 An, uint64 NextPn, uint32 Ssci, const Mka_SakKeyPtrType* KeysPtr, boolean Active)</pre>	
Service ID [hex]	0x2d	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of Ethernet Driver
	An	Association Number to use in the MACsec's transmission secure association
	NextPn	Next accepted Packet Number in the MACsec's transmission secure association
	Ssci	Short Secure Channel Identifier used in the MACsec's transmission secure association
	KeysPtr	Pointer to the SAKs Key (and needed Key information) to use in the MACsec's transmission secure association
	Active	Boolean to enable/disable the MACsec's transmission secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has been rejected
Description	Requests the Ethernet Controller Driver to create a Transmission Secure Association in the Controller. The Short Secure Channel Identifier is included to support XPN configurations. Tags: atp.Status=draft	
Available via	Eth.h	

8.4.35 Eth_MacSecUpdateTxSa

[SWS_Eth_91032] Definition of API function Eth_MacSecUpdateTxSa

Status: DRAFT

[

Service Name	Eth_MacSecUpdateTxSa (draft)	
Syntax	<pre>Std_ReturnType Eth_MacSecUpdateTxSa (uint8 CtrlIdx, uint8 An, uint64 NextPn, boolean Active)</pre>	
Service ID [hex]	0x2e	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of Ethernet Driver
	An	Association Number to use in the MACsec's transmission secure association
	NextPn	Next accepted Packet Number in the MACsec's transmission secure association
	Active	Boolean to enable/disable the MACsec's transmission secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has been rejected
Description	Requests the Ethernet Controller Driver to update the Transmission Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. Tags: atp.Status=draft	
Available via	Eth.h	

]

8.4.36 Eth_MacSecDeleteTxSa

[SWS_Eth_91033] Definition of API function Eth_MacSecDeleteTxSa

Status: DRAFT

[

Service Name	Eth_MacSecDeleteTxSa (draft)	
Syntax	<pre>Std_ReturnType Eth_MacSecDeleteTxSa (uint8 CtrlIdx, uint8 An)</pre>	
Service ID [hex]	0x2f	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for the same CtrlIdx	





Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of Ethernet Driver
	An	Association Number to use in the MACsec's transmission secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The request has been accepted
		E_NOT_OK: The request has been rejected
Description	Request the Ethernet Transceiver Driver to remove the Transmission Secure Association identified by the provided Association Number. Tags: atp.Status=draft	
Available via	Eth.h	

8.4.37 Eth_MacSecAddRxSa

[SWS_Eth_91034] Definition of API function Eth_MacSecAddRxSa

Status: DRAFT

Service Name	Eth_MacSecAddRxSa (draft)	
Syntax	<pre>Std_ReturnType Eth_MacSecAddRxSa (uint8 CtrlIdx, uint8 An, uint64 LowestPn, uint32 Ssci, const Mka_SakKeyPtrType* KeysPtr, boolean Active)</pre>	
Service ID [hex]	0x30	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of Ethernet Driver
	An	Association Number to use in the MACsec's reception secure association
	LowestPn	Next accepted Packet Number in the MACsec's reception secure association
	Ssci	Short Secure Channel Identifier used in the MACsec's reception secure association
	KeysPtr	Pointer to the SAKs Key (and needed Key information) to use in the MACsec's reception secure association
	Active	Boolean to enable/disable the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The request has been accepted
		E_NOT_OK: The request has been rejected





Description	Requests the Ethernet Controller Driver to create a Reception Secure Association in the Controller. The Short Secure Channel Identifier is included to support XPN configurations. Tags: atp.Status=draft
Available via	Eth.h

]

8.4.38 Eth_MacSecUpdateRxSa

[SWS_Eth_91035] Definition of API function Eth_MacSecUpdateRxSa

Status: DRAFT

[

Service Name	Eth_MacSecUpdateRxSa (draft)	
Syntax	<pre>Std_ReturnType Eth_MacSecUpdateRxSa (uint8 CtrlIdx, uint8 An, uint64 LowestPn, boolean Active)</pre>	
Service ID [hex]	0x31	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of Ethernet Driver
	An	Association Number to use in the MACsec's reception secure association
	LowestPn	Next accepted Packet Number in the MACsec's reception secure association
	Active	Boolean to enable/disable the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has been rejected
Description	Requests the Ethernet Controller Driver to update the Reception Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. Tags: atp.Status=draft	
Available via	Eth.h	

]

8.4.39 Eth_MacSecDeleteRxSa

[SWS_Eth_91036] Definition of API function Eth_MacSecDeleteRxSa

Status: DRAFT

[

Service Name	Eth_MacSecDeleteRxSa (draft)	
Syntax	<pre>Std_ReturnType Eth_MacSecDeleteRxSa (uint8 CtrlIdx, uint8 An)</pre>	
Service ID [hex]	0x32	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of Ethernet Driver
	An	Association Number to use in the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has been rejected
Description	Request the Ethernet Transceiver Driver to remove the Reception Secure Association identified by the provided Association Number. Tags: atp.Status=draft	
Available via	Eth.h	

]

8.4.40 Eth_MacSecGetTxSaNextPn

[SWS_Eth_91037] Definition of API function Eth_MacSecGetTxSaNextPn

Status: DRAFT

[

Service Name	Eth_MacSecGetTxSaNextPn (draft)	
Syntax	<pre>Std_ReturnType Eth_MacSecGetTxSaNextPn (uint8 CtrlIdx, uint8 An, uint64* NextPnPtr)</pre>	
Service ID [hex]	0x33	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of Ethernet Driver
	An	Association Number to use in the MACsec's transmission secure association
Parameters (inout)	None	

▽



Parameters (out)	NextPnPtr	Pointer to the Next Packet Number read out from the MACsec Entity (SecY)
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has been rejected
Description	Request the Ethernet Transceiver Driver to return the Packet Number that is used for the next packet in the given Transmission Secure Association. Tags: atp.Status=draft	
Available via	Eth.h	

]

8.4.41 Eth_MacSecSetControlledPortEnabled

[SWS_Eth_91038] Definition of API function Eth_MacSecSetControlledPortEnabled

Status: DRAFT

[

Service Name	Eth_MacSecSetControlledPortEnabled (draft)	
Syntax	Std_ReturnType Eth_MacSecSetControlledPortEnabled (uint8 CtrlIdx, boolean ControlledPortEnabled)	
Service ID [hex]	0x34	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of Ethernet Driver
	ControlledPortEnabled	Boolean to activate the Controlled Port of the PAE
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has been rejected
Description	Requests to set the Controlled Port enabled parameter of a PAE. Tags: atp.Status=draft	
Available via	Eth.h	

]

8.4.42 Eth_MacSecGetMacSecStatistics

[SWS_Eth_91039] Definition of API function Eth_MacSecGetMacSecStatistics

Status: DRAFT

[

Service Name	Eth_MacSecGetMacSecStatistics (draft)	
Syntax	<pre>Std_ReturnType Eth_MacSecGetMacSecStatistics (uint8 CtrlIdx, Mka_Stats_SecYType* MacSecStatsPtr)</pre>	
Service ID [hex]	0x35	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of Ethernet Driver
Parameters (inout)	None	
Parameters (out)	MacSecStatsPtr	Pointer to a structure including the MACsec statistics of an MKA participant
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has been rejected
Description	Request the Ethernet Driver to provide MACsec statistics. Tags: atp.Status=draft	
Available via	Eth.h	

]

8.5 Callback notifications

This chapter lists all functions provided by the Ethernet controller module to lower layer modules. The lower layer module of Eth module is the SPI module. The SPI module, which is part of the MCAL, may used to exchange data between the microcontroller and an external Ethernet controller (i.e. MACPHY [11]).

8.6 Scheduled functions

8.6.1 Eth_MainFunction

[SWS_Eth_00171] Definition of scheduled function Eth_MainFunction [

Service Name	Eth_MainFunction
Syntax	<pre>void Eth_MainFunction (void)</pre>
Service ID [hex]	0x20





Description	The function checks for controller errors and lost frames. Used for polling state changes. Calls EthIf_CtrlModelIndication when the controller mode changed.
Available via	SchM_Eth.h

]

[SWS_Eth_00169] [The function shall check for lost frames. If the check fails, the function shall raise the extended production error event ETH_E_RX_FRAMES_LOST.]

[SWS_Eth_00172] [The function shall check for controller errors (e.g. CRC errors). If the check fails, the function shall raise the extended production error event as defined in section Extended Production Errors (e.g. ETH_E_CRC).]

[SWS_Eth_00240] [Used for polling state changes. Calls EthIf_CtrlModelIndication when the controller mode changed.]

8.7 Expected interfaces

This chapter lists all interfaces required from other modules.

8.7.1 Mandatory Interfaces

This chapter defines all interfaces required to fulfill the core functionality of the module.

[SWS_Eth_00119] Definition of mandatory interfaces required by module Eth [

API Function	Header File	Description
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ((Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType) == STANDARD_REPORTING)
EthIf_CtrlModelIndication	EthIf.h	Called asynchronously when mode has been read out. Triggered by previous <EthDrv>_SetController Mode call. Can directly be called within the trigger functions.
EthIf_GetVersionInfo	EthIf.h	Returns the version information of this module
EthIf_MainFunctionRx	SchM_EthIf.h	The function checks for new received frames and issues reception indications in polling mode.
EthIf_MainFunctionTx	SchM_EthIf.h	The function issues transmission confirmations in polling mode. It checks also for transceiver state changes.
EthIf_RxIndication	EthIf.h	Receive indication of an Ethernet frame which was received by the indexed controller
EthIf_TxConfirmation	EthIf.h	Confirms frame transmission by the indexed controller





API Function	Header File	Description
SchM_Enter_Eth	SchM_<Mip>.h	Invokes the SchM_Enter function to enter a module local exclusive area.
SchM_Exit_Eth	SchM_<Mip>.h	Invokes the SchM_Exit function to exit an exclusive area.

8.7.2 Optional Interfaces

This chapter defines all interfaces required to fulfill an optional functionality of the module.

[SWS_Eth_00120] Definition of optional interfaces requested by module Eth [

API Function	Header File	Description
Det_ReportError	Det.h	Service to report development errors.
EthIf_EthMacSecAddRxSaNotification (draft)	EthIf.h	Callback to notify that the Ethernet controller has finished attempts to create a Reception Secure Association Tags: atp.Status=draft
EthIf_EthMacSecAddTxSaNotification (draft)	EthIf.h	Callback to notify that the Ethernet controller has finished attempts to create a Transmission Secure Association. Tags: atp.Status=draft
EthIf_EthMacSecGetMacSecStatistics Notification (draft)	EthIf.h	Callback to get the MACsec statistics of an Ethernet Controller, which was triggered by a previous call of Eth_MacSecGetMacSecStatistics Tags: atp.Status=draft
EthIf_EthMacSecUpdateSecYNotification (draft)	EthIf.h	Callback to notify that the Ethernet controller has finished attempts to update SecY with provided parameters in a previous call of Eth_MacSecUpdateSecY. Tags: atp.Status=draft
EthIf_SwtEthRxFinishedIndication	EthIf.h	Indication for a finished receive process for a specific Ethernet frame, which results in providing the management information retrieved during the corresponding EthRxProcessFrame call.
EthIf_SwtEthRxProcessFrame	EthIf.h	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in the corresponding EthRxFinished Indication call.
EthIf_SwtEthTxAdaptBufferLength	EthIf.h	Modifies the buffer length to be able to insert management information.
EthIf_SwtEthTxFinishedIndication	EthIf.h	Indication for a finished transmit process for a specific Ethernet frame.
EthIf_SwtEthTxPrepareFrame	EthIf.h	Prepares the Ethernet frame for common Ethernet communication (frame shall be handled by switch according to the common address resolution behavior) and stores the information for processing of the corresponding TxFinishedIndication call.
EthIf_SwtEthTxProcessFrame	EthIf.h	Function inserts management information into the Ethernet frame.
Icu_DisableNotification	Icu.h	This function disables the notification of a channel.





API Function	Header File	Description
Icu_EnableNotification	Icu.h	This function enables the notification on the given channel.
Spi_GetStatus	Spi.h	Service returns the SPI Handler/Driver software module status.
Spi_ReadIB	Spi.h	Service for reading synchronously one or more data from an IB SPI Handler/Driver Channel specified by parameter.
Spi_SetupEB	Spi.h	Service to setup the buffers and the length of data for the EB SPI Handler/Driver Channel specified.
Spi_SyncTransmit	Spi.h	Service to transmit data on the SPI bus
Spi_WriteIB	Spi.h	Service for writing one or more data to an IB SPI Handler/Driver Channel specified by parameter.

└

8.7.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

Terms and definitions:

- **Reentrant:** interface is expected to be reentrant
- **Don't care:** reentrancy of interface not relevant for this module (in general it is in this case not reentrant).

8.7.3.1 Eth_<IngressQueueHandlerFunction>

[SWS_Eth_91024] Definition of configurable interface Eth_<IngressQueueHandlerFunction>(void)

Status: DRAFT

Upstream requirements: [SRS_Eth_00174](#)

┌

Service Name	Eth_<IngressQueueHandlerFunction>(void) (draft)
Syntax	void Eth_<IngressQueueHandlerFunction>(void) (void)
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	None



△

Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Function to handle a specific ingress queue Tags: atp.Status=draft
Available via	

」

9 Sequence diagrams

The usage of the Ethernet Driver is depicted in the sequence diagrams of the Ethernet Interface.

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module Eth.

Chapter 10.3 specifies published information of the module Eth.

10.1 How to read this chapter

For details refer to [3] Chapter 10.1 *“Introduction to configuration specification”*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 1 and Chapter 8.

[SWS_Eth_00257] [The Ethernet Driver module shall reject configurations with partition mappings which are not supported by the implementation.]

[SWS_Eth_00258] [If the driver manages several Ethernet controllers and if a subset of these controllers share peripheral resources or are somehow coupled (E.g. Communication control can only be done globally for all controllers), Ethernet driver shall emulate independent controllers to the upper layers. The coordination (E.g. Communication control) has to be done by the upper layer modules.]

[SWS_Eth_00296]

Status: DRAFT

Upstream requirements: [SRS_BSW_00159](#)

[The code configuration of the Eth module is Ethernet controller specific. If the Ethernet controller is sited on-chip, the code generation tool for the Eth module is microcontroller specific. If the Ethernet controller is an external device (i.e. MACPHY), the generation tool must not be microcontroller specific.]

