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	R23-11

Document Change History

Date	Release	Changed by	Description
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	• Entrency 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 adaptions
2016-11-30	4.3.0	AUTOSAR Release Management	• Quality of Service (QoS) suppor • 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

Contents

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.

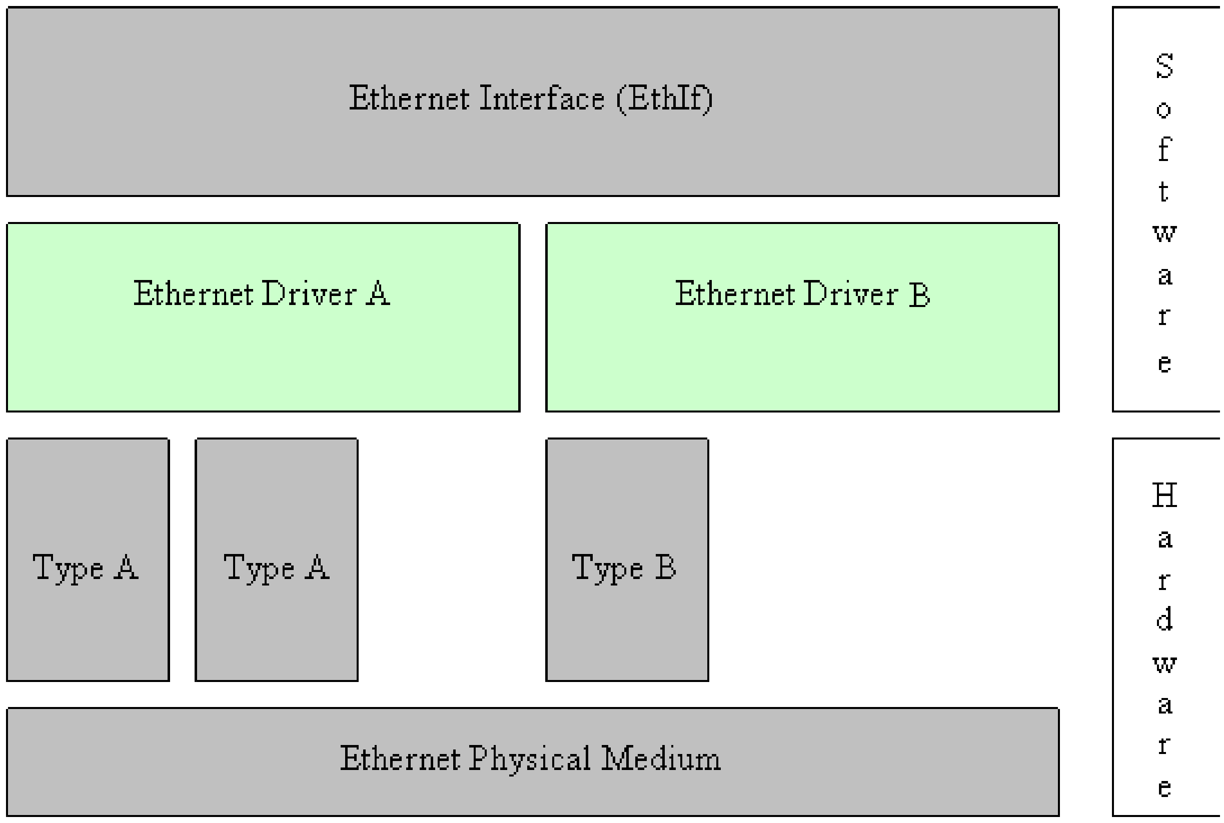


Figure 1.1: Ethernet stack module overview

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

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_SRS_SPALGeneral

[8] Specification of ECU State Manager
AUTOSAR_CP_SWS_ECUStateManager

[9] Requirements on Ethernet Support in AUTOSAR
AUTOSAR_CP_SRS_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 802.1as-2011
https://standards.ieee.org/standard/802_1AS-2011.html

[14] Explanation of Time Sensitive Network features
AUTOSAR_FO_EXP_TimeSensitiveNetworkFeatures

[15] IEEE 802.3-2015
https://www.ieee802.org/3/

[16] STD 59 RFC 2819
https://www.rfc-editor.org/info/rfc2819

3.2 Related standards and norms

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.

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:

- 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]{DRAFT} ⌈If the Ethernet controller is on-chip, the Eth module shall not use any service of other drivers.⌋*(SRS_BSW_00005)*

Note: Not in case of MACPHY

[SWS_Eth_00283]{DRAFT} ⌈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.⌋*(SRS_BSW_00377)*

[SWS_Eth_00284]{DRAFT} ⌈The Mcu module (SPAL see *SPAL General*[7]) shall configure register settings that are "shared" with other modules.⌋*(SRS_BSW_00005)*

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

[SWS_Eth_00285]{DRAFT} ⌈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).⌋*(SRS_BSW_00005)*

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
[RS_TS_20075]	Rate Ratio Calculation	[SWS_Eth_91015] [SWS_Eth_91016]
[SRS_BSW_00005]	Modules of the μC 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_00248] [SWS_Eth_00252] [SWS_Eth_00292] [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_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_Eth_00249] [SWS_Eth_00250] [SWS_Eth_00253] [SWS_Eth_00254] [SWS_Eth_00293] [SWS_Eth_00294]
[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_CONSTR_00004] [SWS_Eth_CONSTR_00005] [SWS_Eth_CONSTR_00006] [SWS_Eth_CONSTR_00007] [SWS_Eth_CONSTR_00008] [SWS_Eth_CONSTR_00009]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_Eth_00249] [SWS_Eth_00250] [SWS_Eth_00253] [SWS_Eth_00254] [SWS_Eth_00293] [SWS_Eth_00294]
[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_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] [SWS_Eth_CONSTR_00004] [SWS_Eth_CONSTR_00005] [SWS_Eth_CONSTR_00006] [SWS_Eth_CONSTR_00007] [SWS_Eth_CONSTR_00008] [SWS_Eth_CONSTR_00009]
[SRS_BSW_00406]	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	[SWS_Eth_00350]
[SRS_BSW_00416]	The sequence of modules to be initialized shall be configurable	[SWS_Eth_00248] [SWS_Eth_00252] [SWS_Eth_00292]
[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_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]

▽

△

Requirement	Description	Satisfied by
[SRS_Eth_00146]	The Ethernet Driver shall provide 10BASE-T1S support	[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_00279] [SWS_Eth_00287] [SWS_Eth_00288] [SWS_Eth_00289] [SWS_Eth_00290] [SWS_Eth_00291] [SWS_Eth_00295] [SWS_Eth_00297] [SWS_Eth_00298] [SWS_Eth_00299] [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_CONSTR_00002] [SWS_Eth_CONSTR_00003]
[SRS_Eth_00147]	The Ethernet Driver shall support SPI	[SWS_Eth_00287] [SWS_Eth_00288] [SWS_Eth_00290] [SWS_Eth_00291] [SWS_Eth_00295] [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_00288] [SWS_Eth_00289] [SWS_Eth_00290] [SWS_Eth_00291]
[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]
[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_CONSTR_00004] [SWS_Eth_CONSTR_00005] [SWS_Eth_CONSTR_00006] [SWS_Eth_CONSTR_00007] [SWS_Eth_CONSTR_00008] [SWS_Eth_CONSTR_00009]
[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]

▽

△

Requirement	Description	Satisfied by
[SRS_Eth_00176]	The Ethernet Interface shall support control of pulse per second signal generation	[SWS_Eth_91021]

Table 6.1: RequirementsTracing

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.

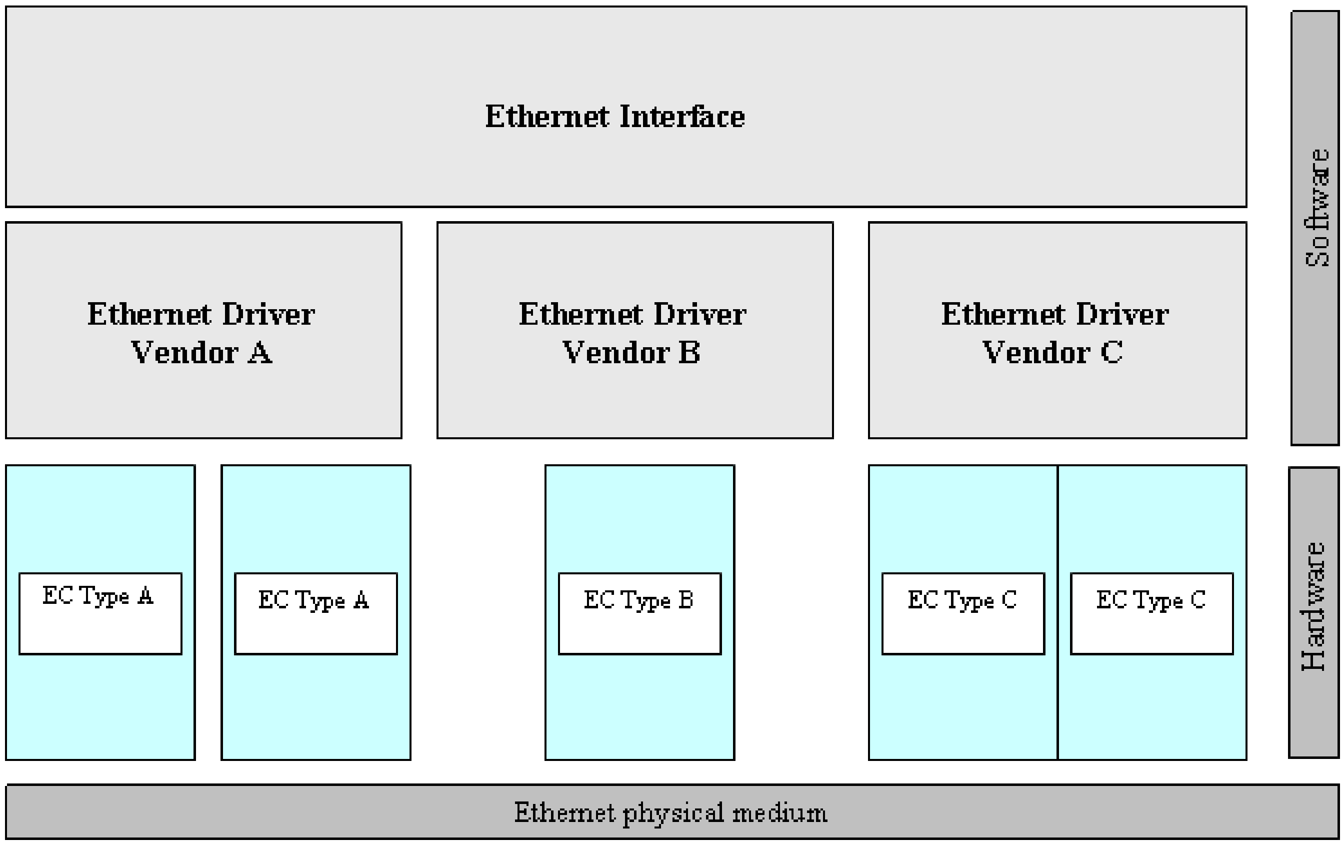


Figure 7.1: Basic Structure of the Ethernet BSW stack

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 (Figure 7.2). The Ethernet Driver ask the Switch Driver for a special treatment to ensure that the current Ethernet frame could be managed in the Switch later on.

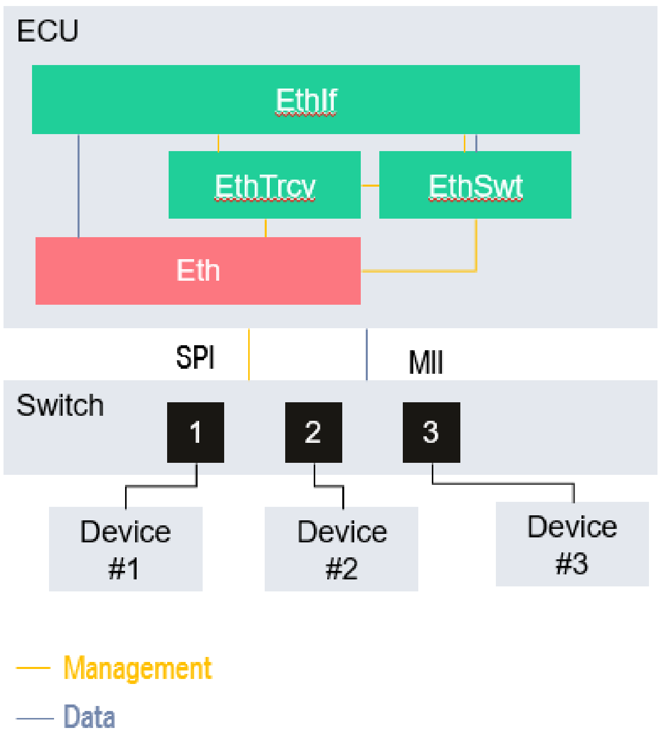


Figure 7.2: HW/SW basic structure including Switch device

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).

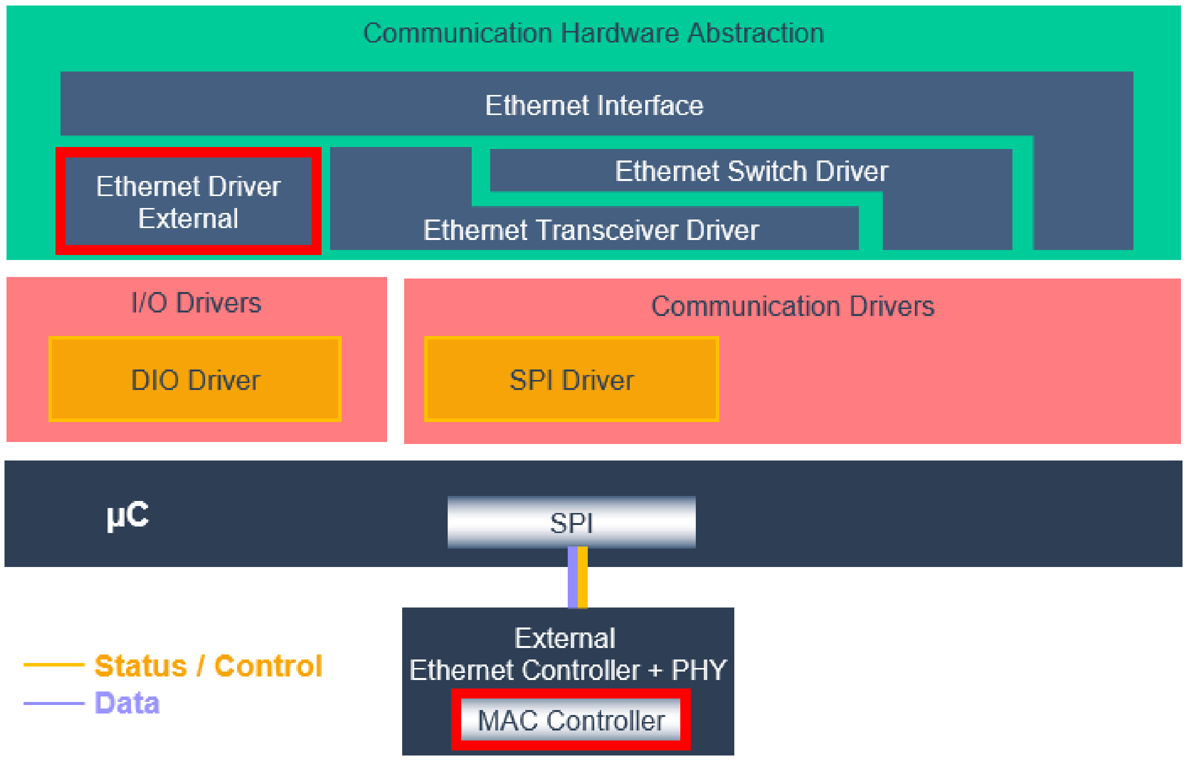


Figure 7.3: External MAC Controller

7.1.3 Indexing scheme

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

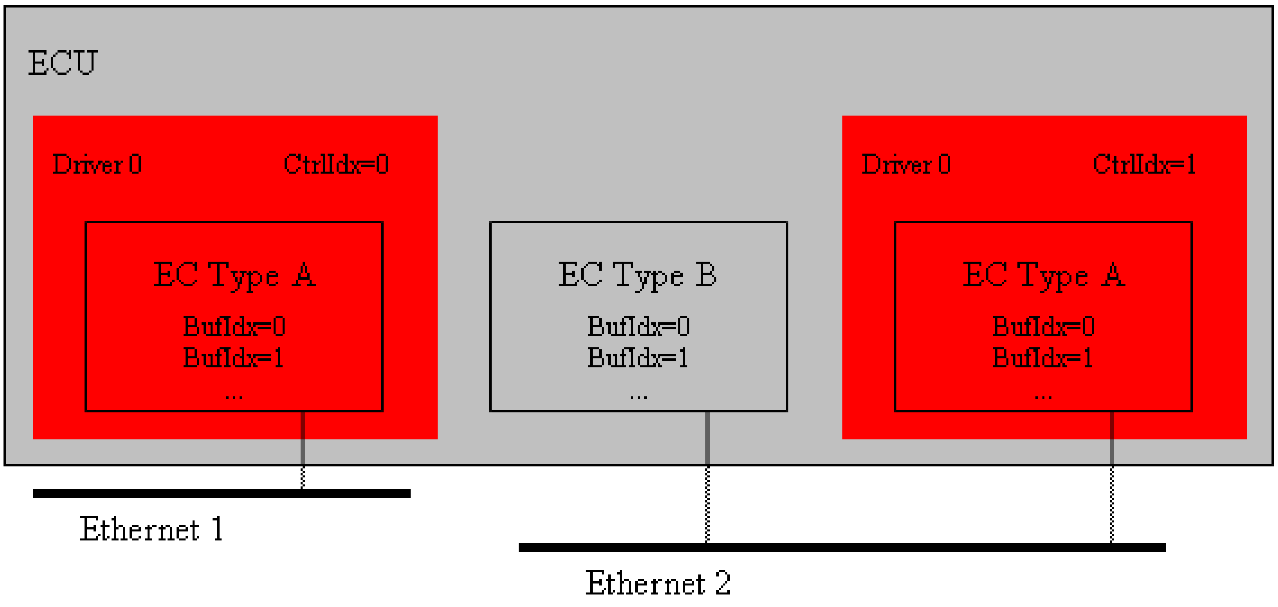


Figure 7.4: Ethernet Driver indexing scheme

[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.⌋*()*

[SWS_Eth_00004] ⌈A buffer index (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_00006] ⌈The header file Eth.h shall include a software and specification version number.⌋*()*

[SWS_Eth_00007] ⌈The Ethernet Driver module shall perform a consistency check between code files and header files based on pre-process-checking the version numbers of related code files and header files.⌋*()*

[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 to to inform the Switch Driver about a required special treatment for Switch management purpose (see document *AUTOSAR_SWS_EthernetInterface* [4]).⌋*()*

7.1.5 Communication

7.1.5.1 Transmission

The Ethernet driver provides two approaches to handle transmission requests.

