

<b>Document Title</b>	Specification of Ethernet Driver
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	430
<b>Document Classification</b>	Standard

<b>Document Status</b>	Final
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	4.3.0

Document Change History			
Date	Release	Changed by	Change Description
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Quality of Service (QoS) support</li> <li>Ethernet statistics counter access</li> </ul>
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Eth_ControllerInit functionality merged into Eth_Init API</li> <li>Development Error Tracer renamed to Default Error Tracer</li> <li>IRQ handler API removed</li> </ul>
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Change from Synchronous to Asynchronous API</li> <li>gPTP Timestamp Support</li> <li>Enhanced Production Errors</li> <li>Changed Access to Statistic Frame Handling Registers</li> </ul>
2014-03-31	4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Introduction of periodic call to Eth_SetControllerMode</li> <li>Support of VLANs (Virtual Local Area Networks)</li> <li>Editorial changes</li> </ul>
2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Introduction of Eth_GeneralTypes.h</li> <li>Support of API deviation for asynchronous implementation</li> <li>Changes in API of EthIf_ProvideTxBuffer and EthIf_SetPhysAddr</li> <li>Editorial changes</li> <li>Removed chapter(s) on change documentation</li> </ul>

Document Change History			
Date	Release	Changed by	Change Description
2013-03-15	4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"><li>Configurable MAC address based filtering</li><li>Detection of lost Ethernet frames</li><li>Buffer handling enhancement</li></ul>
2011-12-22	4.0.3	AUTOSAR Administration	<ul style="list-style-type: none"><li>Description of buffer behaviour in Eth_SetControllerMode extended.</li></ul>
2010-09-30	3.1.5	AUTOSAR Administration	<ul style="list-style-type: none"><li>Enhanced development error detection for active controller before controller access</li><li>Further post-build configurable parameters</li><li>Improved description of 'XxxCtrlIdx' semantics</li><li>'Instance ID' removed from Version Info (concerns Eth_GetVersionInfo API)</li><li>Additional development error in Eth_GetVersionInfo API</li></ul>
2010-02-02	3.1.4	AUTOSAR Administration	<ul style="list-style-type: none"><li>Initial Release</li></ul>

## Disclaimer

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

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

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

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

## Table of Contents

1	Introduction and functional overview .....	8
2	Acronyms and abbreviations .....	10
3	Related documentation.....	11
3.1	Input documents.....	11
3.2	Related standards and norms .....	12
3.3	Related specification .....	12
4	Constraints and assumptions .....	13
4.1	Limitations .....	13
4.2	Applicability to car domains .....	13
5	Dependencies to other modules.....	14
5.1	File structure.....	14
5.1.1	Header file structure.....	14
6	Requirements traceability .....	15
7	Functional specification .....	16
7.1	Ethernet BSW stack .....	16
7.1.1	Indexing scheme.....	17
7.1.2	Requirements.....	18
7.1.3	Configuration description .....	20
7.2	Error classification .....	20
7.2.1	Default Errors.....	20
7.2.2	Runtime Errors.....	21
7.2.3	Transient Faults .....	21
7.2.4	Production Errors .....	21
7.2.5	Extended Production Errors.....	21
8	API specification.....	25
8.1	Imported types.....	25
8.2	Type definitions .....	25
8.2.1	Eth_ConfigType .....	25
8.2.2	Eth_ModeType.....	26
8.2.3	Eth_StateType .....	26
8.2.4	Eth_FrameType .....	26
8.2.5	Eth_DataType .....	26
8.2.6	Eth_BufIdxType .....	26
8.2.7	Eth_RxStatusType.....	27
8.2.8	Eth_FilterActionType.....	27
8.2.9	Eth_TimeStampQualType.....	27
8.2.10	Eth_TimeStampType .....	27
8.2.11	Eth_TimeIntDiffType .....	28
8.2.12	Eth_RateRatioType .....	28
8.2.13	Eth_MacVlanType .....	28
8.2.14	Eth_CounterType .....	29