10.2.1 Eth

[ECUC_Eth_00038] Definition of EcucModuleDef Eth [

Module Name	Eth
Description	Configuration of the Eth (Ethernet Driver) module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	Multiplicity	Dependency
EthConfigSet	1	This container contains the configuration parameters and sub containers of the AUTOSAR Eth module.
EthGeneral	1	General configuration of Ethernet Driver module

10.2.2 EthConfigSet

[ECUC_Eth_00015] Definition of EcucParamConfContainerDef EthConfigSet [

Container Name	EthConfigSet
Parent Container	Eth
Description	This container contains the configuration parameters and sub containers of the AUTOSAR Eth module.
Multiplicity	1
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Dependency
EthCtrlConfig	1..*	Configuration of the individual controller

10.2.3 EthCtrlConfig

[ECUC_Eth_00006] Definition of EcucParamConfContainerDef EthCtrlConfig [

Container Name	EthCtrlConfig
Parent Container	EthConfigSet
Description	Configuration of the individual controller
Multiplicity	1..*
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlEgressHardwareSupportedDataTransferThreshold	1	[ECUC_Eth_00135]
EthCtrlEnableEgressHardwareSupportedDataTransfer	1	[ECUC_Eth_00130]
EthCtrlEnableMii	1	[ECUC_Eth_00012]
EthCtrlEnableMmd	1	[ECUC_Eth_00137]
EthCtrlEnableRxInterrupt	1	[ECUC_Eth_00010]
EthCtrlEnableSpiInterface	0..1	[ECUC_Eth_00073]
EthCtrlEnableTxInterrupt	1	[ECUC_Eth_00011]
EthCtrlFramePreemptionEnable	1	[ECUC_Eth_00142]
EthCtrlIdx	1	[ECUC_Eth_00007]
EthCtrlInterPacketGap	0..1	[ECUC_Eth_00136]
EthCtrlMacLayerSpeed	0..1	[ECUC_Eth_00063]
EthCtrlMacLayerSubType	0..1	[ECUC_Eth_00062]
EthCtrlMacLayerType	1	[ECUC_Eth_00039]
EthCtrlMacSecEnabled	0..1	[ECUC_Eth_00150]
EthCtrlPhyAddress	0..1	[ECUC_Eth_00020]
EthCtrlEcucPartitionRef	0..1	[ECUC_Eth_00065]

Included Containers		
Container Name	Multiplicity	Dependency
EthClkUnit	0..*	This container contains the configuration of HW clock unit in the Ethernet Controller, which encapsulates a HW clock for ingress/ egress timestamping and optionally an adjustable HW clock to follow the PTP time. Tags: atp.Status=draft
EthCtrlClk	0..*	This container contains the configuration of a HW clock in the Ethernet Controller. Please note: It is recommended to always use the same hardware clock tree of the used platform for Ethernet hardware clocks which refer to the same EthClkUnit, otherwise cross-timestamping is needed. Tags: atp.Status=draft
EthCtrlConfigEgress	1	Configuration of one Ethernet controller egress behavior.
EthCtrlConfigIngress	1	Configuration of one Ethernet controller ingress behavior.
EthCtrlConfigSpiConfiguration	0..*	SPI Interface configuration of one Ethernet controller (MACPHY use). Configured only if EthCtrlEnableSpiInterface is set to TRUE. Tags: atp.Status=draft
EthCtrlMacSecConfig	0..1	This container contains the configuration of the MACsec functionality according to IEEE802.1AE (MACsec protocol) and IEEE802.1X (MACsec Key Agreement protocol) which is supported by this Ethernet controller hardware and used if MACsec protection for this Ethernet controller is enabled (EthCtrlMacSecEnabled set to TRUE).
EthCtrlPulsePerSecondConfig	0..1	This container contains the configuration of a HW Pulse per Second (PPS) feature. If not defined the PPS feature is not used. Tags: atp.Status=draft
EthDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.

[ECUC_Eth_00135] Definition of EcucIntegerParamDef EthCtrEgressHardwareSupportedDataTransferThreshold

Status: DRAFT

Parameter Name	EthCtrEgressHardwareSupportedDataTransferThreshold		
Parent Container	EthCtrlConfig		
Description	EthCtrEgressHardwareSupportedDataTransferThreshold define a threshold in bytes, if data, which is requested to be transmitted, shall be transferred with an hardware supported instruction (e.g. DMA) or via CPU copying process. If given data length for transmission exceeds the configured threshold, then the Eth driver shall initiate a hardware supported data transfer from the given source address(es) to the used egress queue entry (e.g. via DMA instruction). Otherwise the Eth driver shall perform a CPU driven copy of data to the used egress queue entry to the corresponding egress queue (e.g. via DMA instruction). Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	0		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00130] Definition of EcucBooleanParamDef EthCtrlEnableEgressHardwareSupportedDataTransfer

Status: DRAFT

Parameter Name	EthCtrlEnableEgressHardwareSupportedDataTransfer		
Parent Container	EthCtrlConfig		
Description	Eth driver shall use hardware supported data transfer form the upper layers to the corresponding egress queue (e.g. via DMA instruction) true: hardware supported data transfer is enabled false: hardware supported data transfer is disabled Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00012] Definition of EcucBooleanParamDef EthCtrlEnableMii [

Parameter Name	EthCtrlEnableMii		
Parent Container	EthCtrlConfig		
Description	Enables / Disables Media Independent Interface (MII) for transceiver access. Note: In case a MACPHY (external Ethernet controller) is use this parameter has to be enabled to ensure the existence of Eth_WriteMii and Eth_ReadMii. Within the function call of Eth_WriteMii and Eth_ReadMii, the register access is transformed to an SPI command.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	This parameter shall be set to TRUE, if EthCtrlEnableSpiInterface is set to TRUE		

]

[ECUC_Eth_00137] Definition of EcucBooleanParamDef EthCtrlEnableMmd [

Parameter Name	EthCtrlEnableMmd		
Parent Container	EthCtrlConfig		
Description	Enables/Disables Clause 45 Media Independent Interface (MII) for transceiver access		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

]

[ECUC_Eth_00010] Definition of EcucBooleanParamDef EthCtrlEnableRxInterrupt [

Parameter Name	EthCtrlEnableRxInterrupt		
Parent Container	EthCtrlConfig		
Description	Enables / Disables receive interrupt. Note: If this parameter is set to TRUE, then all ingress queues are handled in interrupt mode. If specific ingress queue need to be handled in interrupt mode, then this global parameter need to be set to FALSE and the specific ingress queue parameter EthCtrlEnableIngressQueueInterrupt need to be set to TRUE.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	





	Post-build time	–	
Dependency			

[ECUC_Eth_00073] Definition of EcucBooleanParamDef EthCtrlEnableSpiInterface

Parameter Name	EthCtrlEnableSpiInterface		
Parent Container	EthCtrlConfig		
Description	This optional parameter enables the processing of control data and Ethernet frames over the SPI interface specific for MACPHY device. The use of this parameter implies the respect of the SPI protocol described in OA TC6.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00011] Definition of EcucBooleanParamDef EthCtrlEnableTxInterrupt

Parameter Name	EthCtrlEnableTxInterrupt		
Parent Container	EthCtrlConfig		
Description	Enables / Disables transmit interrupt		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00142] Definition of EcucBooleanParamDef EthCtrlFramePreemptionEnable

Status: DRAFT

Parameter Name	EthCtrlFramePreemptionEnable		
Parent Container	EthCtrlConfig		
Description	Configures whether frame preemption for this Ethernet controller is enabled. If this parameter is set to TRUE and the Ethernet controller hardware supports frame preemption, then frame preemption for this Ethernet controller is enabled. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	If an Ethernet controller hardware does not support frame preemption, then this parameter shall be set to false.		

[ECUC_Eth_00007] Definition of EcucIntegerParamDef EthCtrlIdx

Parameter Name	EthCtrlIdx		
Parent Container	EthCtrlConfig		
Description	Specifies the instance ID of the configured controller.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	withAuto = true		

[ECUC_Eth_00136] Definition of EcucIntegerParamDef EthCtrlInterPacketGap

Parameter Name	EthCtrlInterPacketGap		
Parent Container	EthCtrlConfig		
Description	This parameter defines the transmit Inter-Packet Gap (IPG) (also called interframe gap (IFG)) between transmitted Ethernet packets in "byte times". This parameter can only exist if the hardware and driver support setting the IPG.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 10000		





Default value	12		
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	–	
Dependency			

]

[ECUC_Eth_00063] Definition of EcucEnumerationParamDef EthCtrlMacLayerSpeed

Parameter Name	EthCtrlMacLayerSpeed		
Parent Container	EthCtrlConfig		
Description	Defines the baud rate of the MAC layer.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	ETH_MAC_LAYER_SPEED_100M	–	
	ETH_MAC_LAYER_SPEED_10G	–	
	ETH_MAC_LAYER_SPEED_10M	–	
	ETH_MAC_LAYER_SPEED_1G	–	
	ETH_MAC_LAYER_SPEED_2500M	–	
	ETH_MAC_LAYER_SPEED_5G	–	
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Eth_00062] Definition of EcucEnumerationParamDef EthCtrlMacLayerSubType

Parameter Name	EthCtrlMacLayerSubType		
Parent Container	EthCtrlConfig		
Description	Defines the MAC layer subtype of a switch port		
Multiplicity	0..1		





Type	EcucEnumerationParamDef		
Range	REDUCED	–	
	REVERSED	–	
	SERIAL	–	
	STANDARD	–	
	UNIVERSAL_SERIAL	–	
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00039] Definition of EcucEnumerationParamDef EthCtrlMacLayer Type

Parameter Name	EthCtrlMacLayerType		
Parent Container	EthCtrlConfig		
Description	Defines the physical MAC/PHY Ethernet Interface type of the ethernet controller.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	ETH_MAC_LAYER_TYPE_XGMII	MAC layer interface (data) bandwidth class 1Gbit/s (e.g. GMII, RGMII, SGMII, RvGMII, USGMII)	
	ETH_MAC_LAYER_TYPE_XMII	MAC layer interface (data) bandwidth class 10-100 Mbit/s (e.g. MII, RMII, RvMII, SMII)	
	ETH_MAC_LAYER_TYPE_XXGMII	MAC layer interface (data) bandwidth class 10 Gbit/s	
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00150] Definition of EcucBooleanParamDef EthCtrlMacSecEnabled

Parameter Name	EthCtrlMacSecEnabled		
Parent Container	EthCtrlConfig		
Description	Specifies if MACsec according to IEEE802.1AE (MACsec protocol) and IEEE802.1X (MACsec Key Agreement protocol) is supported by the used Ethernet controller hardware and if the functionality (MACsec protection) is enabled. If the parameter is configured, MACsec is supported according to IEEE802.1AE and IEEE802.1X by the used Ethernet controller hardware. If the parameter is set to TRUE, the MACsec functionality according to IEEE802.1AE and IEEE802.1X is enabled. If the parameter is set to FALSE, the MACsec functionality according to IEEE802.1AE and IEEE802.1X is disabled. Note: Disabling of the MACsec functionality at this Ethernet controller is used if connected Ethernet hardware parts (e.g. Ethernet switch port or PHY) perform the MACsec protection.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
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	–	
Dependency			

[ECUC_Eth_00020] Definition of EcucStringParamDef EthCtrlPhyAddress

Parameter Name	EthCtrlPhyAddress		
Parent Container	EthCtrlConfig		
Description	Specifies the unique 48-bit physical address (MAC address) of the controller in network byte order.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	–		
Length	17-17		
Regular Expression	([0-9a-fA-F]{2:}{5}[0-9a-fA-F]{2})		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00065] Definition of EcucReferenceDef EthCtrlEcucPartitionRef [

Parameter Name	EthCtrlEcucPartitionRef		
Parent Container	EthCtrlConfig		
Description	Maps the Ethernet controller to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the Ethernet driver is mapped to.		
Multiplicity	0..1		
Type	Reference to EcucPartition		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
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	–	
Dependency			

]

[SWS_Eth_00260] [The ECUC partitions referenced by EthCtrlEcucPartitionRef shall be a subset of the ECUC partitions referenced by EthEcucPartitionRef.]

[SWS_Eth_00261] [EthCtrlConfig, EthTrcvConfig and EthSwtConfig (if existent in configuration) of one communication channel shall all reference the same ECUC partition.]

[SWS_Eth_CONSTR_00001] [If EthEcucPartitionRef references one or more ECUC partitions, EthCtrlEcucPartitionRef shall have a multiplicity of one and reference one of these ECUC partitions as well.]