7.1.5.1.1 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

Specification for transmission can be found in subsection 8.3.24 and subsection 8.3.23

7.1.5.1.2 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]{DRAFT} ⌈If Eth_ImmediateTransmission has been called and the given CtlrIdx has an EthCtrlConfigEgressQueue configured, then the Ethernet driver shall perform the following precondition checks in the following order:

1. If the configured EthCtrlConfigEgressQueuePriorityAssignment matches to the given priority, then proceed. Otherwise return with E_NOT_OK or, if EthDevErrorDetect is set to TRUE, call Det_ReportError with the error code ETH_E_UNKNOWN_PRIORITY_TX_FAILED
2. If an element of 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.⌋*(SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_00314]{DRAFT} ⌈If Eth_ImmediateTransmission 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_E_EXCEED_EGRESS_QUEUE_ELEMENT. Otherwise proceed with construction of the Ethernet frame.⌋*(SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_00315]{DRAFT} ⌈If Eth_ImmediateTransmission 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

⌋*(SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_00316]{DRAFT} ⌈If Eth_ImmediateTransmission 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 EthCtrEgressHardwareSupportedDataTransferThreshold, 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 EthCtrEgressHardwareSupportedDataTransferThreshold, 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 EthCtrEgressHardwareSupportedDataTransferThreshold, then the Ethernet driver shall prepare a hardware sup-

ported transfer with respect to the given payload length (PayloadLength) and payload location (PayloadPtr), trigger the data transfer and reserve space according the given PayloadLengthin 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 EthCtrEgressHardwareSupportedDataTransferThreshold, 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

⌋*(SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

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]{DRAFT} ⌈If Eth_ImmediateTransmission 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

⌋*(SRS_Eth_00172, SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

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.

7.1.5.2 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.

7.1.5.2.1 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]{DRAFT} ⌈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.⌋*(SRS_Eth_00172, SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

7.1.5.2.2 Direct data provision

[SWS_Eth_00321]{DRAFT} ⌈If `Eth_ImmediateTransmission` 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_ImmediateTransmission` which refer to the same `EthCtrlConfigEgressQueue` element.⌋*(SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

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

[SWS_Eth_00322]{DRAFT} ⌈If `Eth_ImmediateTransmission` 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`

⌋*(SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

7.1.5.3 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 subsection 7.1.6 for more details). Otherwise the `EthCtrlConfigIngressQueue`s 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 (please refer to [SWS_Eth_00096]).⌋*()*

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

[SWS_Eth_00323]{DRAFT} ⌈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`.⌋*(SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_00324]{DRAFT} ⌈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`.⌋*(SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_00327]{DRAFT} ⌈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`.⌋*(SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_00328]{DRAFT} ⌈When calling the callback function `EthIf_RxIndication`, then the Ethernet driver shall keep the affected `EthCtrlConfigIn-`

gressQueue element locked, until Eth_ReleaseRxBuffer is called with RxHandleId associated with the affected EthCtrlConfigIngressQueue element.⌋ *(SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_00329]{DRAFT} ⌈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.⌋*(SRS_Eth_00173, SRS_BSW_00350, SRS_BSW_00386)*

7.1.5.4 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 EthCtrEgressHardwareSupportedDataTransferThreshold.

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]{DRAFT} ⌈If a specific Ethernet controller has EthCtrlEnableEgressHardwareSupportedDataTransfer set to TRUE and the length of data to be transferred exceeds the configured EthCtrEgressHardwareSupportedDataTransferThreshold, 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).⌋*(SRS_Eth_00172, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_00320]{DRAFT} ⌈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⌋*(SRS_Eth_00172, SRS_BSW_00350, SRS_BSW_00386)*

7.1.6 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 EthCtrlConfigEgressQueueBufLenByte 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 subchapters describe the specific properties of EthCtrlConfigIngressQueues and EthCtrlConfigEgressQueues.

7.1.6.1 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]{DRAFT} ⌈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.⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_00332]{DRAFT} ⌈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.⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

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]{DRAFT} ⌈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.⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_CONSTR_00006]{DRAFT} ⌈An Ethernet Controller shall have at most one EthCtrlConfigIngressQueue with the same value of EthCtrlConfigIngressQueueSortingType configured⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

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]{DRAFT} ⌈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.⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_CONSTR_00007]{DRAFT} ⌈An EthCtrlConfigIngressQueue with no EthCtrlConfigIngressQueueSortingType configured, shall always have the lowest EthCtrlConfigIngressQueueSortingPriority.⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_CONSTR_00008]{DRAFT} ⌈An EthCtrlConfig shall have at most one EthCtrlConfigIngressQueue with no EthCtrlConfigIngressQueueSortingTypes configured.⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

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]{DRAFT} ⌈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.⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_00335]{DRAFT} ⌈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.⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

7.1.6.1.1 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 enqueuing 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: enqueuing 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: enqueuing 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 config-

ure 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 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]{DRAFT} ⌈An EthCtrlConfigIngressQueue with EthCtrlConfigIngressQueueHandlerFunction configured, shall be processed in the context of the generated ingress queue handler function.⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

[SWS_Eth_00336]{DRAFT} ⌈An EthCtrlConfigIngressQueue with EthCtrlEnableIngressQueueInterrupt set to TRUE, shall be processed in the context of the signaled interrupt service routine.⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

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

[SWS_Eth_00096]{OBSOLETE} ⌈The function 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.⌋*()*

[SWS_Eth_00337]{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.⌋*()*

7.1.6.2 Egress queue

An EthCtrlConfig could have 1 to 8 EthCtrlConfigEgressQueues configured. For each EthCtrlConfigEngressQueue one or multiple EthCtrlConfigEgressQueuePriorityAssignment could be assigned. Ethernet Frames which are requested to be transmitted with the given VLAN priority that match to the EthCtrlConfigEgressQueuePriorityAssignment of a EthCtrlConfigEgressQueue are added to this EthCtrlConfigEgressQueue. Ethernet frames with VLAN priority that match to the configured EthCtrlConfigEgressQueuePriorityAssignment of a EthCtrlConfigEgressQueue are sorted in that EthCtrlConfigIngressQueue.

[SWS_Eth_CONSTR_00009]{DRAFT} ⌈An Ethernet Controller shall have at most one EthCtrlConfigEgressQueue with the same value of VLAN priority (EthCtrlConfigEgressQueuePriorityAssignment) configured⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

Note: Multiple EthCtrlConfigEgressQueues with the same value of VLAN priority (e.g. 2 egress queues with same VLAN priority set to 0x04 configured) are invalid.

[SWS_Eth_CONSTR_00004]{DRAFT} ⌈An EthCtrlConfig shall have at most one EthCtrlConfigEgressQueue with no EthCtrlConfigEgressQueuePriorityAssignments configured.⌋*(SRS_Eth_00171, SRS_BSW_00350, SRS_BSW_00386)*

Note: A EthCtrlConfigEgressQueue with no VLAN priority configured, represents a queue where all Ethernet frames added, independent of the VLAN priority.

7.1.7 Buffer handling

It is possible to use an optional software buffer handling mechanism. Buffer handling by software is needed in case no hardware feature is available that ensures a fair traffic scheduling. Fair traffic scheduling is needed to avoid uncontrolled postponement of messages due to (too) strict priority handling.

The optional SW buffer handling is based on the so-called Credit Based Shaper algorithm (CBS). A CBS algorithm distributes Ethernet frames into dedicated SW queues based on their priority.

The CBS algorithm uses credits given in Bytes in order to ensure a fair distribution of transmission chances among the different SW queues.

The SW buffer (SW Buffer Pools) and physical memory on PHY level (HW queue) used normally are expanded with the CBS on basis of so-called SW queue. A transmission procedure consider at least the following points:

- Call of *Eth_ProvideTxBuffer()* will reserve a SW buffer pool of the SW buffer, store the given priority, return a pointer to the particular SW buffer pool and the unique buffer index of this SW buffer pool.
- The upper layer will copy the transmission data to the given SW buffer pool
- After data to transmit has been copied to the given SW buffer pool, the upper layer will call *Eth_Transmit()* with the according buffer index. The Ethernet driver

will add the given buffer index to the SW queue according to the provided priority, which was previously given within the call of *Eth_ProvideTxBuffer()*

- SW queue are handled according to the CBS algorithm. If an element of the SW queue is rated to be transmitted by the CBS, the SW buffer pool which corresponds to the buffer index (given by the element of the SW queue) is copied to the HW queue. The SW buffer pool is released and available for further transmission requests.

The CBS, its elements and the different API calls involved are depicted in the following graphic:

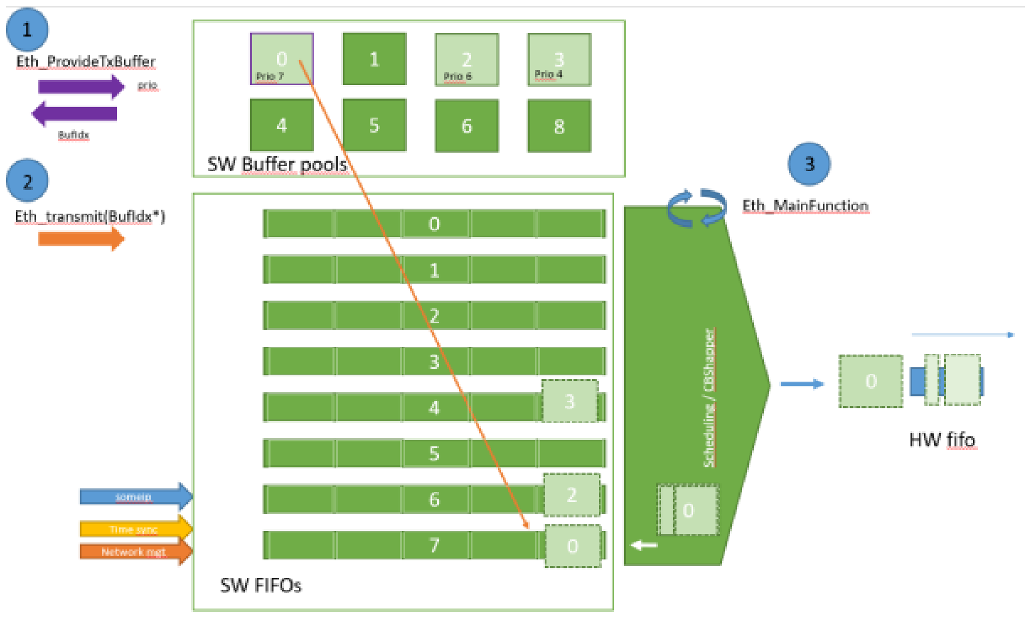


Figure 7.5: CBS algorithm

[SWS_Eth_00263] ⌈If the configuration parameter EthCtrlConfigSwBufferHandling is set to TRUE, then the optional SW buffer handling shall be enabled.⌋*(SRS_Eth_00146)*

Note: If buffer handling is supported by hardware, it is recommended to deactivate the software buffer handling by setting EthCtrlConfigSwBufferHandling to FALSE.

[SWS_Eth_00299]{OBSOLETE} ⌈If the configuration parameter EthCtrlConfigSw BufferHandling is set to TRUE, then one SW FIFO shall be available per configured EthCtrlConfigEgressFifo.⌋*(SRS_Eth_00146)*

[SWS_Eth_CONSTR_00002]{DRAFT} ⌈If the configuration parameter EthCtrlConfigSwBufferHandling is set to TRUE, then one SW queue per configured EthCtrlConfigEgressQueue shall be available.⌋*(SRS_Eth_00146)*

[SWS_Eth_CONSTR_00003]{DRAFT} ⌈If the configuration parameter EthCtrlConfigSwBufferHandling is set to TRUE, then at least two egress queues (via EthCtrlConfigEgressQueue) shall be configured.⌋*(SRS_Eth_00146)*

Note: Each SW queue configuration is derived from exactly one given EthCtrlConfigEgressFifo.

[SWS_Eth_00298]{OBSOLETE} ⌈If the configuration parameter EthCtrlConfigSw BufferHandling is set to TRUE, then each SW FIFO shall handle frames according to the configured priorities given by EthCtrlConfigEgressFifoPriorityAssignment aggregated by the according EthCtrlConfigEgressFifo. If no EthCtrlConfigEgressFifoPriorityAssignment is configured, then any priority shall be handled by this SW FIFO.⌋ *(SRS_Eth_00146)*

[SWS_Eth_00302]{DRAFT} ⌈If the configuration parameter EthCtrlConfigSwBuffer Handling is set to TRUE, then each SW queue shall handle frames according to the configured priorities given by EthCtrlConfigEgressQueuePriorityAssignment aggregated by the according EthCtrlConfigEgressQueue. If no EthCtrlConfigEgressQueue PriorityAssignment is configured, then any priority shall be handled by this SW queue.⌋ *(SRS_Eth_00146)*

Note: It is recommended to assign exactly one priority per EthCtrlConfigEgressQueue to support the performance of a software shaping algorithm.

[SWS_Eth_00264]{OBSOLETE} ⌈If the config parameter EthCtrlConfigSwBufferHandling is set to TRUE, then each SW FIFO shall have the total amount of elements given by EthCtrlConfigEgressFifoBufTotal ([ECUC_Eth_00050]). Each element shall be of type Eth_BufIdxType.⌋ *(SRS_Eth_00146)*

[SWS_Eth_00303]{DRAFT} ⌈If the configuration parameter EthCtrlConfigSwBuffer Handling is set to TRUE, then each SW queue shall have the total amount of elements given by EthCtrlConfigEgressQueueBufTotal (see t.b.d.). Each element shall be of type Eth_BufIdxType.⌋ *(SRS_Eth_00146)*

Note: SW queues have to store the buffer index which was reserved in a previous call of *Eth_ProvideTxBuffer()*.

[SWS_Eth_00297]{OBSOLETE} ⌈If the config parameter EthCtrlConfigSwBufferHandling is set to TRUE, then a SW buffer shall be provided with a size according to all configured EthCtrlConfigEgressFifo's. The size of each EthCtrlConfigEgressFifo shall be calculated in bytes by considering the following formula: size of one EthCtrlConfigEgressFifo = EthCtrlConfigEgressFifoBufTotal * EthCtrlConfigEgressFifoBufLenByte.⌋ *(SRS_Eth_00146)*

[SWS_Eth_00304]{DRAFT} ⌈If the config parameter EthCtrlConfigSwBufferHandling is set to TRUE, then a SW buffer shall be provided with a size according to all configured EthCtrlConfigEgressQueue's. The size of each EthCtrlConfigEgressQueue shall be calculated in bytes by considering the following formula: size of one EthCtrlConfigEgressQueue = EthCtrlConfigEgressQueueBufTotal * EthCtrlConfigEgressQueueBufLenByte.⌋ *(SRS_Eth_00146)*

Note: Along with the SW buffer, the Ethernet driver has to handle the mapping between the given priority (provided by *Eth_ProvideTxBuffer*) and the according buffer index of the reserved SW puffer pool.

[SWS_Eth_00265]{OBSOLETE} ⌈All SW FIFOs shall follow the criteria listed here:

- Each SW FIFO shall be filled and read out according to FIFO principles.
- The SW FIFOs shall support independent configuration regardless of any settings on the rest of SW FIFOs.

⌋*(SRS_Eth_00146)*

[SWS_Eth_00305]{DRAFT} ⌈All SW queues shall follow the criteria listed here:

- Each SW queue shall be filled and read out according to FIFO principles.
- The SW queue shall support independent configuration regardless of any settings on the rest of SW queue.

⌋*(SRS_Eth_00146)*

Note: Regarding last bulletin point, it is recommended to use different settings of EthCtrlConfigEgressQueueCreditBasedShaperIdleSlope and EthCtrlConfigEgressQueueCreditBasedShaperSendSlope per SW queue. Those two configuration parameters can be used to freely configure the output rate of the SW queue as demanded.

[SWS_Eth_00266]{OBSOLETE} ⌈SW FIFOs shall be iterated and their credits account be updated in the following way and order:

- Credits are only accumulated for SW FIFOs which have at least one message queued inside them. Empty SW FIFOs do not accumulate credits and their credits counter shall be set to 0.
- Iterate through all SW FIFOs, starting at the highest priority SW FIFO and descending, and add the amount of credits accumulated since the last *Eth_MainFunction()* call. The amount of credits accumulated is given by EthCtrlConfigShaperIdleSlope.
- If a SW FIFO reaches EthCtrlConfigShaperMaxCredit then the credit accumulation shall stop at that point and the next SW FIFO in the row is handled.

⌋*(SRS_Eth_00146)*

[SWS_Eth_00306]{DRAFT} ⌈SW queue shall be iterated and their credits account be updated in the following way and order:

- Credits are only accumulated for SW queues which have at least one message queued inside them. Empty SW queues do not accumulate credits and their credits counter shall be set to 0.
- Iterate through all SW queues, starting at the highest priority SW queue and descending, and add the amount of credits accumulated since the last *Eth_MainFunction()* call. The amount of credits accumulated is given by EthCtrlConfigEgressQueueCreditBasedShaperIdleSlope.

- If a SW queue reaches EthCtrlConfigQueueCreditBasedShaperMaxCredit then the credit accumulation shall stop at that point and the next SW queue in the row is handled.

⌋*(SRS_Eth_00146)*

[SWS_Eth_00267] ⌈If Eth_ProvideTxBuffer() is called and EthCtrlConfigSwBufferHandling is set to TRUE, a tuple of BuffIdx pointer to the SW buffer pool (which is returned) and priority (provided by argument of the current function call) shall be stored.⌋*(SRS_Eth_00146)*

[SWS_Eth_00268]{OBSOLETE} ⌈When *Eth_Transmit()* is called and EthCrtlConfigSwBufferHandling is set to TRUE, the given BuffIdx pointer shall be assigned to the SW FIFO with the EthCtrlConfigEgressFifoPriorityAssignment which matches the priority given previously by the previous Eth_ProvideTxBuffer() call (see [SWS_Eth_00267]).⌋*(SRS_Eth_00146)*

[SWS_Eth_00307]{DRAFT} ⌈When *Eth_Transmit()* is called and EthCtrlConfigSwBufferHandling is set to TRUE, the given BuffIdx pointer shall be assigned to the SW queue with the EthCtrlConfigEgressQueuePriorityAssignment which matches the priority given previously by the previous Eth_ProvideTxBuffer() call (see [SWS_Eth_00267]).⌋*(SRS_Eth_00146)*

[SWS_Eth_00269]{OBSOLETE} ⌈Upon calling *Eth_Transmit()*, messages from the SW FIFOs shall be moved to the HW FIFO as described in [SWS_Eth_00271].⌋*(SRS_Eth_00146)*

[SWS_Eth_00308]{DRAFT} ⌈Upon calling *Eth_Transmit()*, messages from the SW queue shall be moved to the HW queue as described in [SWS_Eth_00310].⌋*(SRS_Eth_00146)*

[SWS_Eth_00270]{OBSOLETE} ⌈In the context of *Eth_MainFunction()*, the following actions shall be executed in the given order:

- All SW FIFOs shall be iterated and their credits account updated as specified in [SWS_Eth_00266].
- All SW FIFOs shall be iterated and checked for messages which are ready for transmission.
- For each SW FIFO iterated, transmission shall be attempeted as specified in [SWS_Eth_00271].

⌋*(SRS_Eth_00146)*

[SWS_Eth_00309]{DRAFT} ⌈In the context of *Eth_MainFunction()*, the following actions shall be executed in the given order:

- All SW queue shall be iterated and their credits account updated as specified in [SWS_Eth_00306].

- All SW queue shall be iterated and checked for messages which are ready for transmission.
- For each SW queue iterated, transmission shall be attempeted as specified in [SWS_Eth_00310].

⌋*(SRS_Eth_00146)*

[SWS_Eth_00271]{OBSOLETE} ⌈Messages queued inside SW FIFOs shall be moved to the HW FIFO in the following way and order:

- Loop through each SW FIFO, starting at the highest priority in descending order.
- Move the first message inside a SW FIFO whose credit account is at least Eth CtrlConfigShaperMinCredit to the HW FIFO.
- If EthTrcvPhysLayerPLCAMaxBurstCount is set to 0 then only one message is moved to the HW FIFO and the iteration to the next SW FIFOs is stopped.
- Reduce the SW FIFOs credits based on its EthCtrlConfigShaperSendSlope configuration.
- If EthTrcvPhysLayerPLCAMaxBurstCount is higher than 0 then proceed on top as specified in [SWS_Eth_00272].

⌋*(SRS_Eth_00146)*

[SWS_Eth_00310]{DRAFT} ⌈Messages queued inside SW queue shall be moved to the HW queue in the following way and order:

- Loop through each SW queue, starting at the highest priority in descending order.
- Move the first message inside a SW queue whose credit account is at least Eth CtrlConfigEgressQueueCreditBasedShaperMinCredit to the HW queue.
- If EthTrcvPhysLayerPLCAMaxBurstCount is set to 0 then only one message is moved to the HW queue and the iteration to the next SW queue is stopped.
- Reduce the SW FIFOs credits based on its EthCtrlConfigEgressQueueCreditBasedShaperSendSlope configuration.
- If EthTrcvPhysLayerPLCAMaxBurstCount is higher than 0 then proceed on top as specified in [SWS_Eth_00311].

⌋*(SRS_Eth_00146)*

[SWS_Eth_00272]{OBSOLETE} ⌈If EthTrcvPhysLayerPLCAMaxBurstCount is higher than 0, as many messages as EthTrcvPhysLayerPLCAMaxBurstCount indicates shall be moved additionally to the HW FIFO. The selection of each message shall be based on the requirements in [SWS_Eth_00271].⌋*(SRS_Eth_00146)*

[SWS_Eth_00311]{DRAFT} ⌈If EthTrcvPhysLayerPLCAMaxBurstCount is higher than 0, as many messages as EthTrcvPhysLayerPLCAMaxBurstCount indicates shall be moved additionally to the HW queue. The selection of each message shall be based on the requirements in [SWS_Eth_00310].⌋*(SRS_Eth_00146)*

7.1.8 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 subsubsection 7.1.8.1).

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 subsubsection 7.1.8.2).

Finally, if a PHC is supported, the Ethernet Driver may also support the generation of a Pulse-Per-Second (PPS) signal (refer to chapter subsubsection 7.1.8.3)

[SWS_Eth_00176]{OBSOLETE} ⌈The Global Time interfaces shall be used to access the time synchronization functionalities (see document [13]).⌋*()*

7.1.8.1 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_GetIngressTimeStamp` to read the ingress timestamp of a received 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)

7.1.8.2 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]{DRAFT} ⌈If EthGlobalTimeSupport is set to FALSE, then EthPhcSupport shall be set to FALSE⌋*(SRS_Eth_00167)*

[SWS_Eth_00373]{DRAFT} ⌈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.⌋*(SRS_Eth_00167)*

[SWS_Eth_00374]{DRAFT} ⌈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.⌋*(SRS_Eth_00167)*

[SWS_Eth_00375]{DRAFT} ⌈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.⌋*(SRS_Eth_00167)*

[SWS_Eth_CONSTR_00011]{DRAFT} ⌈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.

⌋*(SRS_Eth_00167)*

7.1.8.2.1 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]{DRAFT} ⌈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

⌋*(SRS_Eth_00167)*

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

[SWS_Eth_00340]{DRAFT} ⌈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 `EthClkUnitTimePhcRef`
- the value of the timestamping HW clock which is referenced via `EthClkUnitTimeStampingRef`

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

⌋*(SRS_Eth_00167)*

[SWS_Eth_00341]{DRAFT} ⌈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⌋*(SRS_Eth_00167)*

7.1.8.3 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 [14, FO_EXP_TimeSensitiveNetworkFeatures] for more details. It is assumed that the PPS signal gerneration as configured by the EthCtrlPulsePerSecondConfig is derived automatically from the PHC and driven in hardware.

[SWS_Eth_CONSTR_00012]{DRAFT} ⌈A EthCtrlPulsePerSecondConfig configuration shall be rejected as invalid, if the affected Ethernet controller hardware do not support PPS signal generation.⌋*(SRS_Eth_00168)*

[SWS_Eth_00342]{DRAFT} ⌈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

⌋*(SRS_Eth_00168)*

[SWS_Eth_00377]{DRAFT} ⌈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.⌋*(SRS_Eth_00168)*

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]{DRAFT} ⌈If EthCtrlPulsePerSecondConfig has EthCtrlPulsePerSecondStartEnum set to RISING_EDGE and the Ethernet contoller 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.⌋*(SRS_Eth_00168)*

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

[SWS_Eth_00343]{DRAFT} ⌈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.⌋*(SRS_Eth_00168)*

[SWS_Eth_00378]{DRAFT} ⌈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.⌋*(SRS_Eth_00168)*

[SWS_Eth_00379]{DRAFT} ⌈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.⌋*(SRS_Eth_00168)*

[SWS_Eth_00344]{DRAFT} ⌈If Eth_SetPpsSignalMode is called and EthCtrlPulsePerSecondConfig is NOT configured for the given CtrlIdx, then the Ethernet Driver shall return with E_NOT_OK.⌋*(SRS_Eth_00168)*

7.1.9 Configuration description

[SWS_Eth_00012] ⌈The Ethernet Driver module shall provide an XML file that contains the data, which is required for the SW identification (it shall contain the vendor identification, module ID and software version information), configuration and integration process. This file should describe vendor specific configuration parameters as well as it should contain recommended configuration parameter values.⌋*()*

[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.⌋*()*

[SWS_Eth_00013] ⌈The configuration of the Ethernet Driver module shall be calculated at ECU configuration time. None of the communication parameters shall be calculated at runtime.⌋*()*

[SWS_Eth_00014] ⌈The start address of post-build time configuration data shall be passed during module initialization.⌋*()*

Note: For more details regarding the intialization please refer to section 8.3.1.

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

Section 7.2 "Error Handling" of the document *"General Specification of Basic Software Modules"* [3], 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] Definiton 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
No egress queue for requested priority available	ETH_E_UNKNOWN_EGRESS_PRIORITY	0x08
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
A received Ethernet frame could not be enqueued in any ingress queue	ETH_E_NO_MATCHING_INGRESS_QUEUE_IDENTIFIED	0x0C

⌋*()*

7.2.2 Runtime Errors

[SWS_Eth_91014] Definiton 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

⌋*()*

7.2.3 Transient Faults

There are no transient faults.

7.2.4 Production Errors

There are no production errors.

7.2.5 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] ⌈

Error Name:	ETH_E_ACCESS	
Short Description:	Ethernet Controller Access Failure.	
Long Description:	Monitors the access to the Ethernet Controller.	
Detection Criteria:	Fail	When access to the Ethernet Controller fails the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When access to the Ethernet Controller succeds 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_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] ⌈

Error Name:	ETH_E_CRC	
Short Description:	CRC Failure	
Long Description:	Monitors invalid Ethernet frames during reception.	
Detection Criteria:	Fail	When invalid 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_00220] ⌈

Error Name:	ETH_E_UNDERSIZEFRAME	
Short Description:	Frame Size Underflow	
Long Description:	Monitors undersize Ethernet frames during reception.	
Detection Criteria:	Fail	When invalid 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_00221] ⌈

Error Name:	ETH_E_OVERSIZEFRAME	
Short Description:	Frame Size Overflow	
Long Description:	Monitors oversize Ethernet frames during reception.	
Detection Criteria:	Fail	When invalid 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_00222] ⌈

Error Name:	ETH_E_ALIGNMENT	
Short Description:	Frame Alignment Error	
Long Description:	Monitors alignment errors.	
Detection Criteria:	Fail	When invalid 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_00223] ⌈

Error Name:	ETH_E_SINGLECOLLISION	
Short Description:	Single Frame Collision	
Long Description:	Monitors Ethernet single frame collision.	
Detection Criteria:	Fail	When frame collisions 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_00224] ⌈

Error Name:	ETH_E_MULTIPLECOLLISION	
Short Description:	Multiple Frame Collision	
Long Description:	Monitors Ethernet multiple frame collision.	
Detection Criteria:	Fail	When fram collisions 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_00225] ⌈

Error Name:	ETH_E_LATECOLLISION	
Short Description:	Late Frame Collision	
Long Description:	Monitors Ethernet late frame collision.	
Detection Criteria:	Fail	When frame collisions 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.	

⌋*()*

8 API specification

8.1 Imported types

This chapter lists all types included from the following modules:

[SWS_Eth_00026] Definition of imported datatypes of module Eth ⌈

Module	***Header File***	***Imported Type***
ComStack_Types	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	ListElemStructType (draft)
	ComStackTypes.h	TimeStampQualType (draft)
	ComStackTypes.h	TimeStampType (draft)
	ComStackTypes.h	TimeTupleType (draft)
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Icu	Icu.h	Icu_ChannelType
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

⌋*(SRS_Eth_00127)*

8.2 Type definitions

8.2.1 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

⌋*()*

8.2.2 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 communicaton (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 communcation 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		

⌋*()*

8.2.3 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		

⌋*()*

8.2.4 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

⌋*()*

8.2.5 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	

⌋*()*

8.2.6 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

⌋*()*

8.2.7 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		

⌋*()*

8.2.8 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 taklen for the MAC address given in *PhysAddrPtr.		
Available via	Eth_GeneralTypes.h		

⌋*()*

8.2.9 Eth_TimeStampQualType

[SWS_Eth_00177]{OBSOLETE} **Definition of datatype Eth_TimeStampQualType ⌈**

Name	Eth_TimeStampQualType (obsolete)		
Kind	Enumeration		
Range	ETH_VALID	0	–
	ETH_INVALID	1	–
	ETH_UNCERTAIN	2	–

▽

△

Description	Depending on the HW, quality information regarding the evaluated time stamp might be supported. If not supported, the value shall be always Valid. For Uncertain and Invalid values, the upper layer shall discard the time stamp. **Tags:** atp.Status=obsolete
Available via	Eth_GeneralTypes.h

⌋*()*

8.2.10 Eth_TimeStampType

[SWS_Eth_00178]{OBSOLETE} **Definition of datatype Eth_TimeStampType** ⌈

Name	Eth_TimeStampType (obsolete)	
Kind	Structure	
Elements	nanoseconds	
	Type	uint32
	Comment	Nanoseconds part of the time
	seconds	
	Type	uint32
	Comment	32 bit LSB of the 48 bits Seconds part of the time
	secondsHi	
	Type	uint16
	Comment	16 bit MSB of the 48 bits Seconds part of the time
Description	Variables of this type are used for expressing time stamps including relative time and absolute calendar time. The absolute time starts at 1970-01-01. 0 to 281474976710655s == 3257812230d [0xFFFF FFFF FFFF] 0 to 999999999ns [0x3B9A C9FF] invalid value in nanoseconds: [0x3B9A CA00] to [0x3FFF FFFF] Bit 30 and 31 reserved, default: 0 **Tags:** atp.Status=obsolete	
Available via	Eth_GeneralTypes.h	

⌋*()*

8.2.11 Eth_TimeIntDiffType

[SWS_Eth_00179]{OBSOLETE} **Definition of datatype Eth_TimeIntDiffType** ⌈

Name	Eth_TimeIntDiffType (obsolete)	
Kind	Structure	
Elements	diff	
	Type	Eth_TimeStampType
	Comment	time difference
	sign	
	Type	boolean

▽

△

	Comment	Positive (True) / negative (False) time
Description	Variables of this type are used to express time differences. **Tags:** atp.Status=obsolete	
Available via	Eth_GeneralTypes.h	

⌋()

8.2.12 Eth_RateRatioType

[SWS_Eth_00180]{OBSOLETE} Definition of datatype Eth_RateRatioType ⌈

Name	Eth_RateRatioType (obsolete)	
Kind	Structure	
Elements	IngressTimeStampDelta	
	Type	Eth_TimeIntDiffType
	Comment	IngressTimeStampSync2 - IngressTimeStampSync1
	OriginTimeStampDelta	
	Type	Eth_TimeIntDiffType
	Comment	OriginTimeStampSync2[FUP2] - OriginTimeStampSync1[FUP1]
Description	Variables of this type are used to express frequency ratios. **Tags:** atp.Status=obsolete	
Available via	Eth_GeneralTypes.h	

⌋()

8.2.13 Eth_MacVlanType

[SWS_Eth_91001] Definition of datatype Eth_MacVlanType ⌈

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

⌋*(SRS_Eth_00121, SRS_Eth_00072)*

8.2.14 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	

⌋*()*

8.2.15 Eth_RxStatsType

[SWS_Eth_91002] Definition of datatype Eth_RxStatsType ⌈

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 etherStats BroadcastPkts
	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 etherStats MulticastPkts.
	RxStatsCrcAlignErrors	
	Type	uint32
	Comment	The total number of packets received that had a length of bertween 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 ether StatsUndersizePkts.
	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 ether StatsOversizePkts
	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 etherStats Fragments.
	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 etherStats Pkts64Octets
	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	

⌋(*SRS_Eth_00127*)

8.2.16 Eth_TxStatsType

[SWS_Eth_91003] Definition of datatype Eth_TxStatsType ⌈

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 ifOut UcastPkts.
Description	Statistic counter for diagnostics.	
Available via	Eth_GeneralTypes.h	

⌋(*SRS_Eth_00127*)

8.2.17 Eth_TxErrorCounterValuesType

[SWS_Eth_91004] Definition of datatype Eth_TxErrorCounterValuesType ⌈

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 ifOut Discards
	TxDroppedErrorPkts	
	Type	uint32
	Comment	transmitted because of errors. Also described in IETF RFC1213 MIB if OutErrors
	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 dot3Stats DeferredTransmissions
	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 TxNUcast Pkts and is not counted by the corresponding instance of the Tx MultipleCollision 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 TxNUcast Pkts and is not counted by the corresponding instance of the TxSingle Collision object. Also described in IETF RFC1643 MIB dot3Stats MultipleCollisionFrames.
	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
	TxExcessiveCollison	
	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 dot3StatsExcessiveCollisions
Description	Statistic counters for diagnostics.	
Available via	Eth_GeneralTypes.h	

⌋*(SRS_Eth_00127)*

8.2.18 Eth_SpiStatusType

[SWS_Eth_91013]{DRAFT} **Definition of datatype Eth_SpiStatusType** ⌈

Name	Eth_SpiStatusType (draft)	
Kind	Structure	
Elements	SpiStatusRegister	
	Type	uint32

▽

△

	Comment	Bit mapped status defined by OA TC6 [26] 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 [26]. TRUE: MACPHY has been reset and is not configured. FALSE: MACPHY is configured.
	BufferStatusTxCredit	
	Type	uint8
	Comment	Contains the number of consecutive transmited 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.h	

⌋*(SRS_Eth_00147, SRS_Eth_00120)*

8.2.19 Eth_RateDeviationType

[SWS_Eth_91015]{DRAFT} **Definition of datatype Eth_RateDeviationType** ⌈

Name	Eth_RateDeviationType (draft)	
Kind	Structure	
Elements	rateDeviationValue	
	Type	sint32
	Comment	Rate deviation value (resolution: 2^{-41})

▽

△

	rateDeviationStatus	
	Type	Eth_RateDeviationStatusType
	Comment	Current state of the rate deviation calculation
Description	Rate deviation value and status **Tags:** atp.Status=draft	
Available via	Eth.h	

⌋*(RS_TS_20075)*

8.2.20 Eth_RateDeviationStatusType

[SWS_Eth_91016]{DRAFT} **Definition of datatype Eth_RateDeviationStatusType** ⌈

Name	Eth_RateDeviationStatusType (draft)		
Kind	Type		
Derived from	uint8		
Range	ETH_RATE_OK	0x00	A valid rate deviaton value is available/calculated
	ETH_RATE_NOT_ AVAILABLE	0xFE	No valid rate deviation value available/calculated
	ETH_RATE_EXCEEDED	0xFF	The calculated rate deviation value exceeds limits
Description	Type that indicates the current status of the rate calculation **Tags:** atp.Status=draft		
Available via	Eth.h		

⌋*(RS_TS_20075)*

8.3 Function definitions

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

8.3.1 Eth_Init

[SWS_Eth_00027] Definition of API function Eth_Init ⌈

Service Name	Eth_Init

Syntax	`void Eth_Init (`
	`  const Eth_ConfigType* CfgPtr`
	`)`

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_00300]{OBSOLETE} ⌈If the config parameter EthCtrlConfigSwBufferHandling is set to TRUE, then all SW FIFOs and SW buffer pools shall be initialized with '0'⌋*()*

[SWS_Eth_00312]{DRAFT} ⌈If the config parameter EthCtrlConfigSwBufferHandling is set to TRUE, then all SW queues and SW buffer pools shall be initialized with '0'.⌋*()*

Note: For more details see 7.1.7 Buffer handling.