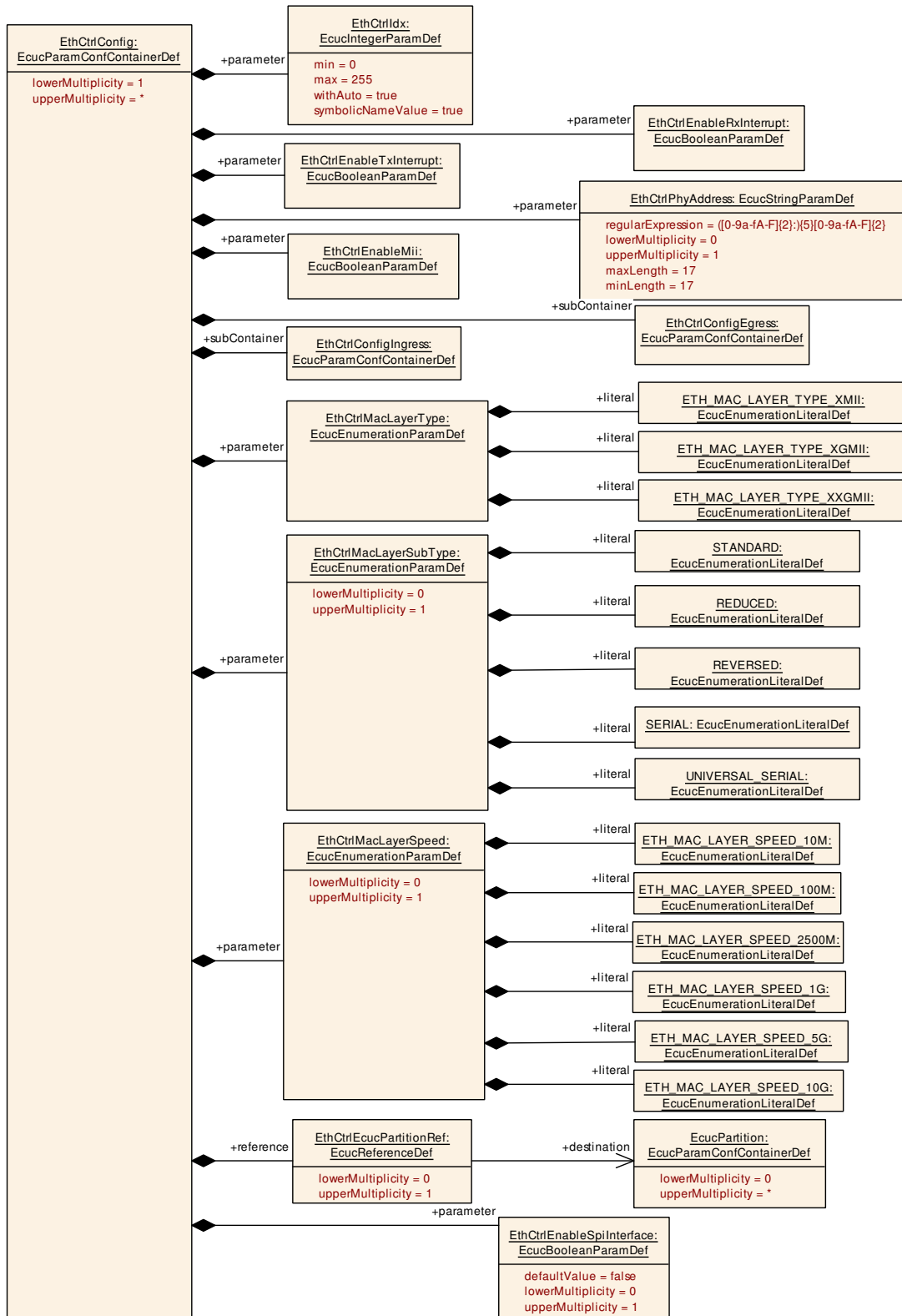


Figure 10.1: Overview EthCtrlConfig configuration

10.2.4 EthCtrlCik and EthCikUnit

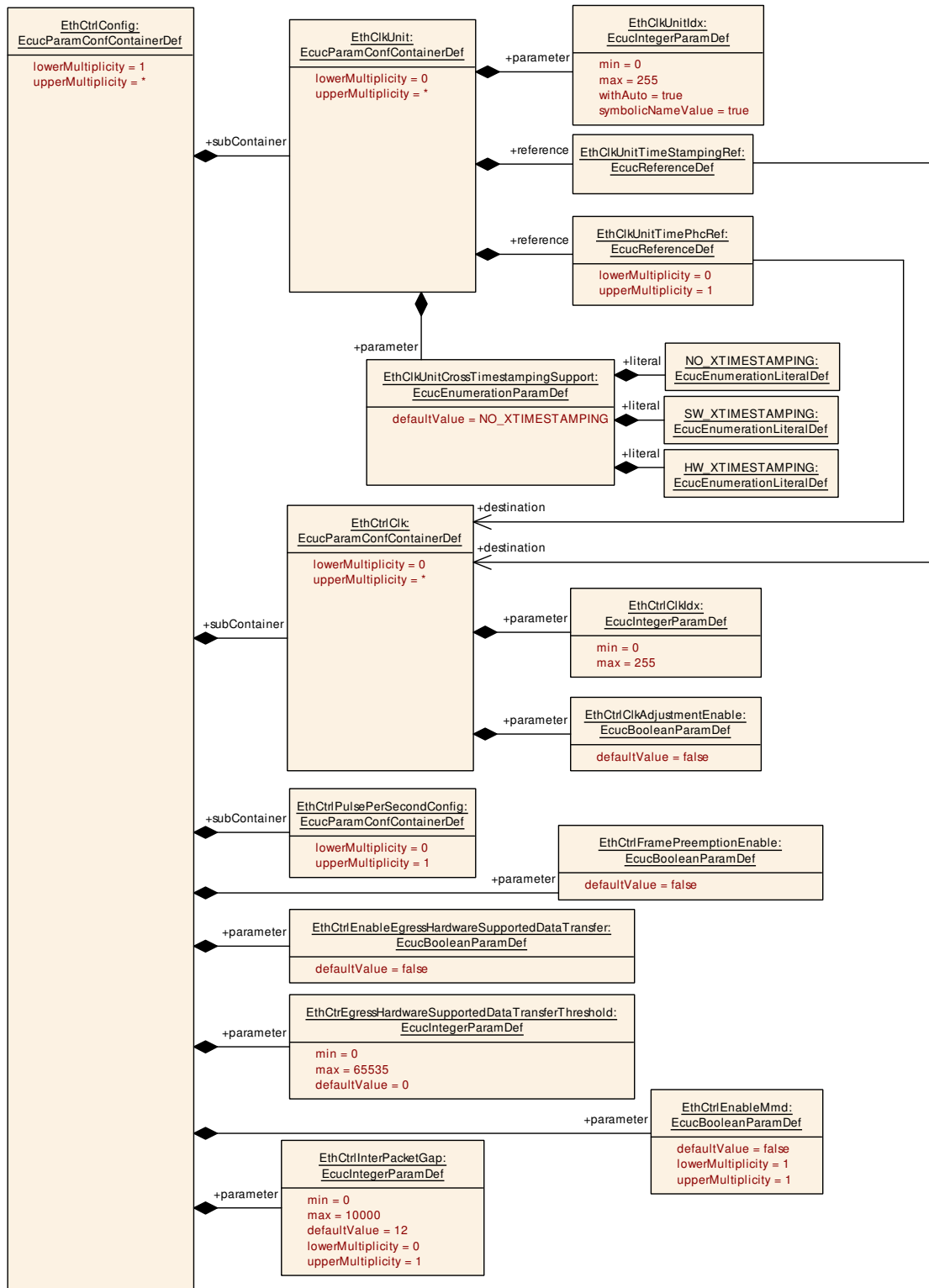


Figure 10.2: Overview EthCtrlCik and EthCikUnit

[ECUC_Eth_00115] Definition of EcucParamConfContainerDef EthCtrlClk

Status: DRAFT

[

Container Name	EthCtrlClk		
Parent Container	EthCtrlConfig		
Description	This container contains the configuration of a HW clock in the Ethernet Controller. Please note: It is recommended to always use the same hardware clock tree of the used platform for Ethernet hardware clocks which refer to the same EthClkUnit, otherwise cross-timestamping is needed. Tags: atp.Status=draft		
Multiplicity	0..*		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlClkAdjustmentEnable	1	[ECUC_Eth_00114]
EthCtrlClkIdx	1	[ECUC_Eth_00113]

No Included Containers

]

[ECUC_Eth_00114] Definition of EcucBooleanParamDef EthCtrlClkAdjustment Enable

Status: DRAFT

[

Parameter Name	EthCtrlClkAdjustmentEnable		
Parent Container	EthCtrlClk		
Description	Defines whether clock adjustment is enabled for this EthCtrlClk. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

]

[ECUC_Eth_00113] Definition of EcucIntegerParamDef EthCtrlClkIdx

Status: DRAFT

Parameter Name	EthCtrlClkIdx		
Parent Container	EthCtrlClk		
Description	Zero-based consecutive index of the HW clocks in the Ethernet Controller. Upper layer BSW modules and the Eth itself use this index to identify a clock in the Ethernet Controller. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00120] Definition of EcucParamConfContainerDef EthClkUnit

Status: DRAFT

Container Name	EthClkUnit		
Parent Container	EthCtrlConfig		
Description	This container contains the configuration of HW clock unit in the Ethernet Controller, which encapsulates a HW clock for ingress/egress timestamping and optionally an adjustable HW clock to follow the PTP time. Tags: atp.Status=draft		
Multiplicity	0..*		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	—	
	Post-build time	—	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthClkUnitCrossTimestampingSupport	1	[ECUC_Eth_00119]
EthClkUnitIdx	1	[ECUC_Eth_00118]
EthClkUnitTimePhcRef	0..1	[ECUC_Eth_00117]
EthClkUnitTimeStampingRef	1	[ECUC_Eth_00116]

No Included Containers

[ECUC_Eth_00119] Definition of EcucEnumerationParamDef EthClkUnitCrossTimestampingSupport

Status: DRAFT

Parameter Name	EthClkUnitCrossTimestampingSupport		
Parent Container	EthClkUnit		
Description	Defines the type of cross-timestamping between 2 HW clocks in the Ethernet Controller. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	HW_XTIMESTAMPING	Cross-timestamping is supported by HW of the Ethernet Controller. Tags: atp.Status=draft	
	NO_XTIMESTAMPING	No cross-timestamping is done (e.g. if only 1 HW clock is supported). Tags: atp.Status=draft	
	SW_XTIMESTAMPING	Cross-timestamping is done by SW of the Ethernet Driver. Tags: atp.Status=draft	
Default value	NO_XTIMESTAMPING		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00118] Definition of EcucIntegerParamDef EthClkUnitIdx

Status: DRAFT

Parameter Name	EthClkUnitIdx		
Parent Container	EthClkUnit		
Description	Zero-based consecutive index of the HW clock units in the Ethernet Controller. Upper layer BSW modules and the Eth itself use this index to identify a clock in the Ethernet Controller. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	withAuto = true		

[ECUC_Eth_00117] Definition of EcucReferenceDef EthClkUnitTimePhcRef

Status: DRAFT

[

Parameter Name	EthClkUnitTimePhcRef		
Parent Container	EthClkUnit		
Description	Reference to a HW clock in the Ethernet controller, which can be configured as PTP hardware clock (PHC). Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to EthCtrlClk		
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	–	
Dependency			

]

[ECUC_Eth_00116] Definition of EcucReferenceDef EthClkUnitTimeStampingRef

Status: DRAFT

[

Parameter Name	EthClkUnitTimeStampingRef		
Parent Container	EthClkUnit		
Description	Reference to a HW clock in the Ethernet controller, which is used by the Ethernet Controller for ingress/egress timestamping of frames. Tags: atp.Status=draft		
Multiplicity	1		
Type	Reference to EthCtrlClk		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

]

10.2.5 EthCtrlPulsePerSecondConfig

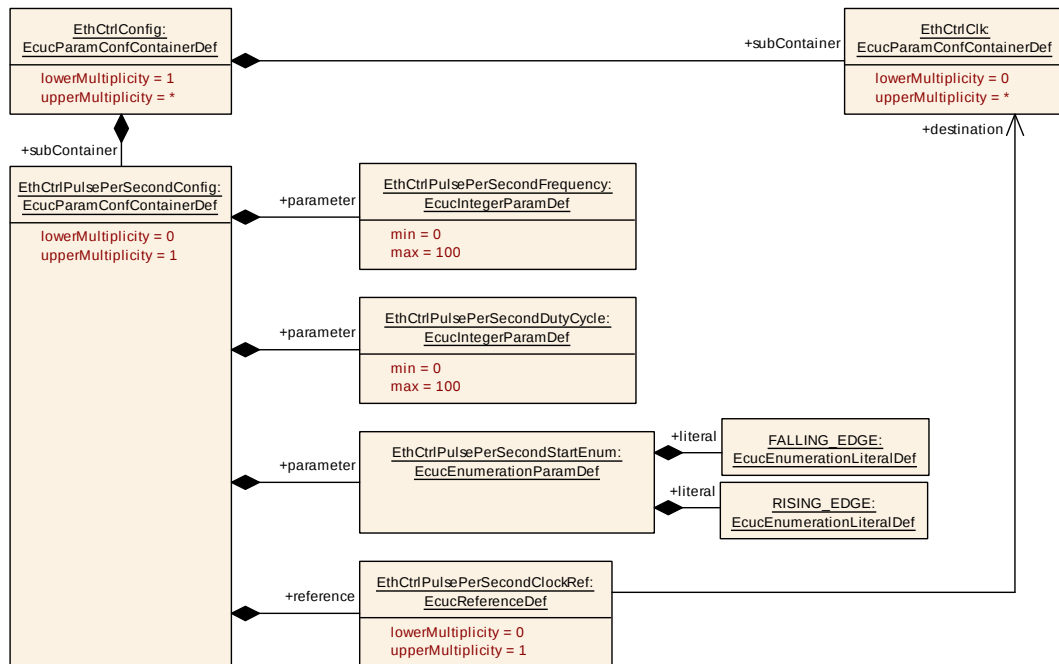


Figure 10.3: EthCtrlPulsePerSecondConfig

[ECUC_Eth_00111] Definition of EcucParamConfContainerDef EthCtrlPulsePerSecondConfig

Status: DRAFT

[

Container Name	EthCtrlPulsePerSecondConfig		
Parent Container	EthCtrlConfig		
Description	This container contains the configuration of a HW Pulse per Second (PPS) feature. If not defined the PPS feature is not used. Tags: atp.Status=draft		
Multiplicity	0..1		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	—	
	Post-build time	—	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlPulsePerSecondDutyCycle	1	[ECUC_Eth_00110]
EthCtrlPulsePerSecondFrequency	1	[ECUC_Eth_00109]
EthCtrlPulsePerSecondStartEnum	1	[ECUC_Eth_00108]
EthCtrlPulsePerSecondClockRef	0..1	[ECUC_Eth_00112]

No Included Containers

]

[ECUC_Eth_00110] Definition of EcucIntegerParamDef EthCtrlPulsePerSecondDutyCycle

Status: DRAFT

[

Parameter Name	EthCtrlPulsePerSecondDutyCycle		
Parent Container	EthCtrlPulsePerSecondConfig		
Description	Configuration how long each Pulse shall be defined in percent. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 100		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

]

[ECUC_Eth_00109] Definition of EcucIntegerParamDef EthCtrlPulsePerSecondFrequency

Status: DRAFT

[

Parameter Name	EthCtrlPulsePerSecondFrequency		
Parent Container	EthCtrlPulsePerSecondConfig		
Description	Configuration how many Pulse per Second pulses shall be created per second. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 100		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

]

[ECUC_Eth_00108] Definition of EcucEnumerationParamDef EthCtrlPulsePerSecondStartEnum

Status: DRAFT

Parameter Name	EthCtrlPulsePerSecondStartEnum		
Parent Container	EthCtrlPulsePerSecondConfig		
Description	Defines whether the pulse starts with a rising or a falling edge. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FALLING_EDGE	PPS starts with a falling edge. Tags: atp.Status=draft	
	RISING_EDGE	PPS starts with a rising edge. Tags: atp.Status=draft	
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	—	
	Post-build time	—	
Dependency			

[ECUC_Eth_00112] Definition of EcucReferenceDef EthCtrlPulsePerSecondClockRef

Status: DRAFT

Parameter Name	EthCtrlPulsePerSecondClockRef		
Parent Container	EthCtrlPulsePerSecondConfig		
Description	Reference to a HW clock in the Ethernet controller, which is taken as the source for the PPS (Pulse Per Second). Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to EthCtrlClk		
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	–	
Dependency			

10.2.6 EthCtrlConfigEgress

[ECUC_Eth_00046] Definition of EcucParamConfContainerDef EthCtrlConfigEgress

Container Name	EthCtrlConfigEgress
Parent Container	EthCtrlConfig
Description	Configuration of one Ethernet controller egress behavior.
Multiplicity	1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlConfigEgressLastSchedulerRef	1	[ECUC_Eth_00052]

Included Containers		
Container Name	Multiplicity	Dependency
EthCtrlConfigEgressQueue	1..*	Represents a queue at the egress side. Tags: atp.Status=draft
EthCtrlConfigScheduler	1..*	Represents a Scheduler on the egress side.