[SWS_Eth_00350]{DRAFT} ⌈If the config parameter `EthPhcSupport` is set to `TRUE`, then the Ethernet driver shall check for all configured Ethernet controllers if `EthCtrlClk`s are configured. If `EthCtrlClk`s are configured, then the Ethernet driver shall initialize the Ethernet controller hardware clocks, set the intialization value to zero and start the hardware clock.⌋*(SRS_BSW_00406)*

[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.⌋*()*

8.3.2 Eth_SetControllerMode

[SWS_Eth_91009] Definition of API function Eth_SetControllerMode ⌈

Service Name	Eth_SetControllerMode	
Syntax	`Std_ReturnType Eth_SetControllerMode ( uint8 CtrlIdx, Eth_ModeType CtrlMode )`	
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`: success `E_NOT_OK`: controller mode could not be changed
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_00043] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00044] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[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()*.⌋*()*

8.3.3 Eth_GetControllerMode

[SWS_Eth_91010] Definition of API function Eth_GetControllerMode ⌈

Service Name	Eth_GetControllerMode	

Syntax	`Std_ReturnType Eth_GetControllerMode (`
	`  uint8 CtrlIdx,`
	`  Eth_ModeType* CtrlModePtr`
	`)`	

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_00048] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00049] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00050] ⌈If development error detection is enabled: the function shall check the parameter CtrlModePtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

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

8.3.4 Eth_GetPhysAddr

[SWS_Eth_00052] Definition of API function Eth_GetPhysAddr ⌈

Service Name	Eth_GetPhysAddr	

Syntax	`void Eth_GetPhysAddr (`
	`  uint8 CtrlIdx,`
	`  uint8* PhysAddrPtr`
	`)`	

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_00054] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00055] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00056] ⌈If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

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

8.3.5 Eth_SetPhysAddr

[SWS_Eth_00151] Definition of API function Eth_SetPhysAddr ⌈

Service Name	Eth_SetPhysAddr	
Syntax	`void Eth_SetPhysAddr ( uint8 CtrlIdx, const uint8* PhysAddrPtr )`	
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_00140] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00141] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00142] ⌈If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

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

8.3.6 Eth_UpdatePhysAddrFilter

[SWS_Eth_00152] Definition of API function Eth_UpdatePhysAddrFilter ⌈

Service Name	Eth_UpdatePhysAddrFilter	

Syntax	`Std_ReturnType Eth_UpdatePhysAddrFilter (`
	`  uint8 CtrlIdx,`
	`  const uint8* PhysAddrPtr,`
	`  Eth_FilterActionType Action`
	`)`	

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.
	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 receive filter of the indexed controller.⌋*()*

[SWS_Eth_00245] ⌈The Ethernet driver module will receive a frame when the destination Address match the PhyAddrPtr 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_00164] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00165] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00166] ⌈If development error detection is enabled the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

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

[SWS_Eth_00144] ⌈If the physical source address (MAC address) is set to FF:FF:FF:FF:FF:FF, this shall completely open the filter.⌋*()*

[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 source address (MAC address) is set to 00:00:00:00:00:00, this shall reduce the filter to the controllers unique unicast MAC address and end promiscuous mode if it was turned on.⌋*()*

8.3.7 Eth_WriteMii

[SWS_Eth_00058] Definition of API function Eth_WriteMii ⌈

Service Name	Eth_WriteMii	
Syntax	`Std_ReturnType Eth_WriteMii ( uint8 CtrlIdx, uint8 TrcvIdx, uint8 RegIdx, uint16 RegVal )`	
Service ID [hex]	0x05	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	TrcvIdx	Index of the transceiver on the MII (see [21] for details)
	RegIdx	Index of the transceiver register on the MII (see [21] for details)
	RegVal	Value to be written into the indexed register (see [21] for details)
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Service accepted `E_NOT_OK`: Service denied
Description	Configures a transceiver register or triggers a function offered by the receiver	
Available via	Eth.h	

⌋*()*

[SWS_Eth_00286]{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]{DRAFT} ⌈The function shall write the specified transceiver register through the MII according to Clause 22 [15] for the indexed controller.⌋*(SRS_Eth_00148)*

[SWS_Eth_00273] ⌈If Clause 45 registers need to be writen via this access mechanism, the API shall use the register 13 and 14 to access them as explicitly specified by the annex 22D [15].⌋*(SRS_Eth_00148)*

[SWS_Eth_00287]{DRAFT} ⌈If EthCtrlEnableSpiInterface is TRUE, the function shall process the write request as described in the TC6 [11].⌋*(SRS_Eth_00147, SRS_Eth_00146)*

[SWS_Eth_00288]{DRAFT} ⌈The function shall call EthTrcv_WriteMiiIndication when the PHY register access finished.⌋*(SRS_Eth_00148, SRS_Eth_00147, SRS_Eth_00146)*

[SWS_Eth_00060] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00061] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[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()*.⌋*()*

8.3.8 Eth_ReadMii

[SWS_Eth_00064] Definition of API function Eth_ReadMii ⌈

Service Name	Eth_ReadMii	
Syntax	`Std_ReturnType Eth_ReadMii ( uint8 CtrlIdx, uint8 TrcvIdx, uint8 RegIdx, uint16* RegValPtr )`	
Service ID [hex]	0x06	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	TrcvIdx	Index of the transceiver on the MII (see [21] for details)
	RegIdx	Index of the transceiver register on the MII (see [21] for details)
Parameters (inout)	None	
Parameters (out)	RegValPtr	Filled with the register content of the indexed register (see [21] for details)
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]{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.⌋*(SRS_Eth_00148, SRS_-Eth_00146)*

[SWS_Eth_00279]{DRAFT} ⌈The function shall read the specified transceiver register through the MII according to Clause 22 [15] for the indexed controller.⌋*(SRS_Eth_-00148, SRS_Eth_00146)*

[SWS_Eth_00274] ⌈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 [15].⌋*(SRS_Eth_00148)*

[SWS_Eth_00290]{DRAFT} ⌈If EthCtrEnableSpiInterface is TRUE, the function shall process the read request as described in the TC6 [11].⌋*(SRS_Eth_00148, SRS_Eth_-00146, SRS_Eth_00147)*

[SWS_Eth_00291]{DRAFT} ⌈The function shall call EthTrcv_ReadMiiIndication when the PHY register access finished.⌋*(SRS_Eth_00148, SRS_Eth_00146, SRS_Eth_-00147)*

[SWS_Eth_00066] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00067] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00068] ⌈If development error detection is enabled: the function shall check the parameter RegValPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

[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()*.⌋*()*

8.3.9 Eth_GetCounterValues

[SWS_Eth_00226] Definition of API function Eth_GetCounterValues ⌈

Service Name	Eth_GetCounterValues	

Syntax	`Std_ReturnType Eth_GetCounterValues (`
	`  uint8 CtrlIdx,`
	`  Eth_CounterType* CounterPtr`
	`)`	

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	

⌋*(SRS_Eth_00127)*

[SWS_Eth_00227] ⌈The function shall read a list of values from the indexed controller.⌋ *()*

[SWS_Eth_00228] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00229] ⌈If dev development elopment error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00230] ⌈If development error detection is enabled: the function shall check the parameter CounterPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

[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()*.⌋*()*

8.3.10 Eth_GetRxStats

[SWS_Eth_00233] Definition of API function Eth_GetRxStats ⌈

Service Name	Eth_GetRxStats	

Syntax	`Std_ReturnType Eth_GetRxStats (`
	`  uint8 CtrlIdx,`
	`  Eth_RxStatsType* RxStats`
	`)`	

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. ether StatsOctets 3. etherStatsPkts 4. etherStatsBroadcastPkts 5. etherStatsMulticastPkts 6. ether StatsCrcAlignErrors 7. etherStatsUndersizePkts 8. etherStatsOversizePkts 9. etherStats Fragments 10. etherStatsJabbers 11. etherStatsCollisions 12. etherStatsPkts64Octets 13. etherStatsPkts65to127Octets 14. etherStatsPkts128to255Octets 15. etherStats Pkts256to511Octets 16. etherStatsPkts512to1023Octets 17. etherStatsPkts1024to1518Octets	
Available via	Eth.h	

⌋*(SRS_Eth_00127)*

[SWS_Eth_00234] ⌈The function shall read a list of values from the indexed controller according to [16].⌋*()*

[SWS_Eth_00235] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00236] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00237] ⌈If development error detection is enabled: the function shall check the parameter RxStats for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

[SWS_Eth_00238] ⌈The function Eth_GetRxStats shall be pre compile time configurable On/Off by the configuration parameter: EthGetRxStatsApi.⌋*()*

8.3.11 Eth_GetTxStats

[SWS_Eth_91005] Definition of API function Eth_GetTxStats ⌈

Service Name	Eth_GetTxStats	

Syntax	`Std_ReturnType Eth_GetTxStats (`
	`  uint8 CtrlIdx,`
	`  Eth_TxStatsType* TxStats`
	`)`	

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

⌋*(SRS_Eth_00127)*

[SWS_Eth_00248] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*(SRS_BSW_00101, SRS_BSW_00416)*

[SWS_Eth_00249] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*(SRS_BSW_00323, SRS_BSW_00369)*

[SWS_Eth_00250] ⌈If development error detection is enabled: the function shall check the parameter TxStats for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*(SRS_BSW_00323, SRS_BSW_00369)*

[SWS_Eth_00251] ⌈The function Eth_GetTxStats shall be pre compile time configurable On/Off by the configuration parameter: EthGetTxStatsApi [ECUC_Eth_00060].⌋*(SRS_Eth_00053)*

8.3.12 Eth_GetTxErrorCounterValues

[SWS_Eth_91006] Definition of API function Eth_GetTxErrorCounterValues ⌈

Service Name	Eth_GetTxErrorCounterValues	

Syntax	`Std_ReturnType Eth_GetTxErrorCounterValues (`
	`  uint8 CtrlIdx,`
	`  Eth_TxErrorCounterValuesType* TxErrorCounterValues`
	`)`	

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)	TxErrorCounterValues	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_TxErrorCounterValuesType, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.	
Available via	Eth.h	

⌋*(SRS_Eth_00127)*

[SWS_Eth_00252] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*(SRS_BSW_00101, SRS_BSW_00416)*

[SWS_Eth_00253] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*(SRS_BSW_00323, SRS_BSW_00369)*

[SWS_Eth_00254] ⌈If development error detection is enabled: the function shall check the parameter TxStats for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*(SRS_BSW_00323, SRS_BSW_00369)*

[SWS_Eth_00255] ⌈The function Eth_GetTxErrorCounterValues shall be pre compile time configurable On/Off by the configuration parameter: EthGetTxErrorCounterValues Api [ECUC_Eth_00061].⌋*(SRS_Eth_00053)*

8.3.13 Eth_GetSpiStatus

[SWS_Eth_91012]{DRAFT} **Definition of API function Eth_GetSpiStatus** ⌈

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 [26] to identify if an error can occured at the SPI interface. **Tags:** atp.Status=draft	
Available via	Eth.h	

⌋*(SRS_Eth_00147, SRS_Eth_00120)*

[SWS_Eth_00292]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*(SRS_BSW_00101, SRS_BSW_00416)*

[SWS_Eth_00293]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*(SRS_BSW_00323, SRS_BSW_00369)*

[SWS_Eth_00294]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter SpiStatusType for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*(SRS_BSW_00323, SRS_BSW_00369)*

[SWS_Eth_00295]{DRAFT} ⌈The function Eth_GetSpiStatus shall be pre compile time configurable On/Off by the configuration parameter: EthCtrlEnableSpiInterface[ECUC_Eth_00073].⌋*(SRS_Eth_00146, SRS_Eth_00147)*

8.3.14 Eth_GetCurrentTime

(OBSOLETE, replaced by Eth_GetCurrentTimeTuple (SWS_Eth_91017))

[SWS_Eth_00181]{OBSOLETE} **Definition of API function Eth_GetCurrentTime** ⌈

Service Name	Eth_GetCurrentTime (obsolete)	

Syntax	`Std_ReturnType Eth_GetCurrentTime (`
	`  uint8 CtrlIdx,`
	`  TimeStampQualType* timeQualPtr,`
	`  TimeStampType* timeStampPtr`
	`)`	

Service ID [hex]	0x16	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the addresses controller.
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`: successful `E_NOT_OK`: failed
Description	Returns a time value out of the HW registers according to the capability of the HW. Is the HW resolution is lower than the Eth_TimeStampType resolution resp. range, than an the remaining bits will be filled with 0. Important Note: Eth_GetCurrentTime may be called within an exclusive area. **Tags:** atp.Status=obsolete	
Available via	Eth.h	

⌋*()*

[SWS_Eth_00182]{OBSOLETE} ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00183]{OBSOLETE} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00184]{OBSOLETE} ⌈If development error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

[SWS_Eth_00210]{OBSOLETE} ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthGlobalTimeSupport`.⌋*()*

[SWS_Eth_00185]{OBSOLETE} ⌈*Eth_Init()* shall be called before *Eth_GetCurrentTime()*.⌋*()*

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_GetCurrentTime (via EthIf_GetCurrentTime) API, means the call of Eth_GetCurrentTime could happen in another partition.

[SWS_Eth_00262]{OBSOLETE} ⌈The Eth module shall apply appropriate mechanisms to allow calls of Eth_GetCurrentTime API from other partitions than its main function, e.g. by providing an Eth satellite.⌋*()*

8.3.15 Eth_GetCurrentTimeTuple

[SWS_Eth_91017]{DRAFT} **Definition of API function Eth_GetCurrentTimeTuple** ⌈

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 oft 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	

⌋*(SRS_Eth_00175)*

[SWS_Eth_00345]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*(SRS_BSW_00386)*

[SWS_Eth_00346]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*(SRS_BSW_00386)*

[SWS_Eth_00347]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter ClkUnitIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CLKUNIT_IDX.⌋*(SRS_BSW_00386)*

[SWS_Eth_00348]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter currentTimeTuplePtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*(SRS_BSW_-00386)*

[SWS_Eth_00349]{DRAFT} ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthPhcSupport`.⌋*(SRS_BSW_00171)*

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]{DRAFT} ⌈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.⌋*(SRS_BSW_00459)*

8.3.16 Eth_SetPhcTime

[SWS_Eth_91018]{DRAFT} **Definition of API function Eth_SetPhcTime** ⌈

Service Name	Eth_SetPhcTime (draft)	

Syntax	`Std_ReturnType Eth_SetPhcTime (`
	`  uint8 CtrlIdx,`
	`  uint8 ClkUnitIdx,`
	`  const TimeStampType* timeStampPtr`
	`)`	

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 oft 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	

⌋*(SRS_Eth_00167)*

[SWS_Eth_00352]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*(SRS_BSW_00386)*

[SWS_Eth_00353]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*(SRS_BSW_00386)*

[SWS_Eth_00354]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter ClkUnitIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CLKUNIT_IDX.⌋*(SRS_BSW_00386)*

[SWS_Eth_00355]{DRAFT} ⌈The function shall be pre compile time configurable On/ Off by the configuration parameter: `EthPhcSupport`.⌋*(SRS_BSW_00171)*

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]{DRAFT} ⌈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.⌋*(SRS_BSW_00459)*

8.3.17 Eth_SetPhcCorrection

[SWS_Eth_91019]{DRAFT} **Definition of API function Eth_SetPhcCorrection** ⌈

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^{-41}), by which the PHC is requested to be corrected
	offset	Time offset, by which the PHC is requested to be updated.
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	

⌋*(SRS_Eth_00167)*

[SWS_Eth_00369]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*(SRS_BSW_00386)*

[SWS_Eth_00370]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*(SRS_BSW_00386)*

[SWS_Eth_00371]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter ClkUnitIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CLKUNIT_IDX.⌋*(SRS_BSW_00386)*

[SWS_Eth_00372]{DRAFT} ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthPhcSupport`.⌋*(SRS_BSW_00171)*

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]{DRAFT} ⌈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.⌋*(SRS_BSW_00459)*

8.3.18 Eth_GetPhcTime

[SWS_Eth_91020]{DRAFT} **Definition of API function Eth_GetPhcTime** ⌈

Service Name	Eth_GetPhcTime (draft)	
Syntax	`Std_ReturnType Eth_GetPhcTime ( uint8 CtrlIdx, uint8 ClkUnitIdx, TimeStampQualType* timeQualPtr, TimeStampType* timeStampPtr )`	
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 oft 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

▽

△

Description	Returns the current time value out of the HW registers of the PHC. **Tags:** atp.Status=draft
Available via	Eth.h

⌋*(SRS_Eth_00175)*

[SWS_Eth_00358]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*(SRS_BSW_00386)*

[SWS_Eth_00359]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*(SRS_BSW_00386)*

[SWS_Eth_00360]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter ClkUnitIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CLKUNIT_IDX.⌋*(SRS_BSW_00386)*

[SWS_Eth_00361]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*(SRS_BSW_00386)*

[SWS_Eth_00362]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter timeStampPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*(SRS_BSW_00386)*

[SWS_Eth_00363]{DRAFT} ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthPhcSupport`.⌋*(SRS_BSW_00171)*

[SWS_Eth_00364]{DRAFT} ⌈*Eth_Init()* shall be called before *Eth_GetPhcTime()*.⌋*(SRS_BSW_00101)*

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]{DRAFT} ⌈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.⌋*(SRS_BSW_00459)*

8.3.19 Eth_SetPpsSignalMode

[SWS_Eth_91021]{DRAFT} **Definition of API function Eth_SetPpsSignalMode** ⌈

Service Name	Eth_SetPpsSignalMode (draft)	
Syntax	`Std_ReturnType Eth_SetPpsSignalMode ( uint8 CtrlIdx, uint8 ClkUnitIdx, boolean signalMode )`	
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 oft 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	

⌋*(SRS_Eth_00176)*

[SWS_Eth_00366]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*(SRS_BSW_00386)*

[SWS_Eth_00367]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*(SRS_BSW_00386)*

[SWS_Eth_00368]{DRAFT} ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthPhcSupport`.⌋*(SRS_BSW_00171)*

8.3.20 Eth_EnableEgressTimeStamp

[SWS_Eth_00186] Definition of API function Eth_EnableEgressTimeStamp ⌈

Service Name	Eth_EnableEgressTimeStamp
Syntax	`void Eth_EnableEgressTimeStamp ( uint8 CtrlIdx, Eth_BufIdxType BufIdx )`
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_00187] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00188] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[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()*.⌋*()*

8.3.21 Eth_GetEgressTimeStamp

[SWS_Eth_00190] Definition of API function Eth_GetEgressTimeStamp ⌈

Service Name	Eth_GetEgressTimeStamp	

Syntax	`Std_ReturnType Eth_GetEgressTimeStamp (`
	`  uint8 CtrlIdx,`
	`  Eth_BufIdxType BufIdx,`
	`  TimeStampQualType* timeQualPtr,`
	`  TimeStampType* timeStampPtr`
	`)`	

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_00191] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00192] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00193] ⌈If development error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

[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()*.⌋*()*

8.3.22 Eth_GetIngressTimeStamp

[SWS_Eth_00195] Definition of API function Eth_GetIngressTimeStamp ⌈

Service Name	Eth_GetIngressTimeStamp	

Syntax	`Std_ReturnType Eth_GetIngressTimeStamp (`
	`  uint8 CtrlIdx,`
	`  const Eth_DataType* DataPtr,`
	`  TimeStampQualType* timeQualPtr,`
	`  TimeStampType* timeStampPtr`
	`)`	

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.
Available via	Eth.h

⌋*()*

[SWS_Eth_00196] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00197] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00198] ⌈If development error detection is enabled: the function shall check the parameter DataPtr, timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

[SWS_Eth_00213] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: EthGlobalTimeSupport [ECUC_Eth_00037].⌋*()*

[SWS_Eth_00199] ⌈*Eth_Init()* shall be called before *Eth_GetIngressTimeStamp()*.⌋*()*

8.3.23 Eth_ProvideTxBuffer

[SWS_Eth_00077] Definition of API function Eth_ProvideTxBuffer ⌈

Service Name	Eth_ProvideTxBuffer	

Syntax	`BufReq_ReturnType Eth_ProvideTxBuffer (`
	`  uint8 CtrlIdx,`
	`  uint8 Priority,`
	`  Eth_BufIdxType* BufIdxPtr,`
	`  uint8** BufPtr,`
	`  uint16* LenBytePtr`
	`)`	

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_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_00081] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00082] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00083] ⌈If development error detection is enabled: the function shall check the parameter BufIdxPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

[SWS_Eth_00084] ⌈If development error detection is enabled: the function shall check the parameter BufPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

[SWS_Eth_00085] ⌈If development error detection is enabled: the function shall check the parameter LenBytePtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

[SWS_Eth_00086] ⌈*Eth_Init()* shall be called before *Eth_ProvideTxBuffer()*.⌋*()*

8.3.24 Eth_Transmit

[SWS_Eth_00087] Definition of API function Eth_Transmit ⌈

Service Name	Eth_Transmit	
Syntax	`Std_ReturnType Eth_Transmit ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, Eth_FrameType FrameType , boolean TxConfirmation, uint16 LenByte, const uint8* PhysAddrPtr )`	
Service ID [hex]	0xA	
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_00090] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00091] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[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_00093] ⌈If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

[SWS_Eth_00129] ⌈If development error detection is enabled: the function shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function shall raise the development error ETH_E_INV_MODE.⌋*()*

[SWS_Eth_00094] ⌈*Eth_ProvideTxBuffer()* shall be called before *Eth_Transmit*.⌋*()*

8.3.25 Eth_Receive

[SWS_Eth_00095] Definition of API function Eth_Receive ⌈

Service Name	Eth_Receive	

Syntax	`void Eth_Receive (`
	`  uint8 CtrlIdx,`
	`  uint8 QueueIdx,`
	`  Eth_RxStatusType* RxStatusPtr`
	`)`	

Service ID [hex]	0xB	
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_00097] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00098] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00132] ⌈If development error detection is enabled: the function shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function shall raise the development error ETH_E_INV_MODE.⌋*()*

[SWS_Eth_00099] ⌈*Eth_Init()* shall be called before *Eth_Receive()*.⌋*()*

8.3.26 Eth_ImmediateTransmit

[SWS_Eth_91022]{DRAFT} **Definition of API function Eth_ImmediateTransmit** ⌈

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	

⌋*(SRS_Eth_00173)*

8.3.27 Eth_ReleaseRxBuffer

[SWS_Eth_91023]{DRAFT} **Definition of API function Eth_ReleaseRxBuffer** ⌈

Service Name	Eth_ReleaseRxBuffer (draft)	

Syntax	`void Eth_ReleaseRxBuffer (`
	`  uint8 CtrlIdx,`
	`  Eth_BufIdxType RxHandleId`
	`)`	

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

⌋*(SRS_Eth_00172)*

8.3.28 Eth_TxConfirmation

[SWS_Eth_00100] Definition of API function Eth_TxConfirmation ⌈

Service Name	Eth_TxConfirmation	

Syntax	`void Eth_TxConfirmation (`
	`  uint8 CtrlIdx`
	`)`	

Service ID [hex]	0xC	
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_00103] ⌈If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_UNINIT.⌋*()*

[SWS_Eth_00104] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.⌋*()*

[SWS_Eth_00134] ⌈If development error detection is enabled: the function shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function shall raise the development error ETH_E_INV_MODE.⌋*()*

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

8.3.29 Eth_GetVersionInfo

[SWS_Eth_00106] Definition of API function Eth_GetVersionInfo ⌈

Service Name	Eth_GetVersionInfo	
Syntax	`void Eth_GetVersionInfo ( Std_VersionInfoType* VersionInfoPtr )`	
Service ID [hex]	0xD	
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	

⌋*()*

[SWS_Eth_00136] ⌈If development error detection is enabled: the function shall check the parameter VersionInfoPtr for being valid. If the check fails, the function shall raise the development error ETH_E_PARAM_POINTER.⌋*()*

8.4 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]).

8.5 Scheduled functions

8.5.1 Eth_MainFunction

[SWS_Eth_00171] Definition of scheduled function Eth_MainFunction ⌈

Service Name	Eth_MainFunction
Syntax	`void Eth_MainFunction ( void )`

▽

△

Service ID [hex]	0x20
Description	The function checks for controller errors and lost frames. Used for polling state changes. Calls EthIf_CtrlModeIndication 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_CtrlModeIndication when the controller mode changed.⌋*()*

8.6 Expected interfaces

This chapter lists all interfaces required from other modules.

8.6.1 Mandatory Interfaces

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

[SWS_Eth_00119] Definition of mandatory interfaces in 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_CtrlModeIndication	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

▽

△

API Function	*Header File*	*Description*
EthIf_TxConfirmation	EthIf.h	Confirms frame transmission by the indexed controller
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.

⌋*()*

8.6.2 Optional Interfaces

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

[SWS_Eth_00120] Definition of optional interfaces in module Eth ⌈

API Function	*Header File*	*Description*
Det_ReportError	Det.h	Service to report development errors.
EthSwt_EthRxFinishedIndication	EthSwt_Eth.h	Indication for a finished receive process for a specific Ethernet frame, which results in providing the management information retrieved during EthSwt_EthRxProcessFrame().
EthSwt_EthRxProcessFrame	EthSwt_Eth.h	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in EthSwt_EthRxFinishedIndication().
EthSwt_EthTxAdaptBufferLength	EthSwt_Eth.h	Modifies the buffer length to be able to insert management information.
EthSwt_EthTxFinishedIndication	EthSwt_Eth.h	Indication for a finished transmit process for a specific Ethernet frame.
EthSwt_EthTxPrepareFrame	EthSwt_Eth.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 EthSwt_EthTxFinishedIndication().
EthSwt_EthTxProcessFrame	EthSwt_Eth.h	Function inserts management information into the Ethernet frame.
Icu_DisableNotification	Icu.h	This function disables the notification of a channel.
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.

⌋*()*

8.6.3 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).

8.6.3.1 Eth_<IngressQueueHandlerFunction>

[SWS_Eth_91024]{DRAFT} **Definition of configurable interface Eth_<IngressQueueHandlerFunction>(void)** ⌈

Service Name	Eth_<IngressQueueHandlerFunction>(void) (draft)

Syntax	`void Eth_<IngressQueueHandlerFunction>(void) (`
	`  void`
	`)`

Sync/Async	–
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	

⌋*(SRS_Eth_00174)*

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 the chapter 10.1 "Introduction to configuration specification" in *SWS_BSWGeneral* [3].

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]{DRAFT} ⌈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.⌋*(SRS_BSW_00159)*

10.2.1 Eth

SWS Item	**[ECUC_Eth_00038]**
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**	**Scope / 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

SWS Item	**[ECUC_Eth_00015]**
Container Name	EthConfigSet
Parent Container	Eth
Description	This container contains the configuration parameters and sub containers of the AUTOSAR Eth module.
Configuration Parameters	

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

10.2.3 EthCtrlConfig

SWS Item	**[ECUC_Eth_00006]**
Container Name	EthCtrlConfig
Parent Container	EthConfigSet
Description	Configuration of the individual controller
Configuration Parameters	

SWS Item	**[ECUC_Eth_00135]**
Parameter Name	EthCtrEgressHardwareSupportedDataTransferThreshold
Parent Container	EthCtrlConfig
Description	EthCtrEgressHardwareSupportedDataTransferThreshold define a threshold in bytes, if data, which is requested to be transmitted, shall be transfered perfromed 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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00071]**		
Parameter Name	EthCtrlConfigSwBufferHandling		
Parent Container	EthCtrlConfig		
Description	Enables / Disables SW buffer management		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00130]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00012]**
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	–	
Scope / Dependency	scope: local dependency: This parameter shall be set to TRUE, if EthCtrlEnableSpiInterface is set to TRUE		

SWS Item	**[ECUC_Eth_00010]**		
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 EthCtrl EnableIngressQueueInterrupt 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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00073]**		
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 TC6 [26].		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00011]**
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00007]**		
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	–	
Scope / Dependency	scope: ECU withAuto = true		

SWS Item	**[ECUC_Eth_00063]**		
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	–	
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, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_Eth_00062]**		
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, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_Eth_00039]**		
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) bandwith class 1Gbit/s (e.g. GMII, RGMII, SGMII, RvGMII, USGMII)	
	ETH_MAC_LAYER_TYPE_XMII	MAC layer interface (data) bandwith class 10-100Mbit/s (e.g. MII, RMII, RvMII, SMII)	
	ETH_MAC_LAYER_TYPE_XXGMII	MAC layer interface (data) bandwith class 10Gbit/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
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_Eth_00020]**
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00065]**		
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	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / 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 controler egress behavior.
EthCtrlConfigIngress	1	Configuration of one Ethernet controler ingress behavior.

▽

△

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthCtrlConfigSpiConfiguration	0..*	SPI Interface configuration of one Ethernet controller (MACPHY use). Configured only if EthCtrlEnableSpiInterface is set to TRUE. **Tags:** atp.Status=draft
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.

[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 EthCtrlEcucPartitionRef references one or more ECUC partitions, EthCtrlEcucPartitionRef shall have a multiplicity of one and reference one of these ECUC partitions as well.⌋*()*

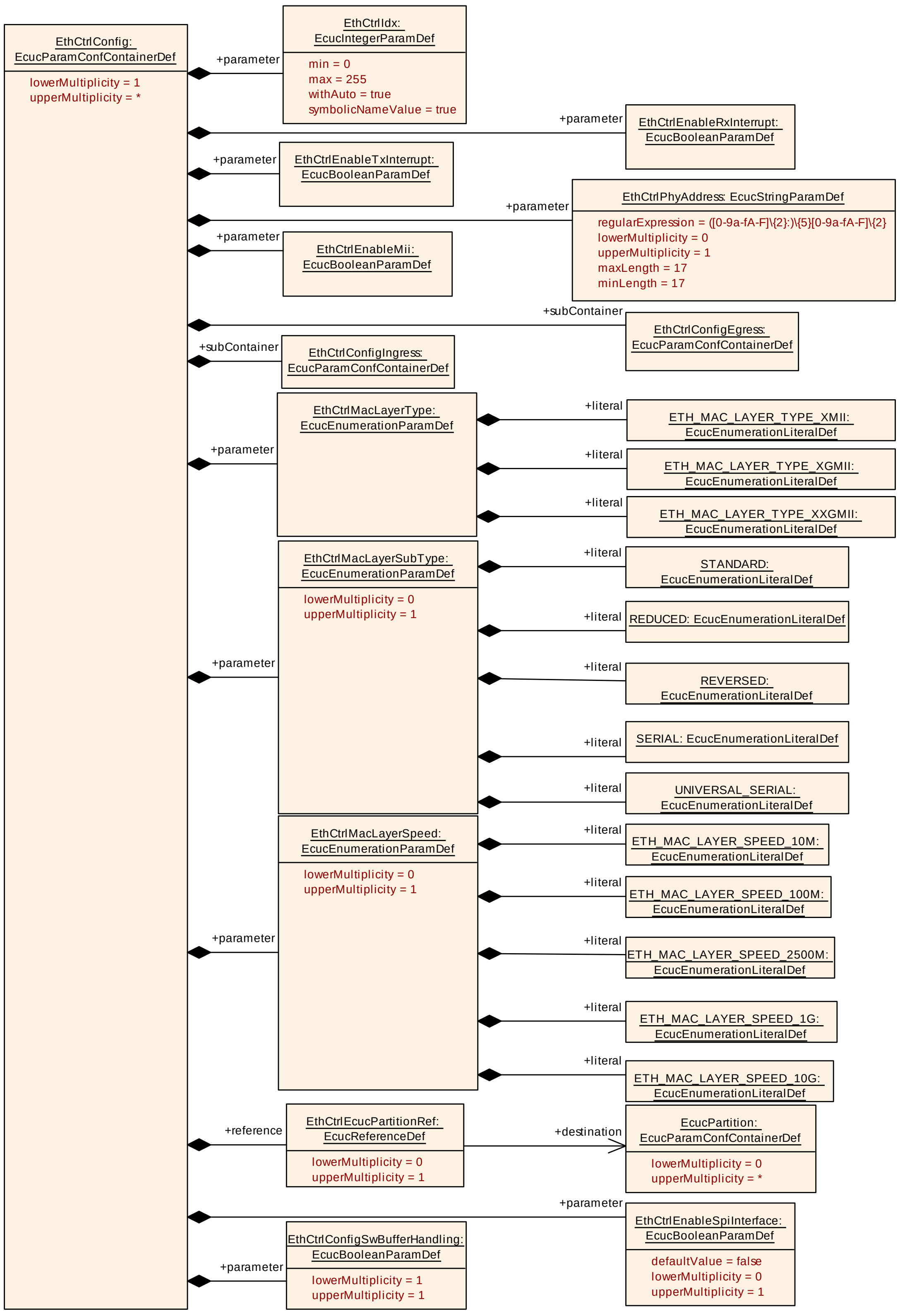

EthCtrlConfig: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
EthCtrlIdx: EcucIntegerParamDef
min = 0
max = 255
withAuto = true
symbolicNameValue = true
+parameter
EthCtrlEnableRxInterrupt: EcucBooleanParamDef
+parameter
EthCtrlEnableTxInterrupt: EcucBooleanParamDef
+parameter
EthCtrlPhyAddress: EcucStringParamDef
regularExpression = ([0-9a-fA-F]\{2}:)\{5}[0-9a-fA-F]\{2}
lowerMultiplicity = 0
upperMultiplicity = 1
maxLength = 17
minLength = 17
+parameter
EthCtrlEnableMii: EcucBooleanParamDef
+subContainer
EthCtrlConfigEgress: EcucParamConfContainerDef
+subContainer
EthCtrlConfigIngress: EcucParamConfContainerDef
+parameter
EthCtrlMacLayerType: EcucEnumerationParamDef
+literal
ETH_MAC_LAYER_TYPE_XMII: EcucEnumerationLiteralDef
+literal
ETH_MAC_LAYER_TYPE_XGMII: EcucEnumerationLiteralDef
+literal
ETH_MAC_LAYER_TYPE_XXGMII: EcucEnumerationLiteralDef
+parameter
EthCtrlMacLayerSubType: EcucEnumerationParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
+literal
STANDARD: EcucEnumerationLiteralDef
+literal
REDUCED: EcucEnumerationLiteralDef
+literal
REVERSED: EcucEnumerationLiteralDef
+literal
SERIAL: EcucEnumerationLiteralDef
+literal
UNIVERSAL_SERIAL: EcucEnumerationLiteralDef
+parameter
EthCtrlMacLayerSpeed: EcucEnumerationParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
+literal
ETH_MAC_LAYER_SPEED_10M: EcucEnumerationLiteralDef
+literal
ETH_MAC_LAYER_SPEED_100M: EcucEnumerationLiteralDef
+literal
ETH_MAC_LAYER_SPEED_2500M: EcucEnumerationLiteralDef
+literal
ETH_MAC_LAYER_SPEED_1G: EcucEnumerationLiteralDef
+literal
ETH_MAC_LAYER_SPEED_10G: EcucEnumerationLiteralDef
+reference
EthCtrlEcucPartitionRef: EcucReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
EcucPartition: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+parameter
EthCtrlEnableSpiInterface: EcucBooleanParamDef
defaultValue = false
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthCtrlConfigSwBufferHandling: EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1