[ECUC_Eth_00052] Definition of EcucReferenceDef EthCtrlConfigEgressLastSchedulerRef

Parameter Name	EthCtrlConfigEgressLastSchedulerRef		
Parent Container	EthCtrlConfigEgress		
Description	Reference to the scheduler which is the last in the egress structure.		
Multiplicity	1		
Type	Reference to EthCtrlConfigScheduler		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

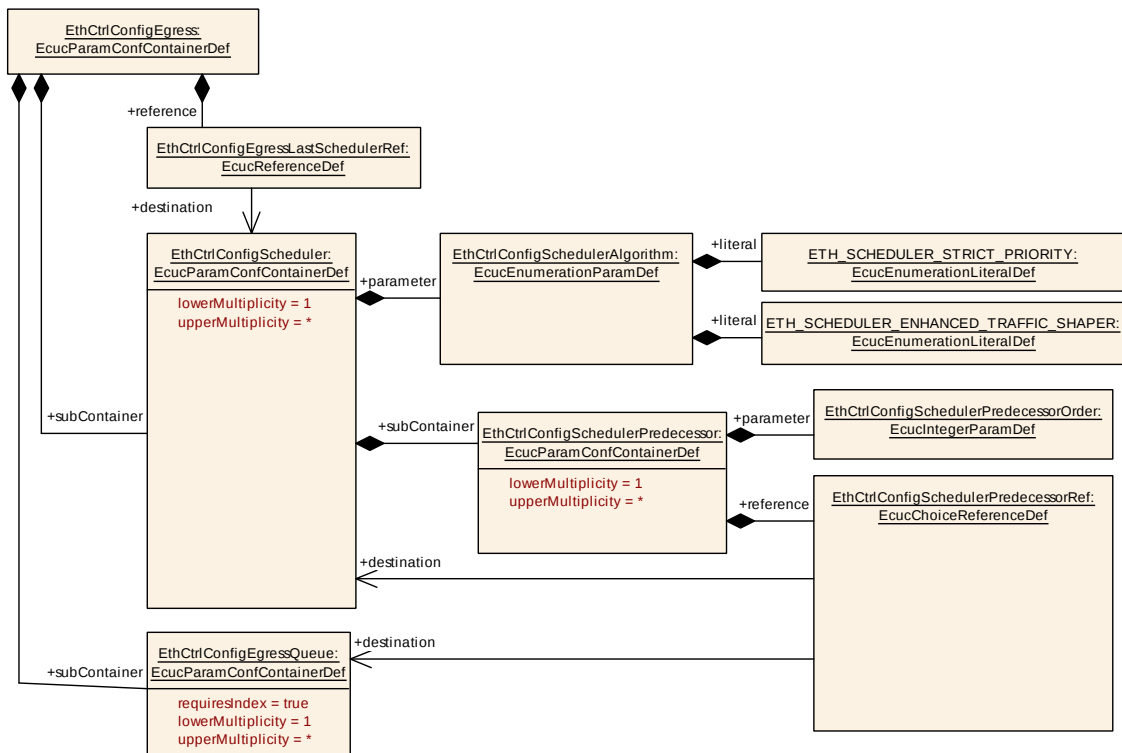


Figure 10.4: Overview EthCtrlConfigEgress configuration - DRAFT

10.2.6.1 EthCtrlConfigEgressQueue - DRAFT

[ECUC_Eth_00090] Definition of EcucParamConfContainerDef EthCtrlConfigEgressQueue

Status: DRAFT

[

Container Name	EthCtrlConfigEgressQueue
Parent Container	EthCtrlConfigEgress
Description	Represents a queue at the egress side. Tags: atp.Status=draft Attributes: requiresIndex=true
Multiplicity	1..*
Post-Build Variant Multiplicity	false
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlConfigEgressQueueBufLenByte	1	[ECUC_Eth_00092]
EthCtrlConfigEgressQueueBufTotal	1	[ECUC_Eth_00093]
EthCtrlConfigEgressQueueIdx	1	[ECUC_Eth_00091]

Included Containers		
Container Name	Multiplicity	Dependency
EthCtrlConfigEgressQueueSortingType	0..1	Defines the sorting filter for the queue. Optional if only one queue is available. Tags: atp.Status=draft
EthCtrlConfigEgressQueueTransmissionSelection	1	Represents the transmission selection of a queue at the egress side. Tags: atp.Status=draft

[ECUC_Eth_00092] Definition of EcucIntegerParamDef EthCtrlConfigEgressQueueBufLenByte

Status: DRAFT

Parameter Name	EthCtrlConfigEgressQueueBufLenByte		
Parent Container	EthCtrlConfigEgressQueue		
Description	Defines the length of one queue element in bytes. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00093] Definition of EcucIntegerParamDef EthCtrlConfigEgressQueueBufTotal

Status: DRAFT

Parameter Name	EthCtrlConfigEgressQueueBufTotal		
Parent Container	EthCtrlConfigEgressQueue		
Description	Defines the count of queue elements for one queue. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD





Dependency	
------------	--

]

[ECUC_Eth_00091] Definition of EcucIntegerParamDef EthCtrlConfigEgressQueueIdx

Status: DRAFT

[

Parameter Name	EthCtrlConfigEgressQueueIdx		
Parent Container	EthCtrlConfigEgressQueue		
Description	Defines the queue index. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	withAuto = true		

]

10.2.6.2 EthCtrlConfigEgressQueueSortingType - DRAFT

[ECUC_Eth_00143] Definition of EcucParamConfContainerDef EthCtrlConfigEgressQueueSortingType

Status: DRAFT

[

Container Name	EthCtrlConfigEgressQueueSortingType		
Parent Container	EthCtrlConfigEgressQueue		
Description	Defines the sorting filter for the queue. Optional if only one queue is available. Tags: atp.Status=draft		
Multiplicity	0..1		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			
No Included Parameters			

Included Containers		
Container Name	Multiplicity	Dependency
EthCtrlConfigEgressQueueSortingEntry	1..*	Defines a field of the sorting filter. Tags: atp.Status=draft

]

10.2.6.3 EthCtrlConfigEgressQueueSortingEntry - DRAFT

[ECUC_Eth_00144] Definition of EcucParamConfContainerDef EthCtrlConfigEgressQueueSortingEntry

Status: DRAFT

[

Container Name	EthCtrlConfigEgressQueueSortingEntry		
Parent Container	EthCtrlConfigEgressQueueSortingType		
Description	Defines a field of the sorting filter. Tags: atp.Status=draft		
Multiplicity	1..*		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlConfigEgressQueueSortingKey	1	[ECUC_Eth_00147]
EthCtrlConfigEgressQueueSortingMask	1	[ECUC_Eth_00148]
EthCtrlConfigEgressQueueSortingOffset	1	[ECUC_Eth_00146]
EthCtrlConfigEgressQueueSortingSize	1	[ECUC_Eth_00145]

No Included Containers

]

[ECUC_Eth_00147] Definition of EcucIntegerParamDef EthCtrlConfigEgressQueueSortingKey

Status: DRAFT

[

Parameter Name	EthCtrlConfigEgressQueueSortingKey
Parent Container	EthCtrlConfigEgressQueueSortingEntry
Description	Defines the value the masked filtered field has to match. Tags: atp.Status=draft





Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Eth_00148] Definition of EcucIntegerParamDef EthCtrlConfigEgressQueueSortingMask

Status: DRAFT

[

Parameter Name	EthCtrlConfigEgressQueueSortingMask		
Parent Container	EthCtrlConfigEgressQueueSortingEntry		
Description	Defines the mask for matching the filtered field. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Eth_00146] Definition of EcucIntegerParamDef EthCtrlConfigEgressQueueSortingOffset

Status: DRAFT

[

Parameter Name	EthCtrlConfigEgressQueueSortingOffset		
Parent Container	EthCtrlConfigEgressQueueSortingEntry		
Description	Defines the offset of the filtered field in bytes, where 0 denotes the position of the destination MAC address. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 47		
Default value	–		





Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Eth_00145] Definition of EcucEnumerationParamDef EthCtrlConfigEgressQueueSortingSize

Status: DRAFT

[

Parameter Name	EthCtrlConfigEgressQueueSortingSize		
Parent Container	EthCtrlConfigEgressQueueSortingEntry		
Description	Defines the size of the filtered field. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	UINT16	–	Tags: atp.Status=draft
	UINT32	–	Tags: atp.Status=draft
	UINT64	–	Tags: atp.Status=draft
	UINT8	–	Tags: atp.Status=draft
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

10.2.6.4 EthCtrlConfigEgressQueueTransmissionSelection - DRAFT

[ECUC_Eth_00100] Definition of EcucParamConfContainerDef EthCtrlConfigEgressQueueTransmissionSelection

Status: DRAFT

[

Container Name	EthCtrlConfigEgressQueueTransmissionSelection
Parent Container	EthCtrlConfigEgressQueue
Description	Represents the transmission selection of a queue at the egress side. Tags: atp.Status=draft
Multiplicity	1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm	1	[ECUC_Eth_00106]

Included Containers		
Container Name	Multiplicity	Dependency
EthCtrlConfigEgressQueueTransmissionSelectionCBSConfig	0..1	Represents the configuration of a credit based shaper transmission selection algorithm for an queue at the egress side. This configuration is used if the EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm is set to ETH_TRANSMISSION_SELECTION_CBS. Lower and upper boundaries of the CBS credit are not configurable by the Eth driver. If the credit counter of an implementation should over- or under-run its upper or lower limit at runtime, the resulting behavior is not specified by AUTOSAR or by IEEE and remains implementation specific. Tags: atp.Status=draft
EthCtrlConfigEgressQueueTransmissionSelectionETS	0..1	Represents the configuration of an enhanced transmission selection algorithm of an egress queue. This configuration is used if the EthCtrlConfigEgressQueueTransmissionSelection is set to ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER. The subordinated configuration parameters allow to configure the ETS in different variants to support commonly used scheduler algorithms (e.g. weighted round robin). Tags: atp.Status=draft

[ECUC_Eth_00106] Definition of EcucEnumerationParamDef EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm

Status: DRAFT

Parameter Name	EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm	
Parent Container	EthCtrlConfigEgressQueueTransmissionSelection	
Description	Represents the transmission selection of a queue at the egress side. Tags: atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ETH_TRANSMISSION_SELECTION_ATS	Ethernet frames are selected from the egress queue for transmission according the asynchronous traffic shaping algorithm. Tags: atp.Status=draft





	ETH_TRANSMISSION_SELECTION_CBS	Ethernet frames are selected from the egress queue for transmission according the credit based shaping algorithm. Tags: atp.Status=draft	
	ETH_TRANSMISSION_SELECTION_ETS	Ethernet frames are selected from the egress queue for transmission according the enhanced transmission selection algorithm. Tags: atp.Status=draft	
	ETH_TRANSMISSION_SELECTION_UNSHAPED	Ethernet frames are selected from the egress queue for transmission in an unshaped manner. Please note: IEEE802.1Q uses the term "strict priority". Term "UNSHAPED" is used to avoid confusion with strict priority in context of EthCtrl ConfigScheduler. Tags: atp.Status=draft	
Default value	ETH_TRANSMISSION_SELECTION_UNSHAPED		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00138] Definition of EcucParamConfContainerDef EthCtrlConfigEgressQueueTransmissionSelectionETS

Status: DRAFT

Container Name	EthCtrlConfigEgressQueueTransmissionSelectionETS		
Parent Container	EthCtrlConfigEgressQueueTransmissionSelection		
Description	Represents the configuration of an enhanced transmission selection algorithm of an egress queue. This configuration is used if the EthCtrlConfigEgressQueueTransmissionSelection is set to ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER. The subordinated configuration parameters allow to configure the ETS in different variants to support commonly used scheduler algorithms (e.g. weighted round robin). Tags: atp.Status=draft		
Multiplicity	0..1		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlETSTConfigAvailableBandwidthInPercent	0..1	[ECUC_Eth_00139]
EthCtrlETSTConfigAvailableBandwidthInWeightValue	0..1	[ECUC_Eth_00140]

No Included Containers

[ECUC_Eth_00139] Definition of EcucIntegerParamDef EthCtrlETSTConfigAvailableBandwidthInPercent

Status: DRAFT

Parameter Name	EthCtrlETSTConfigAvailableBandwidthInPercent		
Parent Container	EthCtrlConfigEgressQueueTransmissionSelectionETS		
Description	Represents the configuration of an enhanced transmission selection algorithm for one egress queue, where the available bandwidth is configured in percent. The percent value represents the available bandwidth for emission opportunities to transmit Ethernet frames calculated in bits. The resolution is 1%. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 100		
Default value	30		
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	–	
Dependency			

[ECUC_Eth_00140] Definition of EcucIntegerParamDef EthCtrlETSTConfigAvailableBandwidthInWeightValue

Status: DRAFT

Parameter Name	EthCtrlETSTConfigAvailableBandwidthInWeightValue		
Parent Container	EthCtrlConfigEgressQueueTransmissionSelectionETS		
Description	Represents the configuration of an enhanced transmission selection algorithm of an egress queue, where the available bandwidth is configured as weight value. The weight value represents the number of emission opportunities to transmit Ethernet frames. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	1		
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	–	
Dependency			

]

10.2.6.5 EthCtrlConfigEgressQueueTransmissionSelectionCBSConfig - DRAFT

[ECUC_Eth_00101] Definition of EcucParamConfContainerDef EthCtrlConfigEgressQueueTransmissionSelectionCBSConfig

Status: DRAFT

[

Container Name	EthCtrlConfigEgressQueueTransmissionSelectionCBSConfig		
Parent Container	EthCtrlConfigEgressQueueTransmissionSelection		
Description	Represents the configuration of a credit based shaper transmission selection algorithm for an queue at the egress side. This configuration is used if the EthCtrlConfigEgressQueueTransmissionSelection Algorithm is set to ETH_TRANSMISSION_SELECTION_CBS. Lower and upper boundaries of the CBS credit are not configurable by the Eth driver. If the credit counter of an implementation should over- or under-run its upper or lower limit at runtime, the resulting behavior is not specified by AUTOSAR or by IEEE and remains implementation specific. Tags: atp.Status=draft		
Multiplicity	0..1		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlConfigEgressQueueCreditBasedShaperIdleSlope	0..1	[ECUC_Eth_00149]

No Included Containers

]

[ECUC_Eth_00149] Definition of EcucIntegerParamDef EthCtrlConfigEgressQueueCreditBasedShaperIdleSlope

Status: DRAFT

[			
Parameter Name	EthCtrlConfigEgressQueueCreditBasedShaperIdleSlope		
Parent Container	EthCtrlConfigEgressQueueTransmissionSelectionCBSSConfig		
Description	Defines the idle slope of a queue at egress side. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	—		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	—	
	Post-build time	—	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