Figure 10.1: Overview EthCtrlConfig configuration

10.2.4 EthCtrlClk and EthClkUnit

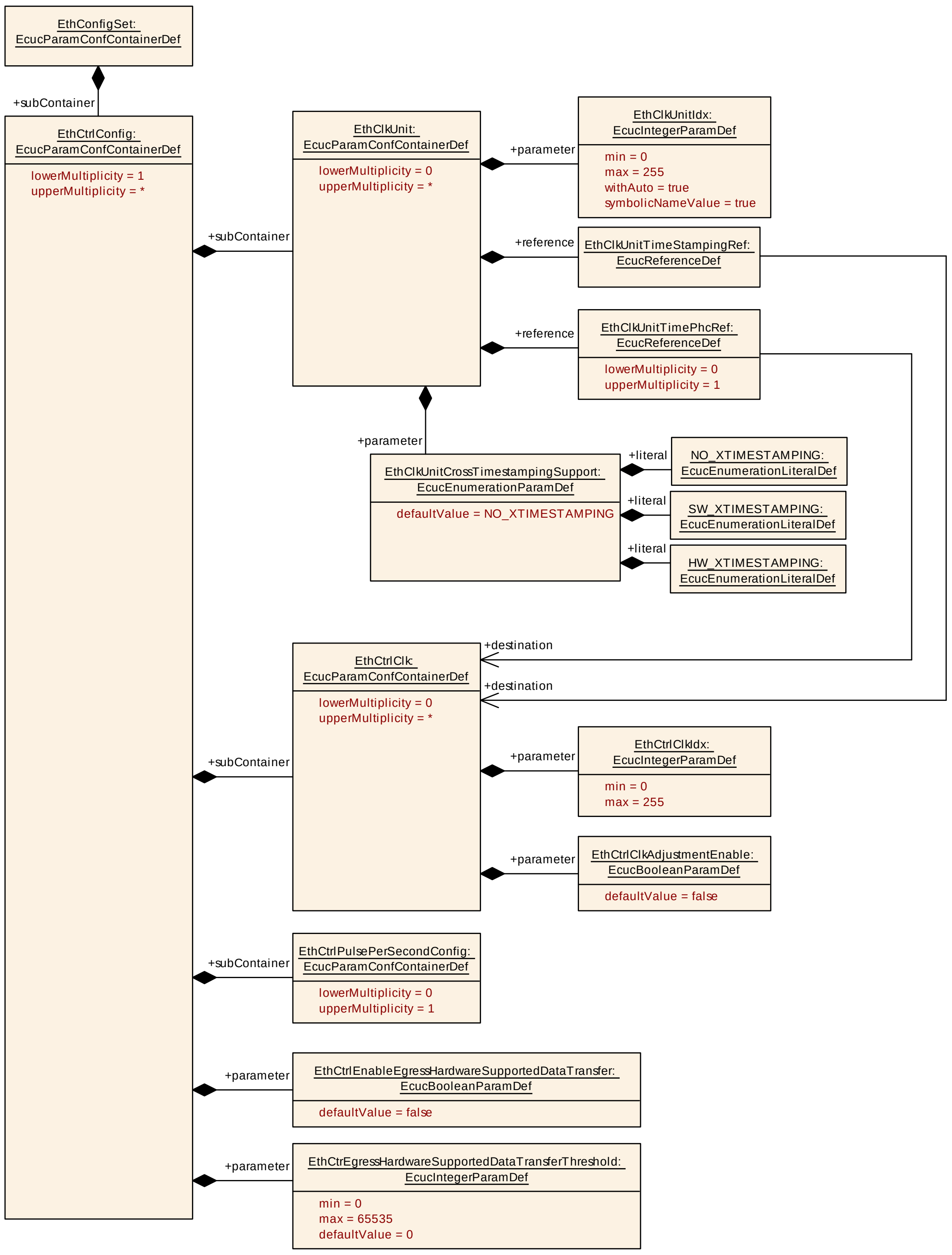

EthConfigSet: EcucParamConfContainerDef
+subContainer
EthCtrlConfig: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+subContainer
EthClkUnit: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+parameter
EthClkUnitIdx: EcucIntegerParamDef
min = 0
max = 255
withAuto = true
symbolicNameValue = true
+reference
EthClkUnitTimeStampingRef: EcucReferenceDef
+reference
EthClkUnitTimePhcRef: EcucReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthClkUnitCrossTimestampingSupport: EcucEnumerationParamDef
defaultValue = NO_XTIMESTAMPING
+literal
NO_XTIMESTAMPING: EcucEnumerationLiteralDef
+literal
SW_XTIMESTAMPING: EcucEnumerationLiteralDef
+literal
HW_XTIMESTAMPING: EcucEnumerationLiteralDef
+destination
+destination
EthCtrlClk: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+subContainer
+parameter
EthCtrlClkIdx: EcucIntegerParamDef
min = 0
max = 255
+parameter
EthCtrlClkAdjustmentEnable: EcucBooleanParamDef
defaultValue = false
+subContainer
EthCtrlPulsePerSecondConfig: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthCtrlEnableEgressHardwareSupportedDataTransfer: EcucBooleanParamDef
defaultValue = false
+parameter
EthCtrEgressHardwareSupportedDataTransferThreshold: EcucIntegerParamDef
min = 0
max = 65535
defaultValue = 0

Figure 10.2: Overview EthCtrlClk and EthClkUnit

SWS Item	**[ECUC_Eth_00115]**		
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		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

SWS Item	**[ECUC_Eth_00114]**		
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	–	
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_Eth_00113]**		
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	–	
Scope / Dependency	scope: ECU		

No Included Containers

SWS Item	**[ECUC_Eth_00120]**		
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		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

SWS Item	**[ECUC_Eth_00119]**		
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	–	
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_Eth_00118]**	
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	–	
Scope / Dependency	scope: ECU withAuto = true		

SWS Item	**[ECUC_Eth_00117]**		
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	–	
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_Eth_00116]**		
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/egrees 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	–	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.5 EthCtrlPulsePerSecondConfig

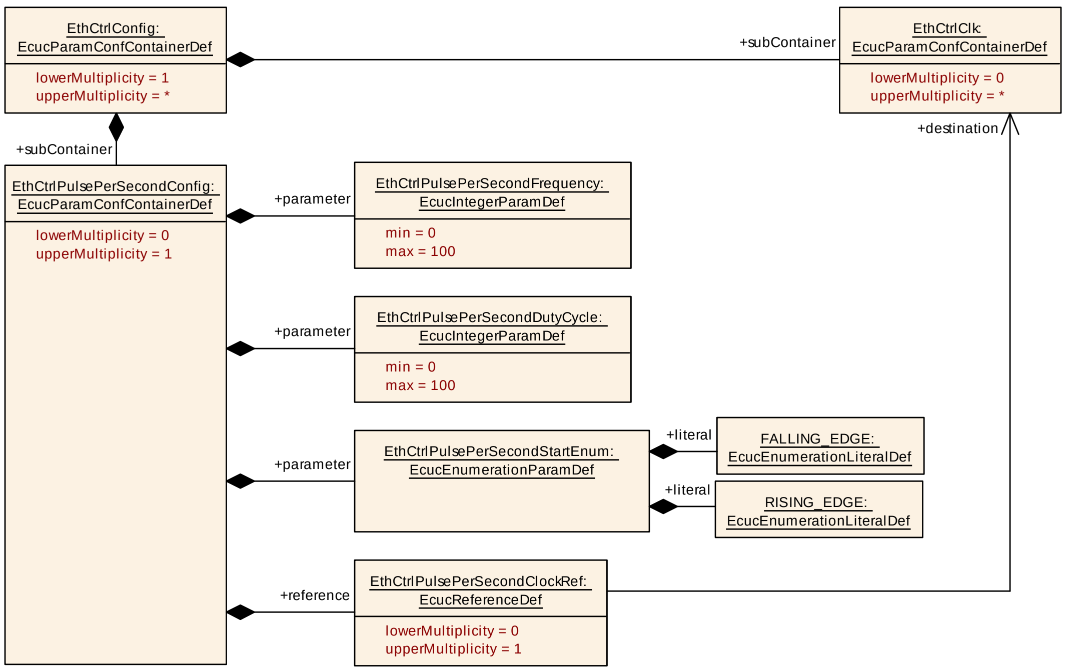

EthCtrlConfig: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+subContainer
EthCtrlClk: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+destination
+subContainer
EthCtrlPulsePerSecondConfig: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthCtrlPulsePerSecondFrequency: EcucIntegerParamDef
min = 0
max = 100
+parameter
EthCtrlPulsePerSecondDutyCycle: EcucIntegerParamDef
min = 0
max = 100
+parameter
EthCtrlPulsePerSecondStartEnum: EcucEnumerationParamDef
+literal
FALLING_EDGE: EcucEnumerationLiteralDef
+literal
RISING_EDGE: EcucEnumerationLiteralDef
+reference
EthCtrlPulsePerSecondClockRef: EcucReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1

Figure 10.3: EthCtrlPulsePerSecondConfig

SWS Item	**[ECUC_Eth_00111]**		
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		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

SWS Item	**[ECUC_Eth_00110]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00109]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00108]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00112]**		
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	–	
Scope / Dependency	scope: local		

No Included Containers

10.2.6 EthCtrlConfigEgress

SWS Item	**[ECUC_Eth_00046]**
Container Name	EthCtrlConfigEgress
Parent Container	EthCtrlConfig
Description	Configuration of one Ethernet controler egress behavior.
Configuration Parameters	

SWS Item	**[ECUC_Eth_00052]**		
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	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthCtrlConfigEgressFifo	0..*	Represents a Fifo at the egress side. **Tags:** atp.Status=obsolete
EthCtrlConfigEgressQueue	1..8	Represents a queue at the egress side. **Tags:** atp.Status=draft
EthCtrlConfigScheduler	1..*	Represents a Scheduler on the egress side.
EthCtrlConfigShaper	0..*	Represents a Shaper an the egress side. **Tags:** atp.Status=obsolete

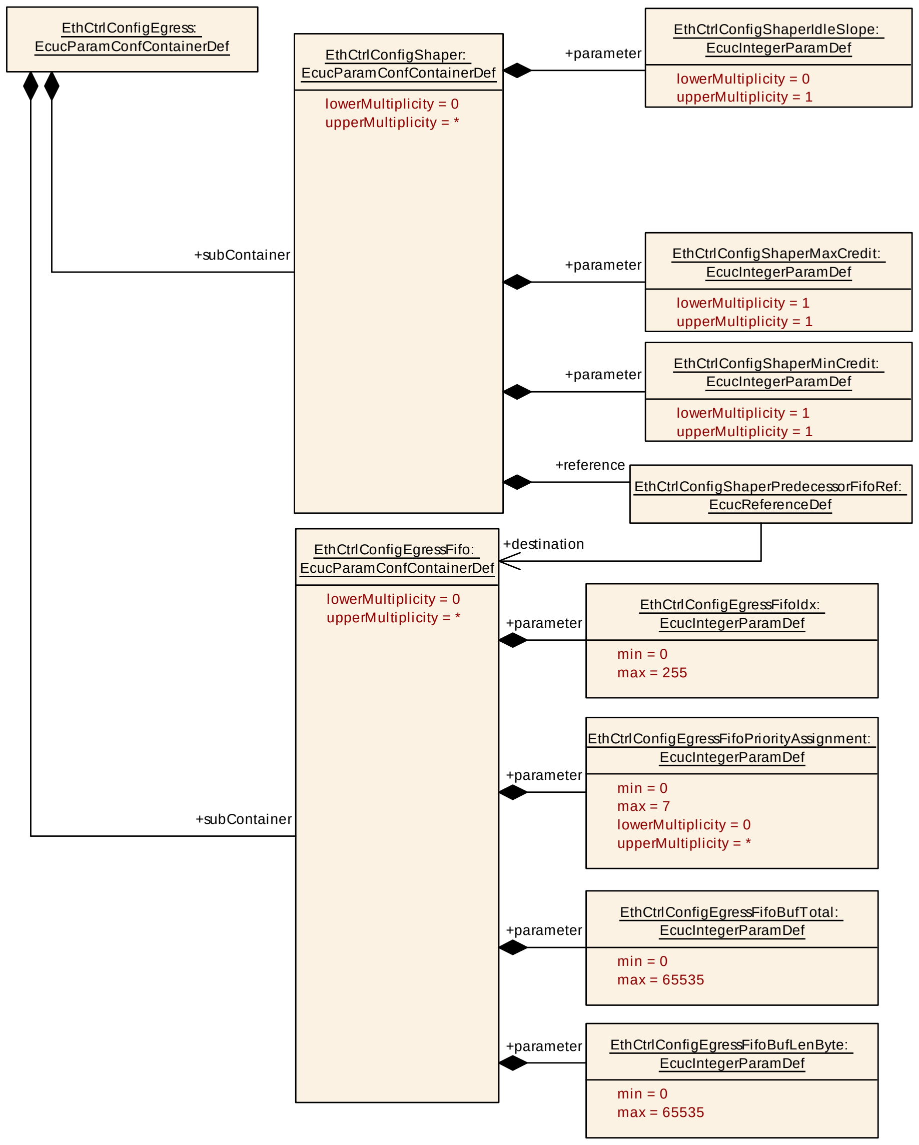

EthCtrlConfigEgress: EcucParamConfContainerDef
EthCtrlConfigShaper: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+subContainer
+parameter
EthCtrlConfigShaperIdleSlope: EcucIntegerParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthCtrlConfigShaperMaxCredit: EcucIntegerParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
EthCtrlConfigShaperMinCredit: EcucIntegerParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
+reference
EthCtrlConfigShaperPredecessorFifoRef: EcucReferenceDef
+destination
EthCtrlConfigEgressFifo: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+parameter
EthCtrlConfigEgressFifoIdx: EcucIntegerParamDef
min = 0
max = 255
+parameter
EthCtrlConfigEgressFifoPriorityAssignment: EcucIntegerParamDef
min = 0
max = 7
lowerMultiplicity = 0
upperMultiplicity = *
+subContainer
+parameter
EthCtrlConfigEgressFifoBufTotal: EcucIntegerParamDef
min = 0
max = 65535
+parameter
EthCtrlConfigEgressFifoBufLenByte: EcucIntegerParamDef
min = 0
max = 65535

Figure 10.4: Overview EthCtrlConfigEgress configuration - OBSOLETE

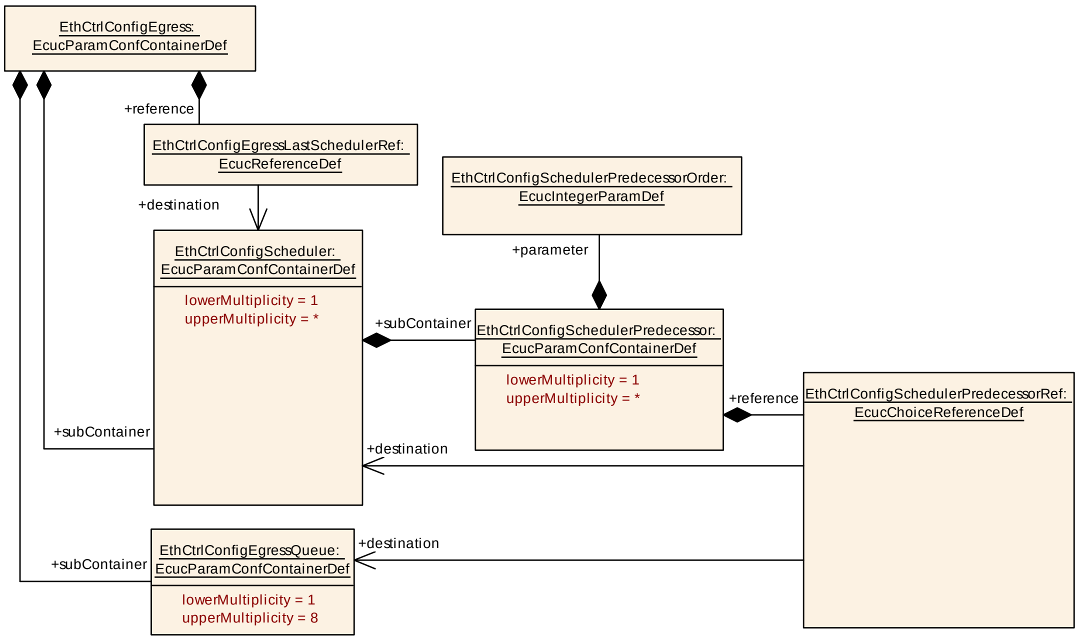

EthCtrlConfigEgress: EcucParamConfContainerDef
+reference
EthCtrlConfigEgressLastSchedulerRef: EcucReferenceDef
+destination
EthCtrlConfigSchedulerPredecessorOrder: EcucIntegerParamDef
+parameter
EthCtrlConfigScheduler: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+subContainer
EthCtrlConfigSchedulerPredecessor: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+reference
EthCtrlConfigSchedulerPredecessorRef: EcucChoiceReferenceDef
+subContainer
+destination
+subContainer
EthCtrlConfigEgressQueue: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 8
+destination

Figure 10.5: Overview EthCtrlConfigEgress configuration - DRAFT

10.2.6.1 EthCtrlConfigEgressFifo - OBSOLETE

SWS Item	**[ECUC_Eth_00047]** (Obsolete)		
Container Name	EthCtrlConfigEgressFifo		
Parent Container	EthCtrlConfigEgress		
Description	Represents a Fifo at the egress side. **Tags:** atp.Status=obsolete		
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			