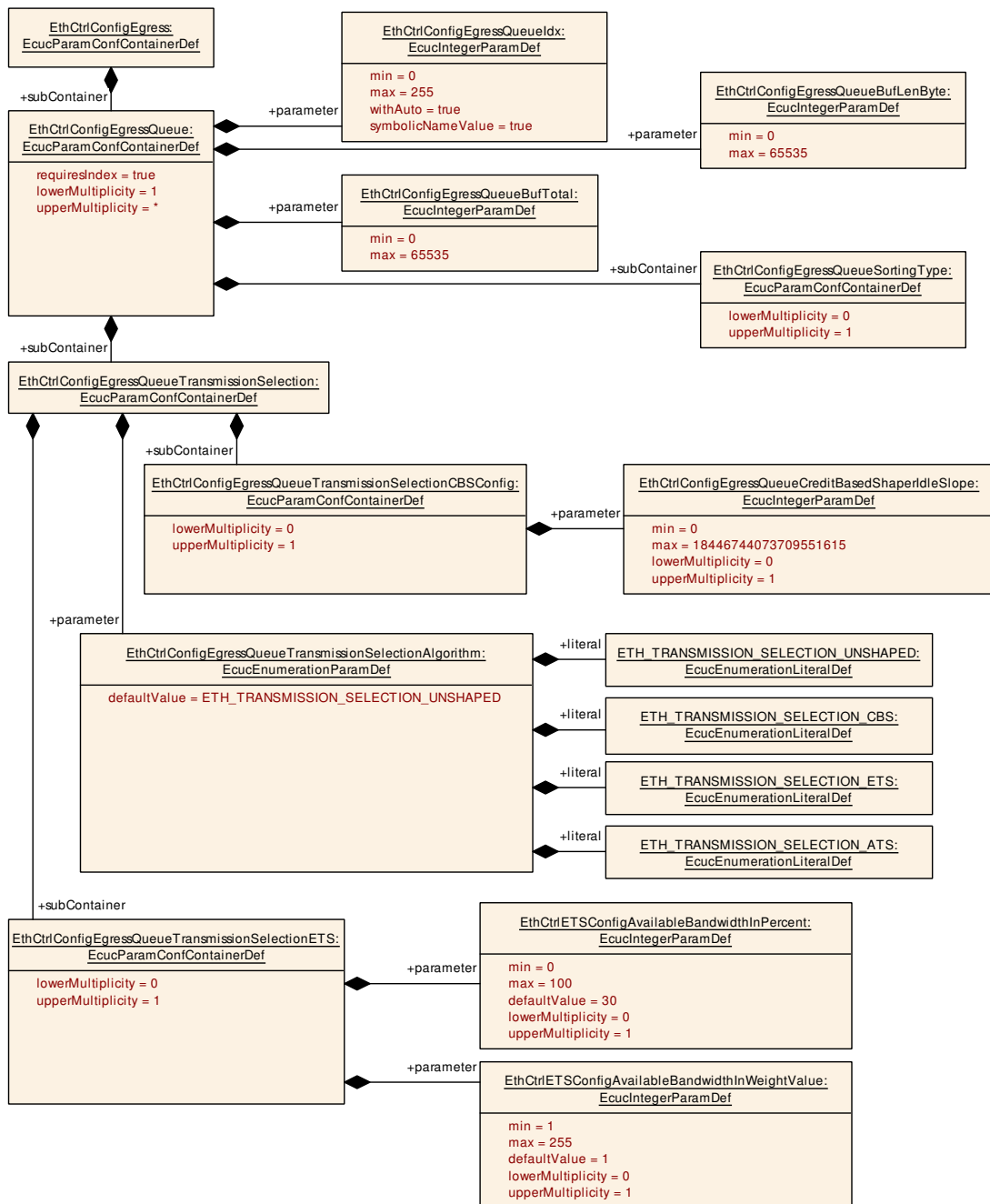


Figure 10.5: Overview EthCtrlConfigEgressQueue configuration - DRAFT

10.2.6.6 EthCtrlConfigScheduler

[ECUC_Eth_00053] Definition of EcucParamConfContainerDef EthCtrlConfig Scheduler

Container Name	EthCtrlConfigScheduler
Parent Container	EthCtrlConfigEgress
Description	Represents a Scheduler on the egress side.
Multiplicity	1..*
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlConfigSchedulerAlgorithm	1	[ECUC_Eth_00141]

Included Containers		
Container Name	Multiplicity	Dependency
EthCtrlConfigSchedulerPredecessor	1..*	Defines an ordered list of predecessors for this scheduler.

[ECUC_Eth_00141] Definition of EcucEnumerationParamDef EthCtrlConfig SchedulerAlgorithm [

Parameter Name	EthCtrlConfigSchedulerAlgorithm		
Parent Container	EthCtrlConfigScheduler		
Description	Defines the scheduler algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	ETH_SCHEDULER_ENHANCED_TRAFFIC_SHAPER	Represents a scheduler used for enhanced traffic shaping (e.g. weighted round robin).	
	ETH_SCHEDULER_STRICT_PRIORITY	Represents a strict priority scheduler.	
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

10.2.6.7 EthCtrlConfigSchedulerPredecessor

[ECUC_Eth_00054] Definition of EcucParamConfContainerDef EthCtrlConfig SchedulerPredecessor [

Container Name	EthCtrlConfigSchedulerPredecessor
Parent Container	EthCtrlConfigScheduler
Description	Defines an ordered list of predecessors for this scheduler.
Multiplicity	1..*
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlConfigSchedulerPredecessorOrder	1	[ECUC_Eth_00055]
EthCtrlConfigSchedulerPredecessorRef	1	[ECUC_Eth_00056]

No Included Containers

]

[[ECUC_Eth_00055](#)] Definition of EcucIntegerParamDef EthCtrlConfigSchedulerPredecessorOrder [

Parameter Name	EthCtrlConfigSchedulerPredecessorOrder		
Parent Container	EthCtrlConfigSchedulerPredecessor		
Description	Defines the order of the scheduler predecessors.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[[ECUC_Eth_00056](#)] Definition of EcucChoiceReferenceDef EthCtrlConfigSchedulerPredecessorRef [

Parameter Name	EthCtrlConfigSchedulerPredecessorRef		
Parent Container	EthCtrlConfigSchedulerPredecessor		
Description	Choice reference to the scheduler predecessor.		
Multiplicity	1		
Type	Choice reference to [EthCtrlConfigEgressQueue , EthCtrlConfigScheduler]		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

]

10.2.7 EthCtrlConfigIngress

[[ECUC_Eth_00040](#)] Definition of EcucParamConfContainerDef EthCtrlConfigIngress [

Container Name	EthCtrlConfigIngress
Parent Container	EthCtrlConfig
Description	Configuration of one Ethernet controller ingress behavior.
Multiplicity	1
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Dependency
EthCtrlConfigIngressQueue	0..*	Represents a queue at the ingress side. Tags: atp.Status=draft
EthCtrlConfigIngressQueueSortingPriority	0..1	Defines the sorting priority of Ethernet frame attributes (priority, MacDstAddr, VlanId). If an Ethernet frame is received and several ingress queues of the same EthCtrl have different EthCtrlConfigIngressQueueSortingTypes configured, then the Ethernet frames shall be sorted regarding the EthCtrlConfigIngressQueueSortingType set to the highest priority. If no matching queue was found, proceed in decending order with the next sorting EthCtrlConfigIngressQueueSortingType. If Ethernet frame could not be sorted in any ingress queue, then drop this Ethernet frame. Tags: atp.Status=draft

]

10.2.7.1 EthCtrlConfigIngressQueue - DRAFT

The following parameter are introduced in [EthCtrlConfigIngressQueue](#)
[EthCtrlConfigIngressQueueOverwriteEnabled](#) [EthCtrlEnableIngressQueueInterrupt](#)

[ECUC_Eth_00095] Definition of EcucParamConfContainerDef EthCtrlConfigIngressQueue

Status: DRAFT

[

Container Name	EthCtrlConfigIngressQueue
Parent Container	EthCtrlConfigIngress
Description	Represents a queue at the ingress side. Tags: atp.Status=draft
Multiplicity	0..*
Post-Build Variant Multiplicity	false
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlConfigIngressQueueBufLenByte	1	[ECUC_Eth_00099]
EthCtrlConfigIngressQueueBufTotal	1	[ECUC_Eth_00098]
EthCtrlConfigIngressQueueHandlerFunction	0..1	[ECUC_Eth_00134]
EthCtrlConfigIngressQueueIdx	1	[ECUC_Eth_00096]
EthCtrlConfigIngressQueueOverwriteEnabled	1	[ECUC_Eth_00133]
EthCtrlEnableIngressQueueInterrupt	1	[ECUC_Eth_00129]

Included Containers		
Container Name	Multiplicity	Dependency
EthCtrlConfigIngressQueueSortingType	0..1	Defines one out of 4 possible sorting criteria for this ingress queue. Tags: atp.Status=draft

[ECUC_Eth_00099] Definition of EcucIntegerParamDef EthCtrlConfigIngressQueueBufLenByte

Status: DRAFT

Parameter Name	EthCtrlConfigIngressQueueBufLenByte		
Parent Container	EthCtrlConfigIngressQueue		
Description	Defines the length of one queue element in bytes. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00098] Definition of EcucIntegerParamDef EthCtrlConfigIngressQueueBufTotal

Status: DRAFT

Parameter Name	EthCtrlConfigIngressQueueBufTotal		
Parent Container	EthCtrlConfigIngressQueue		
Description	Defines the count of queue elements for one queue. Tags: atp.Status=draft		
Multiplicity	1		





Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Eth_00134] Definition of EcucFunctionNameDef EthCtrlConfigIngressQueueHandlerFunction

Status: DRAFT

[

Parameter Name	EthCtrlConfigIngressQueueHandlerFunction		
Parent Container	EthCtrlConfigIngressQueue		
Description	Specifies ingress queue handler function. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	–		
Regular Expression	–		
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	–	
Dependency			

]

[ECUC_Eth_00096] Definition of EcucIntegerParamDef EthCtrlConfigIngressQueueIdx

Status: DRAFT

[

Parameter Name	EthCtrlConfigIngressQueueIdx		
Parent Container	EthCtrlConfigIngressQueue		
Description	Defines the queue index. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		





Range	0 .. 255	
Default value	–	
Post-Build Variant Value	false	
Value Configuration Class	Pre-compile time	X All Variants
	Link time	–
	Post-build time	–
Dependency	withAuto = true	

[ECUC_Eth_00133] Definition of EcucBooleanParamDef EthCtrlConfigIngressQueueOverwriteEnabled

Status: DRAFT

Parameter Name	EthCtrlConfigIngressQueueOverwriteEnabled		
Parent Container	EthCtrlConfigIngressQueue		
Description	Defines the handling if all ingress queue elements are occupied and the Ethernet controller needs to enqueue a further Ethernet frame. FALSE: Overwrite of the eldest available (i.e. not locked by a reception process) ingress queue element disabled. Enqueueing of further Ethernet frames is rejected. TRUE: Overwrite of the eldest available (i.e. not locked by an repetition process) ingress queue element enabled. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00129] Definition of EcucBooleanParamDef EthCtrlEnableIngressQueueInterrupt

Status: DRAFT

Parameter Name	EthCtrlEnableIngressQueueInterrupt		
Parent Container	EthCtrlConfigIngressQueue		
Description	Enables / Disables receive interrupt of this specific queue. Please note: This would enable an interrupt for this specific ingress queue upon reception of an Ethernet frame. Some ingress queue may be handled interrupt mode and some in polling mode. Therefore the global parameter EthCtrlEnableRxInterrupt, where all ingress queues are handled in interrupt mode, need to be set to FALSE. Tags: atp.Status=draft		
Multiplicity	1		





Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	If EthCtrlEnableIngressQueueInterrupt is set to TRUE, then EthCtrlEnableRxInterrupt has to be set to FALSE.		

[ECUC_Eth_00131] Definition of EcucParamConfContainerDef EthCtrlConfigIngressQueueSortingType

Status: DRAFT

Container Name	EthCtrlConfigIngressQueueSortingType		
Parent Container	EthCtrlConfigIngressQueue		
Description	Defines one out of 4 possible sorting criteria for this ingress queue. Tags: atp.Status=draft		
Multiplicity	0..1		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlIngressQueueSortingEtherTypeAssignment	0..1	[ECUC_Eth_00128]
EthCtrlIngressQueueSortingMacDestinationAssignment	0..1	[ECUC_Eth_00127]
EthCtrlIngressQueueSortingVlanIdAssignment	0..1	[ECUC_Eth_00126]
EthCtrlIngressQueueSortingVlanPriorityAssignment	0..8	[ECUC_Eth_00125]

No Included Containers

[ECUC_Eth_00128] Definition of EcucIntegerParamDef EthCtrlIngressQueueSortingEtherTypeAssignment

Status: DRAFT

Parameter Name	EthCtrlIngressQueueSortingEtherTypeAssignment		
Parent Container	EthCtrlConfigIngressQueueSortingType		
Description	Defines that the EtherType shall be used to assign frames to this ingress queue. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
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	–	
Dependency			

[ECUC_Eth_00127] Definition of EcucStringParamDef EthCtrlIngressQueueSortingMacDestinationAssignment

Status: DRAFT

Parameter Name	EthCtrlIngressQueueSortingMacDestinationAssignment		
Parent Container	EthCtrlConfigIngressQueueSortingType		
Description	Defines that the Destination MAC Address shall be used to assign frames to this ingress queue. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	–		
Length	17-17		
Regular Expression	([0-9a-fA-F]{2:}{5}[0-9a-fA-F]{2})		
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	–	





Dependency	
------------	--

]

[ECUC_Eth_00126] Definition of EcucIntegerParamDef EthCtrlIngressQueueSortingVlanIdAssignment

Status: DRAFT

[

Parameter Name	EthCtrlIngressQueueSortingVlanIdAssignment		
Parent Container	EthCtrlConfigIngressQueueSortingType		
Description	Defines that the VLAN ID shall be used to assign frames to this ingress queue. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
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	–	
Dependency			

]

[ECUC_Eth_00125] Definition of EcucIntegerParamDef EthCtrlIngressQueueSortingVlanPriorityAssignment

Status: DRAFT

[

Parameter Name	EthCtrlIngressQueueSortingVlanPriorityAssignment		
Parent Container	EthCtrlConfigIngressQueueSortingType		
Description	Defines that the VLAN priority shall be used to assign frames to this ingress queue. Tags: atp.Status=draft		
Multiplicity	0..8		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default value	–		
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	–	
Dependency			

[ECUC_Eth_00132] Definition of EcucParamConfContainerDef EthCtrlConfigIngressQueueSortingPriority

Status: DRAFT

Container Name	EthCtrlConfigIngressQueueSortingPriority		
Parent Container	EthCtrlConfigIngress		
Description	Defines the sorting priority of Ethernet frame attributes (priority, MacDstAddr, VlanId). If an Ethernet frame is received and several ingress queues of the same EthCtrl have different EthCtrlConfigIngressQueueSortingTypes configured, then the Ethernet frames shall be sorted regarding the EthCtrlConfigIngressQueueSortingType set to the highest priority. If no matching queue was found, proceed in decending order with the next sorting Eth CtrlConfigIngressQueueSortingType. If Ethernet frame could not be sorted in any ingress queue, then drop this Ethernet frame. Tags: atp.Status=draft		
Multiplicity	0..1		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SortingPriorityEtherTypeAssignment	1	[ECUC_Eth_00124]
SortingPriorityMacDestinationAssignment	1	[ECUC_Eth_00123]
SortingPriorityVlanIdAssignment	1	[ECUC_Eth_00122]
SortingPriorityVlanPriorityAssignment	1	[ECUC_Eth_00121]

No Included Containers

[ECUC_Eth_00124] Definition of EcucIntegerParamDef SortingPriorityEtherType Assignment

Status: DRAFT

[

Parameter Name	SortingPriorityEtherTypeAssignment		
Parent Container	EthCtrlConfigIngressQueueSortingPriority		
Description	Defines the sorting priority for EtherType assignment. 0 has the highest priority. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 3		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Eth_00123] Definition of EcucIntegerParamDef SortingPriorityMacDestinationAssignment

Status: DRAFT

[

Parameter Name	SortingPriorityMacDestinationAssignment		
Parent Container	EthCtrlConfigIngressQueueSortingPriority		
Description	Defines the sorting priority for MAC destination assignment. 0 has the highest priority. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 3		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Eth_00122] Definition of EcucIntegerParamDef SortingPriorityVlanIdAssignment