SWS Item	**[ECUC_Eth_00051]** (Obsolete)		
Parameter Name	EthCtrlConfigEgressFifoBufLenByte		
Parent Container	EthCtrlConfigEgressFifo		
Description	Length of Fifo elements in bytes. **Tags:** atp.Status=obsolete		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00050]** (Obsolete)		
Parameter Name	EthCtrlConfigEgressFifoBufTotal		
Parent Container	EthCtrlConfigEgressFifo		
Description	Fifo buffer count. **Tags:** atp.Status=obsolete		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00048]** (Obsolete)		
Parameter Name	EthCtrlConfigEgressFifoIdx		
Parent Container	EthCtrlConfigEgressFifo		
Description	Egress Fifo index. **Tags:** atp.Status=obsolete		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00049]** (Obsolete)		
Parameter Name	EthCtrlConfigEgressFifoPriorityAssignment		
Parent Container	EthCtrlConfigEgressFifo		
Description	Message egress prority assignment. **Tags:** atp.Status=obsolete		
Multiplicity	0..*		
Type	EcucIntegerParamDef		
Range	0 .. 7		
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
Scope / Dependency	scope: local		

No Included Containers

10.2.6.2 EthCtrlConfigEgressQueue - DRAFT

SWS Item	**[ECUC_Eth_00090]**		
Container Name	EthCtrlConfigEgressQueue		
Parent Container	EthCtrlConfigEgress		
Description	Represents a queue at the egress side. **Tags:** atp.Status=draft		
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			

SWS Item	**[ECUC_Eth_00092]**		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00093]**
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00091]**		
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	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local withAuto = true		

SWS Item	**[ECUC_Eth_00094]**		
Parameter Name	EthCtrlConfigEgressQueuePriorityAssignment		
Parent Container	EthCtrlConfigEgressQueue		
Description	Defines the egress queue priority assignment. **Tags:** atp.Status=draft		
Multiplicity	0..8		
Type	EcucIntegerParamDef		
Range	0 .. 7		
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
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthCtrlConfigEgressQueue TransmissionSelection	1	Represents the transmission selection of a queue at the egress side. **Tags:** atp.Status=draft

10.2.6.3 EthCtrlConfigEgressQueueTransmissionSelection - DRAFT

SWS Item	**[ECUC_Eth_00100]**
Container Name	EthCtrlConfigEgressQueueTransmissionSelection
Parent Container	EthCtrlConfigEgressQueue
Description	Represents the transmission selection of a queue at the egress side. **Tags:** atp.Status=draft
Configuration Parameters	

SWS Item	**[ECUC_Eth_00106]**		
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 EthCtrlConfigScheduler. **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
Scope / Dependency	scope: local dependency: If EthCtrlConfigSwBufferHandling is set to TRUE, then EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm shall be set to ETH_TRANSMISSION_SELECTION_CBS.		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthCtrlConfigEgressQueue TransmissionSelectionCBSConfig	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 EthCtrlConfigEgressQueue TransmissionSelectionAlgorithm is set to ETH_ TRANSMISSION_SELECTION_CBS. **Tags:** atp.Status=draft

10.2.6.4 EthCtrlConfigEgressQueueTransmissionSelectionCBSConfig - DRAFT

SWS Item	**[ECUC_Eth_00101]**		
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. **Tags:** atp.Status=draft		
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			

SWS Item	**[ECUC_Eth_00103]**		
Parameter Name	EthCtrlConfigEgressQueueCreditBasedShaperIdleSlope		
Parent Container	EthCtrlConfigEgressQueueTransmissionSelectionCBSConfig		
Description	Defines the increase of credit in bits per second for the AVB shaper. Note: this parameter maps to IEEE802.1Q parameter "ieee8021FqtssAdminIdleSlope Ms" and "ieee8021FqtssAdminIdleSlopeLs". **Tags:** atp.Status=draft		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00102]**
Parameter Name	EthCtrlConfigEgressQueueCreditBasedShaperMaxCredit
Parent Container	EthCtrlConfigEgressQueueTransmissionSelectionCBSConfig

▽

△

Description	Defines the maximum amount of credits that can be accumulated for a queue. **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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00104]**		
Parameter Name	EthCtrlConfigEgressQueueCreditBasedShaperMinCredit		
Parent Container	EthCtrlConfigEgressQueueTransmissionSelectionCBSConfig		
Description	Defines the minimum amount of credits that can be accumulated for a queue. **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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00105]**	
Parameter Name	EthCtrlConfigEgressQueueCreditBasedShaperSendSlope	
Parent Container	EthCtrlConfigEgressQueueTransmissionSelectionCBSConfig	
Description	Defines the send slope of 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
Scope / Dependency	scope: local		

No Included Containers

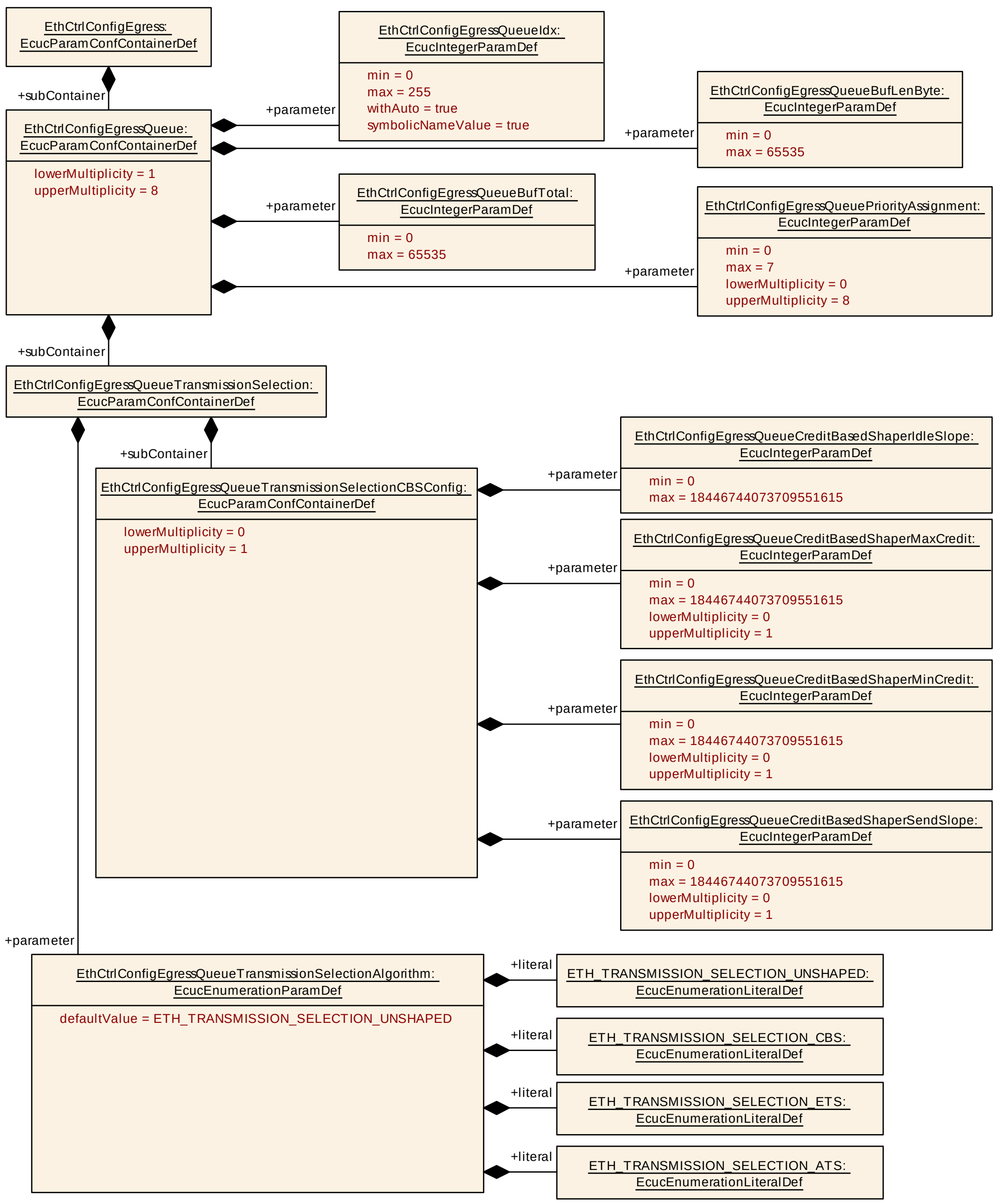

EthCtrlConfigEgress: EcucParamConfContainerDef
+subContainer
EthCtrlConfigEgressQueue: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 8
+parameter
EthCtrlConfigEgressQueueIdx: EcucIntegerParamDef
min = 0
max = 255
withAuto = true
symbolicNameValue = true
+parameter
EthCtrlConfigEgressQueueBufLenByte: EcucIntegerParamDef
min = 0
max = 65535
+parameter
EthCtrlConfigEgressQueueBufTotal: EcucIntegerParamDef
min = 0
max = 65535
+parameter
EthCtrlConfigEgressQueuePriorityAssignment: EcucIntegerParamDef
min = 0
max = 7
lowerMultiplicity = 0
upperMultiplicity = 8
+subContainer
EthCtrlConfigEgressQueueTransmissionSelection: EcucParamConfContainerDef
+subContainer
EthCtrlConfigEgressQueueTransmissionSelectionCBSConfig: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthCtrlConfigEgressQueueCreditBasedShaperIdleSlope: EcucIntegerParamDef
min = 0
max = 18446744073709551615
+parameter
EthCtrlConfigEgressQueueCreditBasedShaperMaxCredit: EcucIntegerParamDef
min = 0
max = 18446744073709551615
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthCtrlConfigEgressQueueCreditBasedShaperMinCredit: EcucIntegerParamDef
min = 0
max = 18446744073709551615
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthCtrlConfigEgressQueueCreditBasedShaperSendSlope: EcucIntegerParamDef
min = 0
max = 18446744073709551615
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthCtrlConfigEgressQueueTransmissionSelectionAlgorithm: EcucEnumerationParamDef
defaultValue = ETH_TRANSMISSION_SELECTION_UNSHAPED
+literal
ETH_TRANSMISSION_SELECTION_UNSHAPED: EcucEnumerationLiteralDef
+literal
ETH_TRANSMISSION_SELECTION_CBS: EcucEnumerationLiteralDef
+literal
ETH_TRANSMISSION_SELECTION_ETS: EcucEnumerationLiteralDef
+literal
ETH_TRANSMISSION_SELECTION_ATS: EcucEnumerationLiteralDef

Figure 10.6: Overview EthCtrlConfigEgressQueue configuration - DRAFT

10.2.6.5 EthCtrlConfigScheduler

SWS Item	[ECUC_Eth_00053]
Container Name	EthCtrlConfigScheduler
Parent Container	EthCtrlConfigEgress
Description	Represents a Scheduler on the egress side.
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
EthCtrlConfigScheduler Predecessor	1..*	Defines an ordered list of predecessors for this scheduler.

10.2.6.6 EthCtrlConfigSchedulerPredecessor

SWS Item	[ECUC_Eth_00054]
Container Name	EthCtrlConfigSchedulerPredecessor
Parent Container	EthCtrlConfigScheduler
Description	Defines an ordered list of predecessors for this scheduler.
Configuration Parameters	

SWS Item	[ECUC_Eth_00055]		
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
Scope / Dependency	scope: local		

SWS Item	[ECUC_Eth_00056]		
Parameter Name	EthCtrlConfigSchedulerPredecessorRef		
Parent Container	EthCtrlConfigSchedulerPredecessor		
Description	Choice reference to the scheduler predecessor.		
Multiplicity	1		
Type	Choice reference to [EthCtrlConfigEgressFifo, EthCtrlConfigEgressQueue, EthCtrlConfigScheduler, EthCtrlConfigShaper]		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	

▽

△

Scope / Dependency	scope: local

No Included Containers

10.2.6.7 EthCtrlConfigShaper

SWS Item	**[ECUC_Eth_00057]** (Obsolete)
Container Name	EthCtrlConfigShaper
Parent Container	EthCtrlConfigEgress
Description	Represents a Shaper an the egress side. **Tags:** atp.Status=obsolete
Configuration Parameters	

SWS Item	**[ECUC_Eth_00058]** (Obsolete)		
Parameter Name	EthCtrlConfigShaperIdleSlope		
Parent Container	EthCtrlConfigShaper		
Description	Defines the increase of credit in bits per second for the AVB shaper. **Tags:** atp.Status=obsolete		
Multiplicity	0..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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00069]** (Obsolete)		
Parameter Name	EthCtrlConfigShaperMaxCredit		
Parent Container	EthCtrlConfigShaper		
Description	Maximum amount of credits that can be accumulated for a queue. **Tags:** atp.Status=obsolete		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00070]** (Obsolete)		
Parameter Name	EthCtrlConfigShaperMinCredit		
Parent Container	EthCtrlConfigShaper		
Description	Minimum amount of credits in bytes that can be accumulated for a queue. **Tags:** atp.Status=obsolete		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00059]** (Obsolete)		
Parameter Name	EthCtrlConfigShaperPredecessorFifoRef		
Parent Container	EthCtrlConfigShaper		
Description	Reference to the fifo which is the predecessor for this shaper. **Tags:** atp.Status=obsolete		
Multiplicity	1		
Type	Reference to EthCtrlConfigEgressFifo		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.2.7 EthCtrlConfigIngress

SWS Item	**[ECUC_Eth_00040]**
Container Name	EthCtrlConfigIngress
Parent Container	EthCtrlConfig
Description	Configuration of one Ethernet controler ingress behavior.
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthCtrlConfigIngressFifo	0..*	Represents a Fifo at the ingress side. **Tags:** atp.Status=obsolete
EthCtrlConfigIngressQueue	0..*	Represents a queue at the ingress side. **Tags:** atp.Status=draft

▽

△

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthCtrlConfigIngressQueueSorting Priority	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 EthCtrlConfigIngressQueue SortingTypes 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

SWS Item	**[ECUC_Eth_00132]**		
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		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

SWS Item	**[ECUC_Eth_00124]**		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00123]**		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00122]**		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00121]**		
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

▽

△

Scope / Dependency	scope: local

No Included Containers

10.2.7.1 EthCtrlConfigIngressFifo - OBSOLETE

SWS Item	**[ECUC_Eth_00041]** (Obsolete)		
Container Name	EthCtrlConfigIngressFifo		
Parent Container	EthCtrlConfigIngress		
Description	Represents a Fifo at the ingress side. **Tags:** atp.Status=obsolete		
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			

SWS Item	**[ECUC_Eth_00045]** (Obsolete)		
Parameter Name	EthCtrlConfigIngressFifoBufLenByte		
Parent Container	EthCtrlConfigIngressFifo		
Description	Length of Fifo elements in bytes. **Tags:** atp.Status=obsolete		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00044]** (Obsolete)		
Parameter Name	EthCtrlConfigIngressFifoBufTotal		
Parent Container	EthCtrlConfigIngressFifo		
Description	Fifo buffer count. **Tags:** atp.Status=obsolete		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00043]** (Obsolete)		
Parameter Name	EthCtrlConfigIngressFifoIdx		
Parent Container	EthCtrlConfigIngressFifo		
Description	Ingress Fifo index. **Tags:** atp.Status=obsolete		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
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
Scope / Dependency	scope: local withAuto = true		

SWS Item	**[ECUC_Eth_00042]** (Obsolete)		
Parameter Name	EthCtrlConfigIngressFifoPriorityAssignment		
Parent Container	EthCtrlConfigIngressFifo		
Description	Message ingress prority assignment. **Tags:** atp.Status=obsolete		
Multiplicity	0..*		
Type	EcucIntegerParamDef		
Range	0 .. 7		
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
Scope / Dependency	scope: local		

No Included Containers

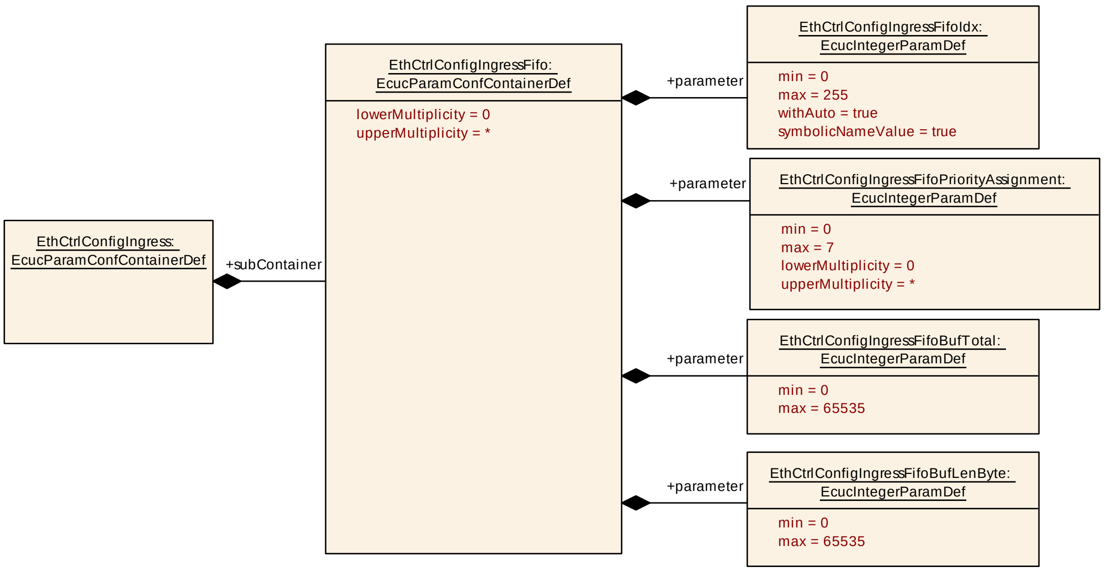

EthCtrlConfigIngress: EcucParamConfContainerDef
+subContainer
EthCtrlConfigIngressFifo: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+parameter
EthCtrlConfigIngressFifoIdx: EcucIntegerParamDef
min = 0
max = 255
withAuto = true
symbolicNameValue = true
+parameter
EthCtrlConfigIngressFifoPriorityAssignment: EcucIntegerParamDef
min = 0
max = 7
lowerMultiplicity = 0
upperMultiplicity = *
+parameter
EthCtrlConfigIngressFifoBufTotal: EcucIntegerParamDef
min = 0
max = 65535
+parameter
EthCtrlConfigIngressFifoBufLenByte: EcucIntegerParamDef
min = 0
max = 65535

Figure 10.7: Overview EthCtrlConfigIngress configuration - OBSOLETE

10.2.7.2 EthCtrlConfigIngressQueue - DRAFT

The following parameter are introduced in `EthCtrlConfigIngressQueue` `EthCtrlConfigIngressQueueOverwriteEnabled` `EthCtrlEnableIngressQueueInterrupt`

SWS Item	**[ECUC_Eth_00095]**		
Container Name	EthCtrlConfigIngressQueue		
Parent Container	EthCtrlConfigIngress		
Description	Represents a queue at the ingress side. **Tags:** atp.Status=draft		
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			

SWS Item	**[ECUC_Eth_00099]**	
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00098]**		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00134]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00096]**
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	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local withAuto = true		

SWS Item	**[ECUC_Eth_00133]**		
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 repetion 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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00129]**		
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	–	
Scope / Dependency	scope: local dependency: If EthCtrlEnableIngressQueueInterrupt is set to TRUE, then EthCtrl EnableRxInterrupt has to be set to FALSE.		

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

SWS Item	**[ECUC_Eth_00131]**		
Container Name	EthCtrlConfigIngressQueueSortingType		
Parent Container	EthCtrlConfigIngressQueue		
Description	Defines one out of 4 possible sorting criteria for this ingress queue. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

SWS Item	**[ECUC_Eth_00128]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00127]**
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00126]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00125]**		
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	–	

▽

△

Scope / Dependency	scope: local

No Included Containers

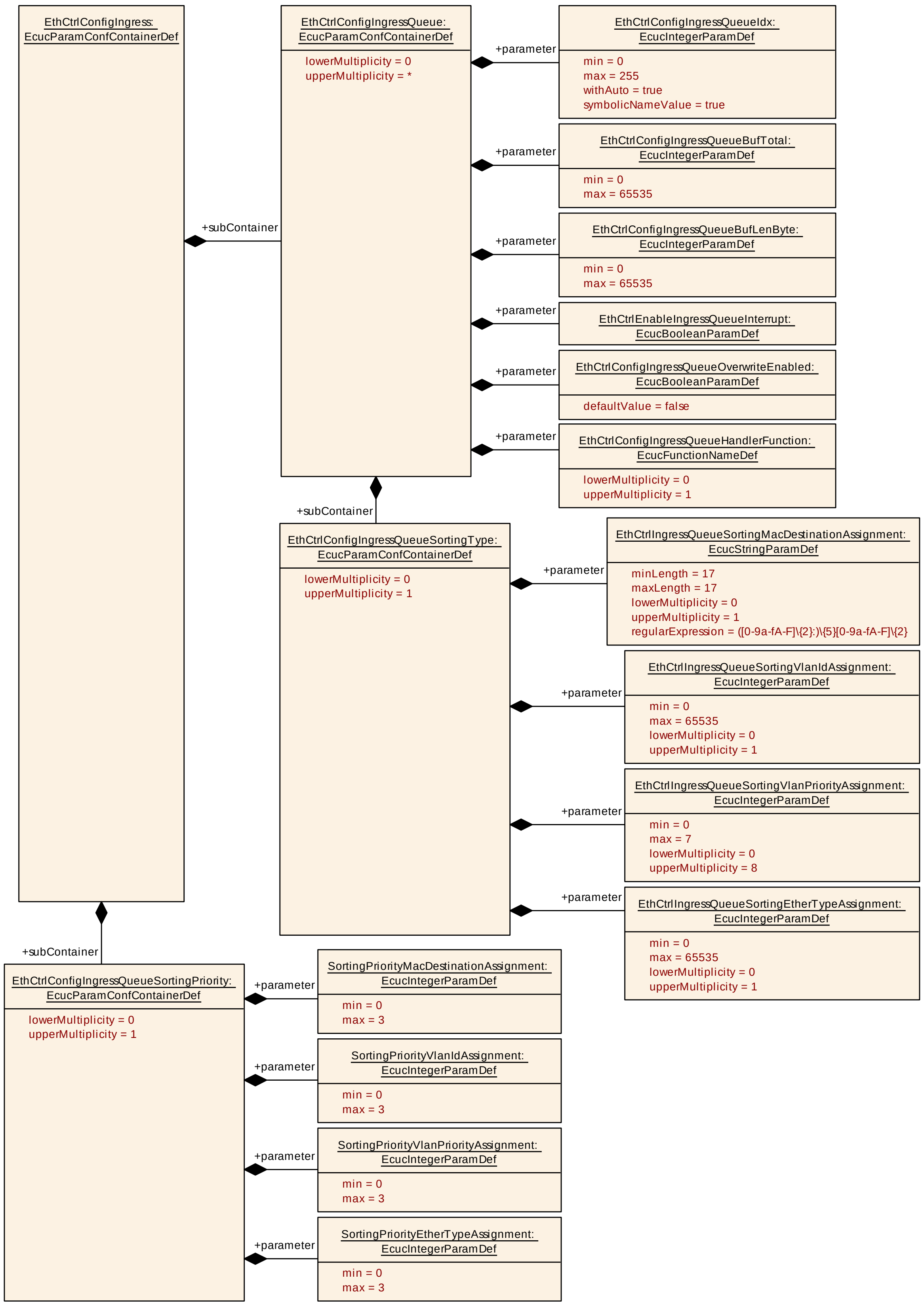

EthCtrlConfigIngress: EcucParamConfContainerDef
+subContainer
EthCtrlConfigIngressQueue: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+parameter
EthCtrlConfigIngressQueueIdx: EcucIntegerParamDef
min = 0
max = 255
withAuto = true
symbolicNameValue = true
+parameter
EthCtrlConfigIngressQueueBufTotal: EcucIntegerParamDef
min = 0
max = 65535
+parameter
EthCtrlConfigIngressQueueBufLenByte: EcucIntegerParamDef
min = 0
max = 65535
+parameter
EthCtrlEnableIngressQueueInterrupt: EcucBooleanParamDef
+parameter
EthCtrlConfigIngressQueueOverwriteEnabled: EcucBooleanParamDef
defaultValue = false
+parameter
EthCtrlConfigIngressQueueHandlerFunction: EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
EthCtrlConfigIngressQueueSortingType: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthCtrlIngressQueueSortingMacDestinationAssignment: EcucStringParamDef
minLength = 17
maxLength = 17
lowerMultiplicity = 0
upperMultiplicity = 1
regularExpression = ([0-9a-fA-F]\{2}:)\{5}[0-9a-fA-F]\{2}
+parameter
EthCtrlIngressQueueSortingVlanIdAssignment: EcucIntegerParamDef
min = 0
max = 65535
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthCtrlIngressQueueSortingVlanPriorityAssignment: EcucIntegerParamDef
min = 0
max = 7
lowerMultiplicity = 0
upperMultiplicity = 8
+parameter
EthCtrlIngressQueueSortingEtherTypeAssignment: EcucIntegerParamDef
min = 0
max = 65535
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
EthCtrlConfigIngressQueueSortingPriority: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
SortingPriorityMacDestinationAssignment: EcucIntegerParamDef
min = 0
max = 3
+parameter
SortingPriorityVlanIdAssignment: EcucIntegerParamDef
min = 0
max = 3
+parameter
SortingPriorityVlanPriorityAssignment: EcucIntegerParamDef
min = 0
max = 3
+parameter
SortingPriorityEtherTypeAssignment: EcucIntegerParamDef
min = 0
max = 3

Figure 10.8: Overview EthCtrlConfigIngress configuration - DRAFT

10.2.8 EthCtrlConfigSpiConfiguration

SWS Item	**[ECUC_Eth_00074]**		
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		
Post-Build Variant Multiplicity	false		
Configuration Parameters			

SWS Item	**[ECUC_Eth_00079]**		
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
Scope / Dependency	scope: local dependency: This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE.		

SWS Item	**[ECUC_Eth_00075]**		
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

▽

△

Scope / Dependency	scope: local 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.		

SWS Item	**[ECUC_Eth_00076]**		
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
Scope / Dependency	scope: local 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.		

SWS Item	**[ECUC_Eth_00081]**		
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 [26]. FALSE: Control data read/write protection is disabled (unprotected). TRUE: Control data read/write rotection 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	–	
Scope / Dependency	scope: local dependency: This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

SWS Item	**[ECUC_Eth_00085]**
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
Scope / Dependency	scope: local dependency: This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

SWS Item	**[ECUC_Eth_00082]**		
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
Scope / Dependency	scope: local dependency: This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

SWS Item	**[ECUC_Eth_00084]**		
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
Scope / Dependency	scope: local dependency: This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

SWS Item	**[ECUC_Eth_00080]**		
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
Scope / Dependency	scope: local dependency: This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

SWS Item	**[ECUC_Eth_00086]**		
Parameter Name	EthCtrlConfigSpiEnableTxChecksum		
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	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

▽

△

Scope / Dependency	scope: local dependency: This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

SWS Item	**[ECUC_Eth_00089]**		
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	–	
Scope / Dependency	scope: local dependency: This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

SWS Item	**[ECUC_Eth_00087]**		
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
Scope / Dependency	scope: local dependency: This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

SWS Item	**[ECUC_Eth_00083]**		
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
Scope / Dependency	scope: local dependency: This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE.		

Included Containers		
Container Name	**Multiplicity**	**Scope / 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 hardwares 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

SWS Item	**[ECUC_Eth_00077]**
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 hardwares 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
Configuration Parameters	

SWS Item	**[ECUC_Eth_00078]**
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
Scope / Dependency	scope: local dependency: This parameter is valid, if EthCtrlEnableSpiInterface is configured and set to TRUE		

SWS Item	**[ECUC_Eth_00088]**		
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	–	
Scope / Dependency	scope: local dependency: SpiSequence		

No Included Containers

10.2.9 EthDemEventParameterRefs

SWS Item	**[ECUC_Eth_00016]**
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.
Configuration Parameters	

SWS Item	[ECUC_Eth_00017]		
Parameter Name	ETH_E_ACCESS		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Controller access failed" has occured.		
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
Scope / Dependency	scope: local		

SWS Item	[ECUC_Eth_00026]		
Parameter Name	ETH_E_ALIGNMENT		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Alignment Error" has occured.		
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
Scope / Dependency	scope: local		

SWS Item	[ECUC_Eth_00023]		
Parameter Name	ETH_E_CRC		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "CRC Failure" has occured.		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00029]**		
Parameter Name	ETH_E_LATECOLLISION		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Late Collisions" has occured.		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00028]**		
Parameter Name	ETH_E_MULTIPLECOLLISION		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Multiple Collisions" has occured.		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00025]**		
Parameter Name	ETH_E_OVERSIZEFRAME		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Oversized Frame" has occured.		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00021]**		
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 occured.		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00027]**		
Parameter Name	ETH_E_SINGLECOLLISION		
Parent Container	EthDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Single Collisions" has occured.		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00024]**
Parameter Name	ETH_E_UNDERSIZEFRAME
Parent Container	EthDemEventParameterRefs
Description	Reference to the DemEventParameter which shall be issued when the error "Undersized Frame" has occured.
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
Scope / Dependency	scope: local		

No Included Containers

10.2.10 EthGeneral

SWS Item	**[ECUC_Eth_00001]**
Container Name	EthGeneral
Parent Container	Eth
Description	General configuration of Ethernet Driver module
Configuration Parameters	

SWS Item	**[ECUC_Eth_00003]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00035]**		
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	–	

▽

△

Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00072]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00061]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00060]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00037]**
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00018]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00022]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00002]**		
Parameter Name	EthMaxCtrlsSupported		
Parent Container	EthGeneral		
Description	Limits the total number of supported controllers.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants

▽

△

	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

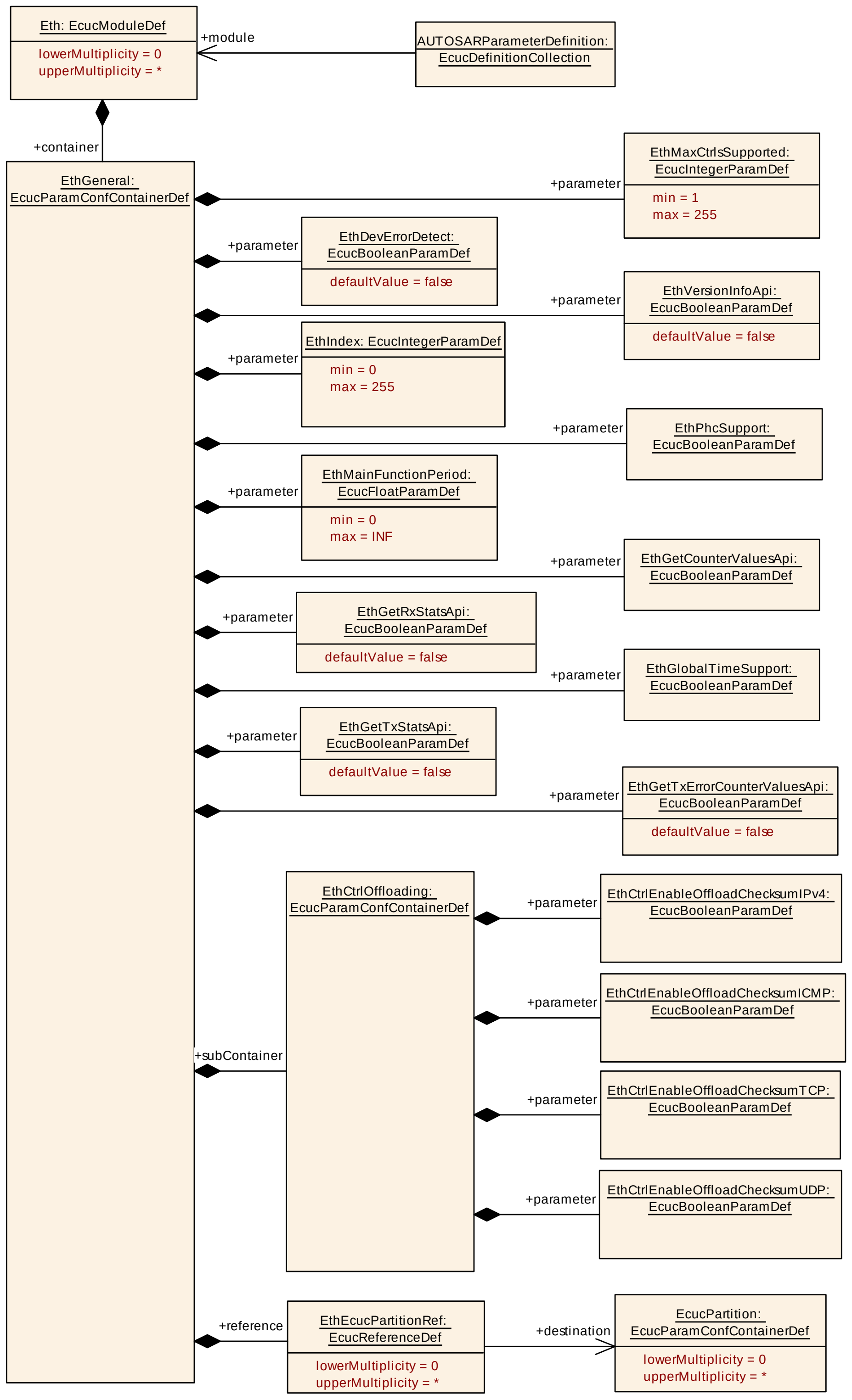
SWS Item	**[ECUC_Eth_00107]**		
Parameter Name	EthPhcSupport		
Parent Container	EthGeneral		
Description	Enables/Disables the PTP HW Clock (PHC). **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00004]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00064]**		
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. The Ethernet driver will operate as an independent instance in each of the partitions.		
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	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthCtrlOffloading	1	Configuration of hardware offloading features.

[SWS_Eth_00259] ⌈The module will operate as an independent instance in each of the partitions, means the called API will only target the partition it is called in.⌋*()*


Eth: EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = *
+module
AUTOSARParameterDefinition: EcucDefinitionCollection
+container
EthGeneral: EcucParamConfContainerDef
+parameter
EthMaxCtrlsSupported: EcucIntegerParamDef
min = 1
max = 255
EthDevErrorDetect: EcucBooleanParamDef
defaultValue = false
EthVersionInfoApi: EcucBooleanParamDef
defaultValue = false
EthIndex: EcucIntegerParamDef
min = 0
max = 255
EthPhcSupport: EcucBooleanParamDef
EthMainFunctionPeriod: EcucFloatParamDef
min = 0
max = INF
EthGetCounterValuesApi: EcucBooleanParamDef
EthGetRxStatsApi: EcucBooleanParamDef
defaultValue = false
EthGlobalTimeSupport: EcucBooleanParamDef
EthGetTxStatsApi: EcucBooleanParamDef
defaultValue = false
EthGetTxErrorCounterValuesApi: EcucBooleanParamDef
defaultValue = false
EthCtrlOffloading: EcucParamConfContainerDef
EthCtrlEnableOffloadChecksumIPv4: EcucBooleanParamDef
EthCtrlEnableOffloadChecksumICMP: EcucBooleanParamDef
EthCtrlEnableOffloadChecksumTCP: EcucBooleanParamDef
EthCtrlEnableOffloadChecksumUDP: EcucBooleanParamDef
+subContainer
+reference
EthEcucPartitionRef: EcucReferenceDef
lowerMultiplicity = 0
upperMultiplicity = *
+destination
EcucPartition: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
Figure 10.9: Overview Eth general configuration

10.2.10.1 EthCtrlOffloading

SWS Item	**[ECUC_Eth_00030]**
Container Name	EthCtrlOffloading
Parent Container	EthGeneral
Description	Configuration of hardware offloading features.
Configuration Parameters	

SWS Item	**[ECUC_Eth_00032]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00031]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00033]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Eth_00034]**		
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	–	
Scope / Dependency	scope: local		

No Included Containers

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in SWS_BSWGeneral [3].

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 R23-11

B.1.1 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.1.2 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.1.3 Deleted Specification Items in R23-11

none

B.1.4 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.1.5 Changed Constraints in R23-11

none

B.1.6 Deleted Constraints in R23-11

none