Status: DRAFT

Parameter Name	SortingPriorityVlanIdAssignment		
Parent Container	EthCtrlConfigIngressQueueSortingPriority		
Description	Defines the sorting priority for VLAN ID assignment. 0 has the highest priority. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 3		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00121] Definition of EcucIntegerParamDef SortingPriorityVlanPriorityAssignment

Status: DRAFT

Parameter Name	SortingPriorityVlanPriorityAssignment		
Parent Container	EthCtrlConfigIngressQueueSortingPriority		
Description	Defines the sorting priority for VLAN priority assignment. 0 has the highest priority. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 3		
Default value	—		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

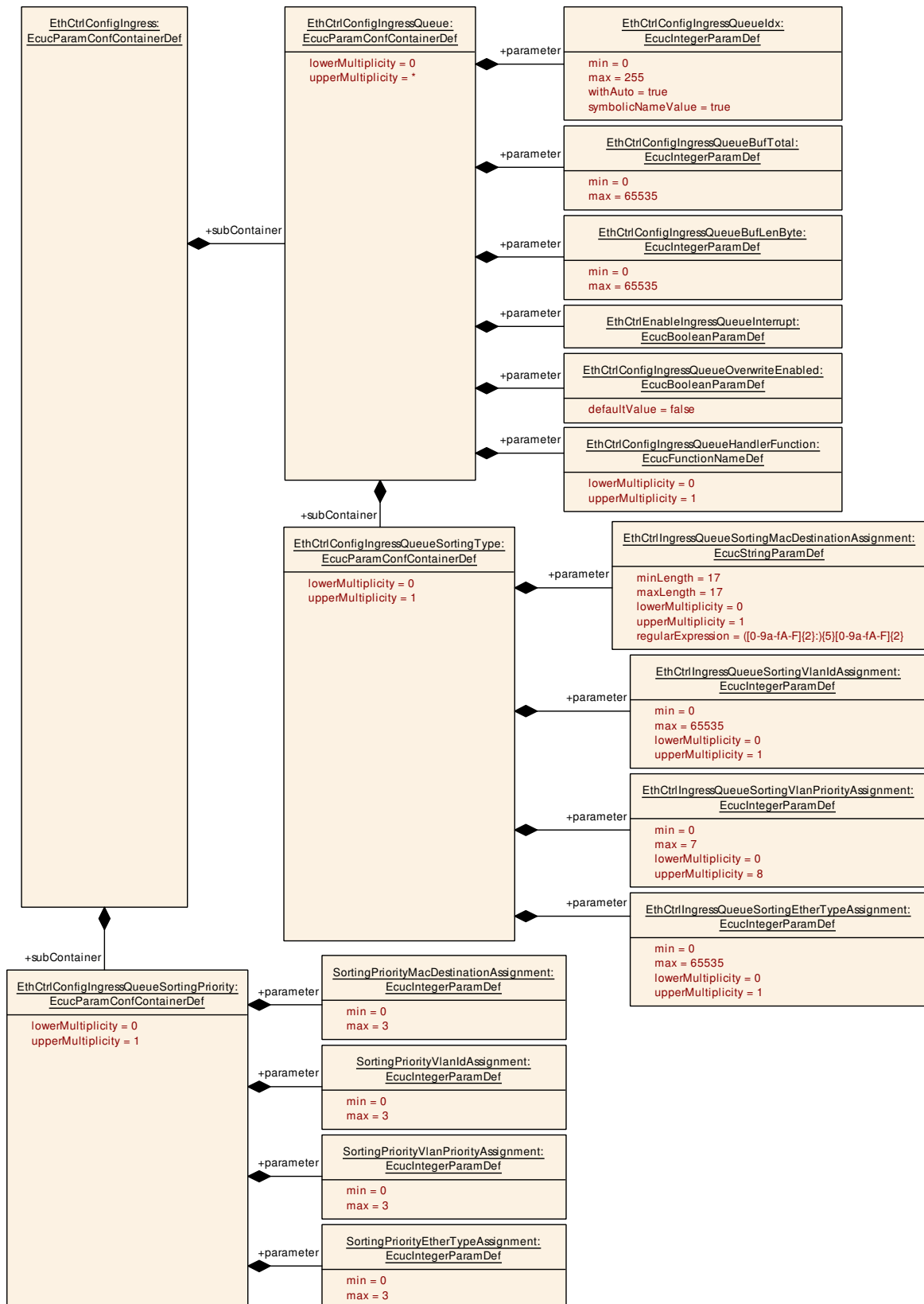


Figure 10.6: Overview EthCtrlConfigIngress configuration - DRAFT

10.2.8 EthCtrlConfigSpiConfiguration

[ECUC_Eth_00074] Definition of EcucParamConfContainerDef EthCtrlConfigSpi Configuration

Status: DRAFT

[

Container Name	EthCtrlConfigSpiConfiguration
Parent Container	EthCtrlConfig
Description	SPI Interface configuration of one Ethernet controller (MACPHY use). Configured only if EthCtrlEnableSpiInterface is set to TRUE. Tags: atp.Status=draft
Multiplicity	0..*
Post-Build Variant Multiplicity	false
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlConfigSpiChunkPayloadSize	1	[ECUC_Eth_00079]
EthCtrlConfigSpiCommRetries	1	[ECUC_Eth_00075]
EthCtrlConfigSpiCommTimeout	1	[ECUC_Eth_00076]
EthCtrlConfigSpiEnableControlDataProtection	1	[ECUC_Eth_00081]
EthCtrlConfigSpiEnableRxCSAlign	1	[ECUC_Eth_00085]
EthCtrlConfigSpiEnableRxCutThrough	0..1	[ECUC_Eth_00082]
EthCtrlConfigSpiEnableRxZeroAlign	1	[ECUC_Eth_00084]
EthCtrlConfigSpiEnableTransmitDataHdrSequence	0..1	[ECUC_Eth_00080]
EthCtrlConfigSpiEnableTxChecksum	0..1	[ECUC_Eth_00086]
EthCtrlConfigSpiEnableTxCutThrough	0..1	[ECUC_Eth_00089]
EthCtrlConfigSpiSelectTimeStamp	0..1	[ECUC_Eth_00087]
EthCtrlConfigSpiTransmitCreditThreshold	1	[ECUC_Eth_00083]

Included Containers		
Container Name	Multiplicity	Dependency
EthCtrlConfigSpiSequence	0..*	Container gives Ethernet controller driver information about one SPI sequence. One SPI sequence used by Ethernet controller driver is in exclusive use for it. No other driver is allowed to access this sequence. Ethernet controller driver may use one sequence to access n Ethernet controller hardware chips of the same type or n sequences are used to access one single Ethernet controller hardware chip. If a Ethernet controller hardware has no SPI interface, there is no instance of this container. Tags: atp.Status=draft

]

[ECUC_Eth_00079] Definition of EcucIntegerParamDef EthCtrlConfigSpiChunkPayloadSize

Status: DRAFT

Parameter Name	EthCtrlConfigSpiChunkPayloadSize		
Parent Container	EthCtrlConfigSpiConfiguration		
Description	Configures the size of the payload chunks which will be transferred over the SPI interface. Note: The chunk is the basic element for data transaction over the SPI which can be a section of an Ethernet frame or management command. The configured value has to be a multiple of 8. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	8 .. 64		
Default value	64		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE.		

[ECUC_Eth_00075] Definition of EcucIntegerParamDef EthCtrlConfigSpiCommRetries

Status: DRAFT

Parameter Name	EthCtrlConfigSpiCommRetries		
Parent Container	EthCtrlConfigSpiConfiguration		
Description	Indicates the maximum number of communication retries in case of a failed SPI communication (applies both to timed out communication and to errors/NACK in the response data). If configured value is '0', no retry is allowed (communication is expected to succeed at first try). Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	—		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE. This parameter exists only if at least one SPI Sequence is referenced.		

[ECUC_Eth_00076] Definition of EcucFloatParamDef EthCtrlConfigSpiCommTimeout

Status: DRAFT

Parameter Name	EthCtrlConfigSpiCommTimeout		
Parent Container	EthCtrlConfigSpiConfiguration		
Description	Indicates the maximum time allowed to the Ethernet controller for replying (either positively or negatively) to a SPI command. Timeout is configured in seconds. Timeout value of '0' means that no specific timeout is to be used by Ethernet controller and the communication is executed at the best of the SPI HW capacity. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 0.1]		
Default value	–		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	–	
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE. This parameter exists only if at least one SPI Sequence is referenced.		

[ECUC_Eth_00081] Definition of EcucBooleanParamDef EthCtrlConfigSpiEnableControlDataProtection

Status: DRAFT

Parameter Name	EthCtrlConfigSpiEnableControlDataProtection		
Parent Container	EthCtrlConfigSpiConfiguration		
Description	Enables the control data protection. When set, all control data written to and read from the MACPHY will be transferred with its complement for detection of bit errors as defined in OA TC6. FALSE: Control data read/write protection is disabled (unprotected). TRUE: Control data read/write protection is enabled (protected). Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

[ECUC_Eth_00085] Definition of EcucBooleanParamDef EthCtrlConfigSpiEnableRxCSAlign

Status: DRAFT

Parameter Name	EthCtrlConfigSpiEnableRxCSAlign		
Parent Container	EthCtrlConfigSpiConfiguration		
Description	Configures the CSn Align Receive frame. TRUE: all received Ethernet frames data shall start at the beginning of the first receive data chunk payload following CSn assertion FALSE: received frames may begin within any receive data chunk of the transaction when this bit is clear. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

[ECUC_Eth_00082] Definition of EcucBooleanParamDef EthCtrlConfigSpiEnableRxCutThrough

Status: DRAFT

Parameter Name	EthCtrlConfigSpiEnableRxCutThrough		
Parent Container	EthCtrlConfigSpiConfiguration		
Description	When supported by the HW, enables the cut through mode of frame from the network to the SPI host. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

[ECUC_Eth_00084] Definition of EcucBooleanParamDef EthCtrlConfigSpiEnableRxZeroAlign

Status: DRAFT

Parameter Name	EthCtrlConfigSpiEnableRxZeroAlign		
Parent Container	EthCtrlConfigSpiConfiguration		
Description	Configures the zero-align receive frame. TRUE: all received Ethernet frames data shall be aligned to start at the beginning of any receive data chunk payload. FALSE: Received frames may begin anywhere within the receive data chunk payload. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

[ECUC_Eth_00080] Definition of EcucBooleanParamDef EthCtrlConfigSpiEnableTransmitDataHdrSequence

Status: DRAFT

Parameter Name	EthCtrlConfigSpiEnableTransmitDataHdrSequence		
Parent Container	EthCtrlConfigSpiConfiguration		
Description	When supported by the HW, enables the transmit data sequence monitoring. FALSE: transmit data header sequence bit monitoring disabled. TRUE: transmit data header sequence bit monitoring enabled. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

[ECUC_Eth_00086] Definition of EcucBooleanParamDef EthCtrlConfigSpiEnableTxChecksum

Status: DRAFT

Parameter Name	EthCtrlConfigSpiEnableTxChecksum		
Parent Container	EthCtrlConfigSpiConfiguration		
Description	When supported by the HW, enables the transmit frame check sequence validation. • TRUE: the final 4 octets of all Ethernet frames received will be validated as an Ethernet FCS. • FALSE : validation deactivated. Note: additionally the MAC must be configured to expect the SPI host to pad frames to the minimum frame size and append the FCS to the frame. The MAC should also be configured to pass the FCS to the SPI host with the received FCS for validation. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

[ECUC_Eth_00089] Definition of EcucBooleanParamDef EthCtrlConfigSpiEnableTxCutThrough

Status: DRAFT

Parameter Name	EthCtrlConfigSpiEnableTxCutThrough		
Parent Container	EthCtrlConfigSpiConfiguration		
Description	When supported by the HW, enables the cut through mode of frame from SPI host to the network. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
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	–	
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

]

[ECUC_Eth_00087] Definition of EcucBooleanParamDef EthCtrlConfigSpiSelectTimeStamp

Status: DRAFT

[

Parameter Name	EthCtrlConfigSpiSelectTimeStamp		
Parent Container	EthCtrlConfigSpiConfiguration		
Description	When timestamp supported by the HW, selects size and format of the timestamps. FALSE: 32-bits timestamps TRUE: 64-bit timestamps Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE AND EthGlobalTimeSupport is set to TRUE.		

]

[ECUC_Eth_00083] Definition of EcucIntegerParamDef EthCtrlConfigSpiTransmitCreditThreshold

Status: DRAFT

[

Parameter Name	EthCtrlConfigSpiTransmitCreditThreshold		
Parent Container	EthCtrlConfigSpiConfiguration		
Description	Configures the minimum of available transmit credit before the writing IRQn is asserted. As per OA TC6, this information is notified by the TXC field. 0 = 1 credit 1 = 4 credits 2 = 8 credits 3 = 16 credits Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 3		
Default value	0		
Post-Build Variant Value	true		





Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE.		

[ECUC_Eth_00077] Definition of EcucParamConfContainerDef EthCtrlConfigSpi Sequence

Status: DRAFT

Container Name	EthCtrlConfigSpiSequence
Parent Container	EthCtrlConfigSpiConfiguration
Description	Container gives Ethernet controller driver information about one SPI sequence. One SPI sequence used by Ethernet controller driver is in exclusive use for it. No other driver is allowed to access this sequence. Ethernet controller driver may use one sequence to access n Ethernet controller hardware chips of the same type or n sequences are used to access one single Ethernet controller hardware chip. If a Ethernet controller hardware has no SPI interface, there is no instance of this container. Tags: atp.Status=draft
Multiplicity	0..*
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlConfigSpiAccessSynchronous	0..1	[ECUC_Eth_00078]
EthCtrlConfigSpiSequenceName	0..*	[ECUC_Eth_00088]

No Included Containers

[ECUC_Eth_00078] Definition of EcucBooleanParamDef EthCtrlConfigSpiAccess Synchronous

Status: DRAFT

Parameter Name	EthCtrlConfigSpiAccessSynchronous
Parent Container	EthCtrlConfigSpiSequence
Description	This parameter is used to define whether the access to the Spi sequence is synchronous or asynchronous. true: SPI access is synchronous. false: SPI access is asynchronous. Tags: atp.Status=draft
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Post-Build Variant Multiplicity	true





Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

[ECUC_Eth_00088] Definition of EcucReferenceDef EthCtrlConfigSpiSequence Name

Status: DRAFT

Parameter Name	EthCtrlConfigSpiSequenceName		
Parent Container	EthCtrlConfigSpiSequence		
Description	Reference to a Spi sequence configuration container. Tags: atp.Status=draft		
Multiplicity	0..*		
Type	Symbolic name reference to SpiSequence		
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	–	
Dependency	This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

10.2.9 EthCtrlMacSecConfig

[ECUC_Eth_00151] Definition of EcucParamConfContainerDef EthCtrlMacSec Config

Container Name	EthCtrlMacSecConfig		
Parent Container	EthCtrlConfig		
Description	This container contains the configuration of the MACsec functionality according to IEEE802.1AE (MACsec protocol) and IEEE802.1X (MACsec Key Agreement protocol) which is supported by this Ethernet controller hardware and used if MACsec protection for this Ethernet controller is enabled (EthCtrlMacSecEnabled set to TRUE).		
Multiplicity	0..1		





Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlMacSecBypassDestinationMacAddress	0..255	[ECUC_Eth_00152]
EthCtrlMacSecBypassEtherType	0..255	[ECUC_Eth_00154]
EthCtrlMacSecBypassVlan	0..255	[ECUC_Eth_00153]

No Included Containers

[ECUC_Eth_00152] Definition of EcucStringParamDef EthCtrlMacSecBypassDestinationMacAddress

Parameter Name	EthCtrlMacSecBypassDestinationMacAddress		
Parent Container	EthCtrlMacSecConfig		
Description	Bypassed destination MAC address (48-bit physical address). Ethernet frames, which carrying the configured destination MAC address, shall bypass the MACsec protection handling.		
Multiplicity	0..255		
Type	EcucStringParamDef		
Default value	–		
Length	17-17		
Regular Expression	([0-9a-fA-F]{2:}){5}[0-9a-fA-F]{2}		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00154] Definition of EcucIntegerParamDef EthCtrlMacSecBypassEtherType

Parameter Name	EthCtrlMacSecBypassEtherType		
Parent Container	EthCtrlMacSecConfig		
Description	Bypassed EtherType. Ethernet frames, which carrying the configured EtherType, shall bypass the MACsec protection handling.		





Multiplicity	0..255		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00153] Definition of EcucIntegerParamDef EthCtrlMacSecBypassVlan

Parameter Name	EthCtrlMacSecBypassVlan		
Parent Container	EthCtrlMacSecConfig		
Description	Bypassed VLAN-ID. Ethernet frames, which carrying the configured VLAN-ID, shall bypass the MACsec protection handling. Please note: VLAN-ID 0 is interpreted as no VLAN, i.e. untagged Ethernet frames bypass the MACsec protection handling.		
Multiplicity	0..255		
Type	EcucIntegerParamDef		
Range	0 .. 4094		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

10.2.10 EthDemEventParameterRefs

[ECUC_Eth_00016] Definition of EcucParamConfContainerDef EthDemEventParameterRefs

Container Name	EthDemEventParameterRefs
Parent Container	EthCtrlConfig
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
Multiplicity	0..1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
ETH_E_ACCESS	0..1	[ECUC_Eth_00017]
ETH_E_ALIGNMENT	0..1	[ECUC_Eth_00026]
ETH_E_CRC	0..1	[ECUC_Eth_00023]
ETH_E_LATECOLLISION	0..1	[ECUC_Eth_00029]
ETH_E_MULTIPLECOLLISION	0..1	[ECUC_Eth_00028]
ETH_E_OVERSIZEFRAME	0..1	[ECUC_Eth_00025]
ETH_E_RX_FRAMES_LOST	0..1	[ECUC_Eth_00021]
ETH_E_SINGLECOLLISION	0..1	[ECUC_Eth_00027]
ETH_E_UNDERSIZEFRAME	0..1	[ECUC_Eth_00024]

No Included Containers

[[ECUC_Eth_00017](#)] Definition of EcucReferenceDef [ETH_E_ACCESS](#) [

Parameter Name	ETH_E_ACCESS		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Controller access failed" has occurred.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00026] Definition of EcucReferenceDef ETH_E_ALIGNMENT [

Parameter Name	ETH_E_ALIGNMENT		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Alignment Error" has occurred.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00023] Definition of EcucReferenceDef ETH_E_CRC [

Parameter Name	ETH_E_CRC		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "CRC Failure" has occurred.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00029] Definition of EcucReferenceDef ETH_E_LATECOLLISION [

Parameter Name	ETH_E_LATECOLLISION		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Late Collisions" has occurred.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE



△

Value Configuration Class	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00028] Definition of EcucReferenceDef ETH_E_MULTIPLECOLLISION

Parameter Name	ETH_E_MULTIPLECOLLISION		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Multiple Collisions" has occurred.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00025] Definition of EcucReferenceDef ETH_E_OVERSIZEFRAME

Parameter Name	ETH_E_OVERSIZEFRAME		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Oversized Frame" has occurred.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00021] Definition of EcucReferenceDef ETH_E_RX_FRAMES_LOST

Parameter Name	ETH_E_RX_FRAMES_LOST		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "receive frames lost" has occurred.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00027] Definition of EcucReferenceDef ETH_E_SINGLECOLLISION

Parameter Name	ETH_E_SINGLECOLLISION		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Single Collisions" has occurred.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Eth_00024] Definition of EcucReferenceDef ETH_E_UNDERSIZEFRAME

Parameter Name	ETH_E_UNDERSIZEFRAME		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Undersized Frame" has occurred.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		





Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

10.2.11 EthGeneral

[ECUC_Eth_00001] Definition of EcucParamConfContainerDef EthGeneral [

Container Name	EthGeneral
Parent Container	Eth
Description	General configuration of Ethernet Driver module
Multiplicity	1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthDevErrorDetect	1	[ECUC_Eth_00003]
EthGetCounterValuesApi	1	[ECUC_Eth_00035]
EthGetRxStatsApi	1	[ECUC_Eth_00072]
EthGetTxErrorCounterValuesApi	1	[ECUC_Eth_00061]
EthGetTxStatsApi	1	[ECUC_Eth_00060]
EthGlobalTimeSupport	1	[ECUC_Eth_00037]
EthIndex	1	[ECUC_Eth_00018]
EthMainFunctionPeriod	1	[ECUC_Eth_00022]
EthPhcSupport	1	[ECUC_Eth_00107]
EthVersionInfoApi	1	[ECUC_Eth_00004]
EthEcucPartitionRef	0..*	[ECUC_Eth_00064]

Included Containers		
Container Name	Multiplicity	Dependency
EthCtrlOffloading	1	Configuration of hardware offloading features.

]

[ECUC_Eth_00003] Definition of EcucBooleanParamDef EthDevErrorDetect [

Parameter Name	EthDevErrorDetect		
Parent Container	EthGeneral		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00035] Definition of EcucBooleanParamDef EthGetCounterValues Api [

Parameter Name	EthGetCounterValuesApi		
Parent Container	EthGeneral		
Description	Enables / Disables Eth_GetCounterValues API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00072] Definition of EcucBooleanParamDef EthGetRxStatsApi [

Parameter Name	EthGetRxStatsApi		
Parent Container	EthGeneral		
Description	Enables/Disables Eth_GetRxStats API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00061] Definition of EcucBooleanParamDef EthGetTxErrorCounterValuesApi

Parameter Name	EthGetTxErrorCounterValuesApi		
Parent Container	EthGeneral		
Description	Enables/Disables Eth_GetTxErrorCounterValues API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00060] Definition of EcucBooleanParamDef EthGetTxStatsApi

Parameter Name	EthGetTxStatsApi		
Parent Container	EthGeneral		
Description	Enables/Disables Eth_GetTxStats API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00037] Definition of EcucBooleanParamDef EthGlobalTimeSupport

Parameter Name	EthGlobalTimeSupport		
Parent Container	EthGeneral		
Description	Enables/Disables the GlobalTime APIs used amongst others by Global Time Synchronization over Ethernet.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00018] Definition of EcucIntegerParamDef EthIndex [

Parameter Name	EthIndex		
Parent Container	EthGeneral		
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

]

[ECUC_Eth_00022] Definition of EcucFloatParamDef EthMainFunctionPeriod [

Parameter Name	EthMainFunctionPeriod		
Parent Container	EthGeneral		
Description	Specifies the period of main function Eth_MainFunction in seconds. Ethernet driver does not require this information but the BSW scheduler.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

]

[ECUC_Eth_00107] Definition of EcucBooleanParamDef EthPhcSupport

Status: DRAFT

[

Parameter Name	EthPhcSupport		
Parent Container	EthGeneral		
Description	Enables/Disables the PTP HW Clock (PHC). Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	





	Post-build time	–	
Dependency			

[ECUC_Eth_00004] Definition of EcucBooleanParamDef EthVersionInfoApi

Parameter Name	EthVersionInfoApi		
Parent Container	EthGeneral		
Description	Enables / Disables version info API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00064] Definition of EcucReferenceDef EthEcucPartitionRef

Parameter Name	EthEcucPartitionRef		
Parent Container	EthGeneral		
Description	Maps the Ethernet driver to zero or multiple ECUC partitions to make the modules API available in this partition.		
Multiplicity	0..*		
Type	Reference to EcucPartition		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
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	–	
Dependency			

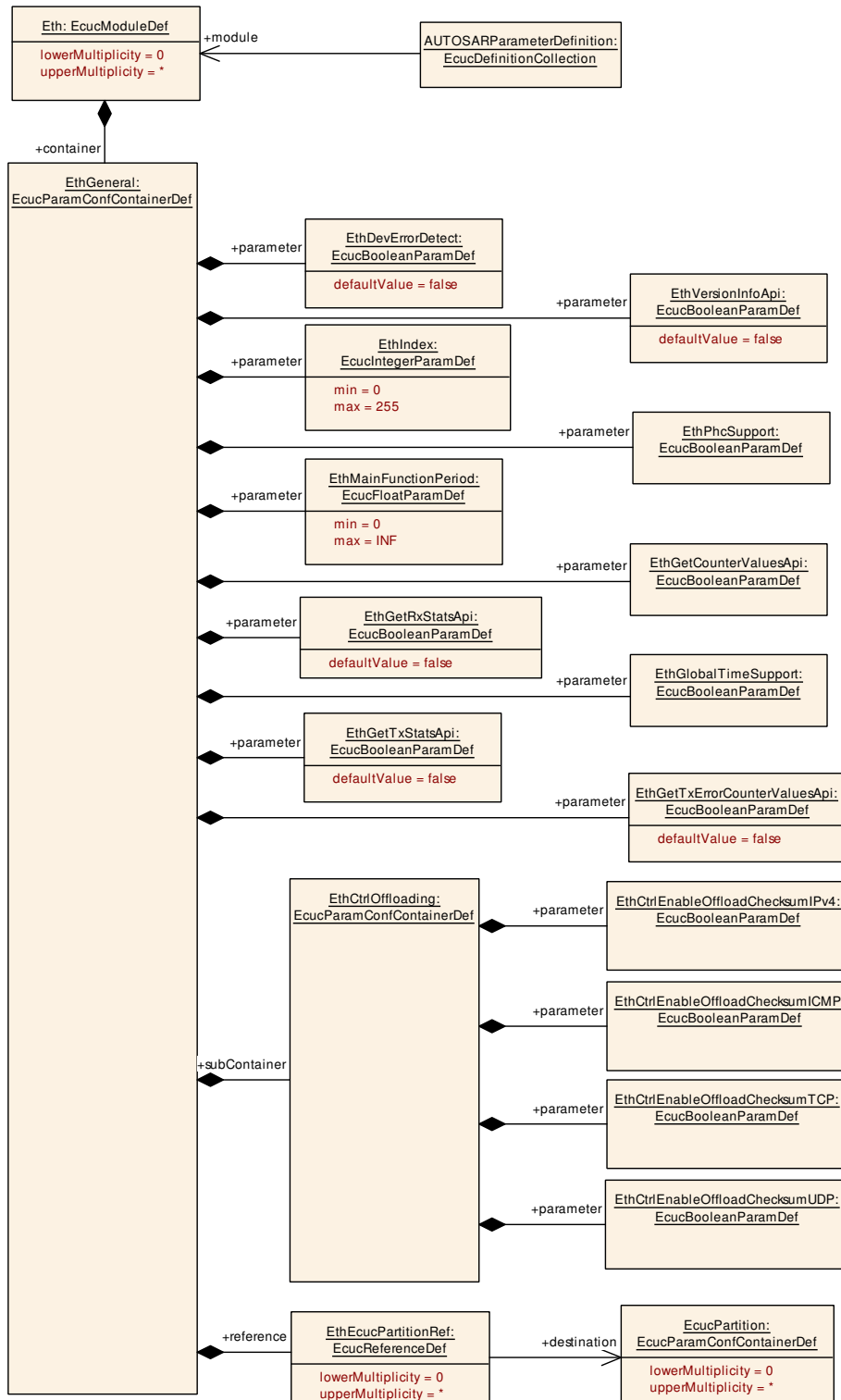


Figure 10.7: Overview Eth general configuration

10.2.11.1 EthCtrlOffloading

[ECUC_Eth_00030] Definition of EcucParamConfContainerDef EthCtrlOffloading

Container Name	EthCtrlOffloading	
Parent Container	EthGeneral	
Description	Configuration of hardware offloading features.	
Multiplicity	1	
Configuration Parameters		

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthCtrlEnableOffloadChecksumICMP	1	[ECUC_Eth_00032]
EthCtrlEnableOffloadChecksumIPv4	1	[ECUC_Eth_00031]
EthCtrlEnableOffloadChecksumTCP	1	[ECUC_Eth_00033]
EthCtrlEnableOffloadChecksumUDP	1	[ECUC_Eth_00034]

No Included Containers

[ECUC_Eth_00032] Definition of EcucBooleanParamDef EthCtrlEnableOffloadChecksumICMP

Parameter Name	EthCtrlEnableOffloadChecksumICMP		
Parent Container	EthCtrlOffloading		
Description	Enables / Disables hardware offloading for ICMP checksums.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00031] Definition of EcucBooleanParamDef EthCtrlEnableOffloadChecksumIPv4

Parameter Name	EthCtrlEnableOffloadChecksumIPv4		
Parent Container	EthCtrlOffloading		
Description	Enables / Disables hardware offloading for IPv4 checksums.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		





Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00033] Definition of EcucBooleanParamDef EthCtrlEnableOffloadChecksumTCP

Parameter Name	EthCtrlEnableOffloadChecksumTCP		
Parent Container	EthCtrlOffloading		
Description	Enables / Disables hardware offloading for TCP checksums.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Eth_00034] Definition of EcucBooleanParamDef EthCtrlEnableOffloadChecksumUDP

Parameter Name	EthCtrlEnableOffloadChecksumUDP		
Parent Container	EthCtrlOffloading		
Description	Enables / Disables hardware offloading for UDP checksums.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

10.3 Published Information

For details refer to [3] Chapter 10.3 “*Published Information*”.

A Not applicable requirements

No items.

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R25-11

B.1.1 Added Specification Items in R25-11

[\[ECUC_Eth_00149\]](#) [\[ECUC_Eth_00150\]](#) [\[ECUC_Eth_00151\]](#) [\[ECUC_Eth_00152\]](#)
[\[ECUC_Eth_00153\]](#) [\[ECUC_Eth_00154\]](#) [\[SWS_Eth_00418\]](#) [\[SWS_Eth_00419\]](#)
[\[SWS_Eth_00420\]](#) [\[SWS_Eth_00421\]](#) [\[SWS_Eth_00422\]](#) [\[SWS_Eth_00423\]](#) [\[SWS_Eth_00424\]](#)
[\[SWS_Eth_00425\]](#) [\[SWS_Eth_00426\]](#) [\[SWS_Eth_00427\]](#) [\[SWS_Eth_00428\]](#) [\[SWS_Eth_00429\]](#)
[\[SWS_Eth_00430\]](#) [\[SWS_Eth_00431\]](#) [\[SWS_Eth_00432\]](#) [\[SWS_Eth_00433\]](#) [\[SWS_Eth_00434\]](#)
[\[SWS_Eth_00435\]](#) [\[SWS_Eth_00436\]](#) [\[SWS_Eth_00437\]](#) [\[SWS_Eth_00438\]](#) [\[SWS_Eth_00439\]](#)
[\[SWS_Eth_00440\]](#) [\[SWS_Eth_00441\]](#) [\[SWS_Eth_00442\]](#) [\[SWS_Eth_00443\]](#) [\[SWS_Eth_00444\]](#)
[\[SWS_Eth_00445\]](#) [\[SWS_Eth_00446\]](#) [\[SWS_Eth_00447\]](#) [\[SWS_Eth_00448\]](#) [\[SWS_Eth_91028\]](#)
[\[SWS_Eth_91029\]](#) [\[SWS_Eth_91030\]](#) [\[SWS_Eth_91031\]](#) [\[SWS_Eth_91032\]](#) [\[SWS_Eth_91033\]](#)
[\[SWS_Eth_91034\]](#) [\[SWS_Eth_91035\]](#) [\[SWS_Eth_91036\]](#) [\[SWS_Eth_91037\]](#)
[\[SWS_Eth_91038\]](#) [\[SWS_Eth_91039\]](#)

B.1.2 Changed Specification Items in R25-11

[\[ECUC_Eth_00006\]](#) [\[ECUC_Eth_00040\]](#) [\[ECUC_Eth_00046\]](#) [\[ECUC_Eth_00056\]](#)
[\[ECUC_Eth_00081\]](#) [\[ECUC_Eth_00090\]](#) [\[ECUC_Eth_00091\]](#) [\[ECUC_Eth_00095\]](#)
[\[ECUC_Eth_00096\]](#) [\[ECUC_Eth_00100\]](#) [\[ECUC_Eth_00101\]](#) [\[ECUC_Eth_00106\]](#)
[\[SWS_Eth_00026\]](#) [\[SWS_Eth_00058\]](#) [\[SWS_Eth_00064\]](#) [\[SWS_Eth_00120\]](#) [\[SWS_Eth_00129\]](#)
[\[SWS_Eth_00132\]](#) [\[SWS_Eth_00134\]](#) [\[SWS_Eth_00144\]](#) [\[SWS_Eth_00147\]](#) [\[SWS_Eth_00150\]](#)
[\[SWS_Eth_00152\]](#) [\[SWS_Eth_00173\]](#) [\[SWS_Eth_00195\]](#) [\[SWS_Eth_00199\]](#) [\[SWS_Eth_00213\]](#)
[\[SWS_Eth_00219\]](#) [\[SWS_Eth_00220\]](#) [\[SWS_Eth_00221\]](#) [\[SWS_Eth_00222\]](#) [\[SWS_Eth_00223\]](#)
[\[SWS_Eth_00224\]](#) [\[SWS_Eth_00225\]](#) [\[SWS_Eth_00244\]](#) [\[SWS_Eth_00245\]](#) [\[SWS_Eth_00247\]](#)
[\[SWS_Eth_00405\]](#) [\[SWS_Eth_00409\]](#) [\[SWS_Eth_91009\]](#) [\[SWS_Eth_91012\]](#) [\[SWS_Eth_91013\]](#)
[\[SWS_Eth_91019\]](#) [\[SWS_Eth_91024\]](#) [\[SWS_Eth_91025\]](#) [\[SWS_Eth_91026\]](#)

B.1.3 Deleted Specification Items in R25-11

[\[ECUC_Eth_00041\]](#) [\[ECUC_Eth_00042\]](#) [\[ECUC_Eth_00043\]](#) [\[ECUC_Eth_00044\]](#)
[\[ECUC_Eth_00045\]](#) [\[ECUC_Eth_00047\]](#) [\[ECUC_Eth_00048\]](#) [\[ECUC_Eth_00049\]](#)
[\[ECUC_Eth_00050\]](#) [\[ECUC_Eth_00051\]](#) [\[ECUC_Eth_00057\]](#) [\[ECUC_Eth_00058\]](#)

[ECUC_Eth_00059] [ECUC_Eth_00069] [ECUC_Eth_00070] [ECUC_Eth_00071]
[ECUC_Eth_00102] [ECUC_Eth_00104] [ECUC_Eth_00105] [SWS_Eth_00004]
[SWS_Eth_00096] [SWS_Eth_00176] [SWS_Eth_00177] [SWS_Eth_00178] [SWS_Eth_00179]
[SWS_Eth_00180] [SWS_Eth_00181] [SWS_Eth_00185] [SWS_Eth_00210] [SWS_Eth_00262]
[SWS_Eth_00263] [SWS_Eth_00264] [SWS_Eth_00265] [SWS_Eth_00266] [SWS_Eth_00267]
[SWS_Eth_00268] [SWS_Eth_00269] [SWS_Eth_00270] [SWS_Eth_00271] [SWS_Eth_00272]
[SWS_Eth_00297] [SWS_Eth_00298] [SWS_Eth_00299] [SWS_Eth_00300] [SWS_Eth_00302]
[SWS_Eth_00303] [SWS_Eth_00304] [SWS_Eth_00305] [SWS_Eth_00306] [SWS_Eth_00307]
[SWS_Eth_00308] [SWS_Eth_00309] [SWS_Eth_00310] [SWS_Eth_00311] [SWS_Eth_00312]
[SWS_Eth_91015] [SWS_Eth_91016]

B.1.4 Added Constraints in R25-11

[SWS_Eth_CONSTR_00023]

B.1.5 Changed Constraints in R25-11

none

B.1.6 Deleted Constraints in R25-11

[SWS_Eth_CONSTR_00002] [SWS_Eth_CONSTR_00003]

B.2 Traceable item history of this document according to AUTOSAR Release R24-11

B.2.1 Added Specification Items in R24-11

[ECUC_Eth_00136] [ECUC_Eth_00137] [ECUC_Eth_00138] [ECUC_Eth_00139]
[ECUC_Eth_00140] [ECUC_Eth_00141] [ECUC_Eth_00142] [ECUC_Eth_00143]
[ECUC_Eth_00144] [ECUC_Eth_00145] [ECUC_Eth_00146] [ECUC_Eth_00147]
[ECUC_Eth_00148] [SWS_Eth_00390] [SWS_Eth_00391] [SWS_Eth_00393] [SWS_Eth_00394]
[SWS_Eth_00395] [SWS_Eth_00396] [SWS_Eth_00397] [SWS_Eth_00398] [SWS_Eth_00399]
[SWS_Eth_00400] [SWS_Eth_00401] [SWS_Eth_00402] [SWS_Eth_00403] [SWS_Eth_00404]
[SWS_Eth_00405] [SWS_Eth_00406] [SWS_Eth_00407] [SWS_Eth_00408] [SWS_Eth_00409]
[SWS_Eth_00410] [SWS_Eth_00411] [SWS_Eth_00412] [SWS_Eth_00413] [SWS_Eth_00414]
[SWS_Eth_00415] [SWS_Eth_00416] [SWS_Eth_00417] [SWS_Eth_91025] [SWS_Eth_91026]
[SWS_Eth_91027]

B.2.2 Changed Specification Items in R24-11

[ECUC_Eth_00001] [ECUC_Eth_00006] [ECUC_Eth_00046] [ECUC_Eth_00053]
[ECUC_Eth_00062] [ECUC_Eth_00063] [ECUC_Eth_00064] [ECUC_Eth_00086]
[ECUC_Eth_00087] [ECUC_Eth_00088] [ECUC_Eth_00090] [ECUC_Eth_00100]
[ECUC_Eth_00101] [ECUC_Eth_00135] [SWS_Eth_00016] [SWS_Eth_00026]
[SWS_Eth_00058] [SWS_Eth_00064] [SWS_Eth_00176] [SWS_Eth_00302] [SWS_Eth_00307]
[SWS_Eth_00313] [SWS_Eth_91009] [SWS_Eth_91014]

B.2.3 Deleted Specification Items in R24-11

[ECUC_Eth_00002] [ECUC_Eth_00094] [ECUC_Eth_00103] [SWS_Eth_00006]
[SWS_Eth_00007] [SWS_Eth_00012] [SWS_Eth_00013] [SWS_Eth_00014] [SWS_Eth_00043]
[SWS_Eth_00044] [SWS_Eth_00048] [SWS_Eth_00049] [SWS_Eth_00050] [SWS_Eth_00054]
[SWS_Eth_00055] [SWS_Eth_00056] [SWS_Eth_00060] [SWS_Eth_00061] [SWS_Eth_00066]
[SWS_Eth_00067] [SWS_Eth_00068] [SWS_Eth_00081] [SWS_Eth_00082] [SWS_Eth_00083]
[SWS_Eth_00084] [SWS_Eth_00085] [SWS_Eth_00090] [SWS_Eth_00091] [SWS_Eth_00093]
[SWS_Eth_00097] [SWS_Eth_00098] [SWS_Eth_00103] [SWS_Eth_00104] [SWS_Eth_00136]
[SWS_Eth_00140] [SWS_Eth_00141] [SWS_Eth_00142] [SWS_Eth_00164] [SWS_Eth_00165]
[SWS_Eth_00166] [SWS_Eth_00182] [SWS_Eth_00183] [SWS_Eth_00184] [SWS_Eth_00187]
[SWS_Eth_00188] [SWS_Eth_00191] [SWS_Eth_00192] [SWS_Eth_00193] [SWS_Eth_00196]
[SWS_Eth_00197] [SWS_Eth_00198] [SWS_Eth_00228] [SWS_Eth_00229] [SWS_Eth_00230]
[SWS_Eth_00235] [SWS_Eth_00236] [SWS_Eth_00237] [SWS_Eth_00248] [SWS_Eth_00249]
[SWS_Eth_00250] [SWS_Eth_00252] [SWS_Eth_00253] [SWS_Eth_00254] [SWS_Eth_00259]
[SWS_Eth_00288] [SWS_Eth_00291] [SWS_Eth_00292] [SWS_Eth_00293] [SWS_Eth_00294]
[SWS_Eth_00345] [SWS_Eth_00346] [SWS_Eth_00347] [SWS_Eth_00348] [SWS_Eth_00352]
[SWS_Eth_00353] [SWS_Eth_00354] [SWS_Eth_00358] [SWS_Eth_00359] [SWS_Eth_00360]
[SWS_Eth_00361] [SWS_Eth_00362] [SWS_Eth_00366] [SWS_Eth_00367] [SWS_Eth_00369]
[SWS_Eth_00370] [SWS_Eth_00371]

B.2.4 Added Constraints in R24-11

[SWS_Eth_CONSTR_00013] [SWS_Eth_CONSTR_00014] [SWS_Eth_CONSTR_00015]
[SWS_Eth_CONSTR_00016] [SWS_Eth_CONSTR_00017] [SWS_Eth_CONSTR_00018]
[SWS_Eth_CONSTR_00019] [SWS_Eth_CONSTR_00020] [SWS_Eth_CONSTR_00021]
[SWS_Eth_CONSTR_00022]

B.2.5 Changed Constraints in R24-11

[SWS_Eth_CONSTR_00001]

B.2.6 Deleted Constraints in R24-11

[SWS_Eth_CONSTR_00004] [SWS_Eth_CONSTR_00009]

B.3 Traceable item history of this document according to AUTOSAR Release R23-11

B.3.1 Added Specification Items in R23-11

[SWS_Eth_00313] [SWS_Eth_00314] [SWS_Eth_00315] [SWS_Eth_00316] [SWS_Eth_00317] [SWS_Eth_00318] [SWS_Eth_00319] [SWS_Eth_00320] [SWS_Eth_00321] [SWS_Eth_00322] [SWS_Eth_00323] [SWS_Eth_00324] [SWS_Eth_00325] [SWS_Eth_00327] [SWS_Eth_00328] [SWS_Eth_00329] [SWS_Eth_00331] [SWS_Eth_00332] [SWS_Eth_00333] [SWS_Eth_00334] [SWS_Eth_00335] [SWS_Eth_00336] [SWS_Eth_00337] [SWS_Eth_00339] [SWS_Eth_00340] [SWS_Eth_00341] [SWS_Eth_00342] [SWS_Eth_00343] [SWS_Eth_00344] [SWS_Eth_00345] [SWS_Eth_00346] [SWS_Eth_00347] [SWS_Eth_00348] [SWS_Eth_00349] [SWS_Eth_00350] [SWS_Eth_00351] [SWS_Eth_00352] [SWS_Eth_00353] [SWS_Eth_00354] [SWS_Eth_00355] [SWS_Eth_00357] [SWS_Eth_00358] [SWS_Eth_00359] [SWS_Eth_00360] [SWS_Eth_00361] [SWS_Eth_00362] [SWS_Eth_00363] [SWS_Eth_00364] [SWS_Eth_00365] [SWS_Eth_00366] [SWS_Eth_00367] [SWS_Eth_00368] [SWS_Eth_00369] [SWS_Eth_00370] [SWS_Eth_00371] [SWS_Eth_00372] [SWS_Eth_00373] [SWS_Eth_00374] [SWS_Eth_00375] [SWS_Eth_00376] [SWS_Eth_00377] [SWS_Eth_00378] [SWS_Eth_00379] [SWS_Eth_00387] [SWS_Eth_91015] [SWS_Eth_91016] [SWS_Eth_91017] [SWS_Eth_91018] [SWS_Eth_91019] [SWS_Eth_91020] [SWS_Eth_91021] [SWS_Eth_91022] [SWS_Eth_91023] [SWS_Eth_91024]

B.3.2 Changed Specification Items in R23-11

[SWS_Eth_00016] [SWS_Eth_00026] [SWS_Eth_00096] [SWS_Eth_00119] [SWS_Eth_00176] [SWS_Eth_00177] [SWS_Eth_00178] [SWS_Eth_00179] [SWS_Eth_00180] [SWS_Eth_00181] [SWS_Eth_00182] [SWS_Eth_00183] [SWS_Eth_00184] [SWS_Eth_00185] [SWS_Eth_00190] [SWS_Eth_00195] [SWS_Eth_00210] [SWS_Eth_00234] [SWS_Eth_00262] [SWS_Eth_00273] [SWS_Eth_00274] [SWS_Eth_00278] [SWS_Eth_00279] [SWS_Eth_00287] [SWS_Eth_00290] [SWS_Eth_00294] [SWS_Eth_91014]

B.3.3 Deleted Specification Items in R23-11

none

B.3.4 Added Constraints in R23-11

[SWS_Eth_CONSTR_00004] [SWS_Eth_CONSTR_00005] [SWS_Eth_CONSTR_00006] [SWS_Eth_CONSTR_00007] [SWS_Eth_CONSTR_00008] [SWS_Eth_CONSTR_00009] [SWS_Eth_CONSTR_00010] [SWS_Eth_CONSTR_00011] [SWS_Eth_CONSTR_00012]

B.3.5 Changed Constraints in R23-11

none

B.3.6 Deleted Constraints in R23-11

none