8.2.15	Eth_RxStatsType.....	30
8.2.16	Eth_TxStatsType .....	32
8.2.17	Eth_TxErrorCounterValuesType.....	33
8.3	Function definitions.....	34
8.3.1	Eth_Init.....	34
8.3.2	Eth_SetControllerMode.....	35
8.3.3	Eth_GetControllerMode .....	36
8.3.4	Eth_GetPhysAddr .....	37
8.3.5	Eth_SetPhysAddr.....	37
8.3.6	Eth_UpdatePhysAddrFilter .....	38
8.3.7	Eth_WriteMii.....	40
8.3.8	Eth_ReadMii .....	40
8.3.9	Eth_GetCounterValues .....	41
8.3.10	Eth_GetRxStats .....	42
8.3.11	Eth_GetTxStats .....	44
8.3.12	Eth_GetTxErrorCounterValues.....	44
8.3.13	Eth_GetCurrentTime.....	45
8.3.14	Eth_EnableEgressTimeStamp.....	46
8.3.15	Eth_GetEgressTimeStamp .....	47
8.3.16	Eth_GetIngressTimeStamp .....	48
8.3.17	Eth_ProvideTxBuffer.....	49
8.3.18	Eth_Transmit .....	50
8.3.19	Eth_Receive .....	52
8.3.20	Eth_TxConfirmation .....	52
8.3.21	Eth_GetVersionInfo .....	53
8.4	Callback notifications.....	54
8.5	Scheduled functions .....	54
8.5.1	Eth_MainFunction .....	54
8.6	Expected Interfaces.....	54
8.6.1	Mandatory Interfaces .....	55
8.6.2	Optional Interfaces.....	55
8.6.3	Configurable interfaces .....	55
9	Sequence diagrams .....	57
10	Configuration specification.....	58
10.1	Containers and configuration parameters .....	59
10.1.1	Eth .....	63
10.1.2	EthConfigSet .....	63
10.1.3	EthCtrlConfig .....	63
10.1.4	EthCtrlConfigEgress .....	65
10.1.5	EthCtrlConfigEgressFifo .....	66
10.1.6	EthCtrlConfigScheduler .....	67
10.1.7	EthCtrlConfigSchedulerPredecessor .....	67
10.1.8	EthCtrlConfigShaper.....	68
10.1.9	EthCtrlConfigIngress.....	69
10.1.10	EthCtrlConfigIngressFifo.....	69
10.1.11	EthDemEventParameterRefs .....	70
10.1.12	EthGeneral .....	73
10.1.13	EthCtrlOffloading .....	76

11	Not applicable requirements .....	78
----	-----------------------------------	----

### **Known Limitations**

Currently, chapter 5 Dependencies to other modules 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*.

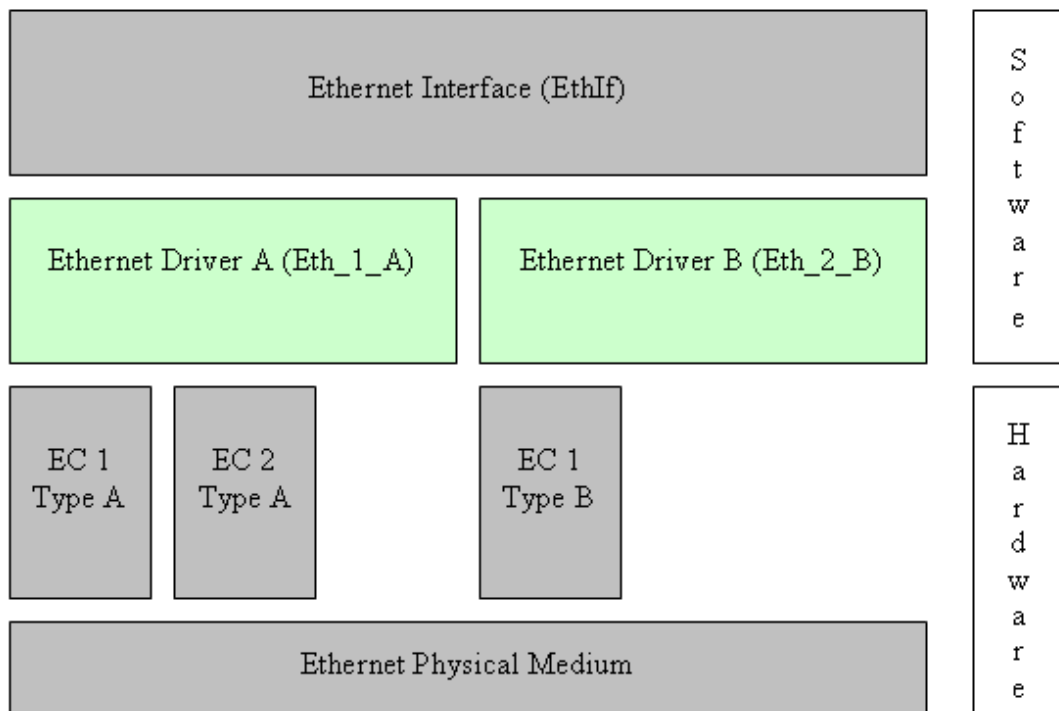
This indicates the main task of the Ethernet Driver:

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 and abbreviations

<b>Abbreviation / Acronym:</b>	<b>Description:</b>
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
MCG	Module Configuration Generator
MII	Media Independent Interface (standardized Interface provided by Ethernet controllers to access Ethernet transceivers)
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

## 3 Related documentation

### 3.1 Input documents

- [1] List of Basic Software Modules  
AUTOSAR\_TR\_BSWModuleList.pdf
- [2] Layered Software Architecture  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf
- [3] AUTOSAR General Requirements on Basic Software Modules  
AUTOSAR\_SRS\_BSWGeneral.pdf
- [4] Specification of Communication  
AUTOSAR\_SWS\_COM.pdf
- [5] Requirements on Ethernet Support in AUTOSAR  
AUTOSAR\_SRS\_Ethernet.pdf
- [6] Specification of Ethernet Interface  
AUTOSAR\_SWS\_EthernetInterface.pdf
- [7] Specification of Ethernet State Manager  
AUTOSAR\_SWS\_EthernetStateManager.pdf
- [8] Specification of Ethernet Transceiver Driver  
AUTOSAR\_SWS\_EthernetTransceiver.pdf
- [9] Specification of Socket Adapter  
AUTOSAR\_SWS\_SocketAdapter.pdf
- [10] Specification of UDP Network Management  
AUTOSAR\_SWS\_UDPNetworkManagement.pdf
- [11] Specification of PDU Router  
AUTOSAR\_SWS\_PDURouter.pdf
- [12] BSW Scheduler Specification  
AUTOSAR\_SWS\_Scheduler.pdf
- [13] Specification of ECU Configuration  
AUTOSAR\_TPS\_ECUConfiguration.pdf
- [14] Specification of Memory Mapping  
AUTOSAR\_SWS\_MemoryMapping.pdf
- [15] Specification of Standard Types  
AUTOSAR\_SWS\_StandardTypes.pdf

- [16] Specification of Default Error Tracer  
AUTOSAR\_SWS\_DefaultErrorTracer.pdf
- [17] Specification of Diagnostics Event Manager  
AUTOSAR\_SWS\_DiagnosticEventManager
- [18] Specification of ECU State Manager  
AUTOSAR\_SWS\_ECUSTateManager.pdf
- [19] General Specification of Basic Software Modules  
AUTOSAR\_SWS\_BSWGeneral.pdf

## 3.2 Related standards and norms

- [20] IEEE 802.3-2006
- [21] IEC 7498-1 The Basic Model, IEC Norm, 1994
- [22] IETF RFC 2819
- [23] IEEE Standard 802.1AS™- 30 of March 2011  
<http://standards.ieee.org/getieee802/download/802.1AS-2011.pdf>

## 3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [19] (SWS BSW General), 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**

The Ethernet Driver module is only able to handle a single thread of execution. The execution must not be pre-empted by itself.

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)
- Ethernet Transceiver Driver (EthTrcv)

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 File structure

#### 5.1.1 Header file structure

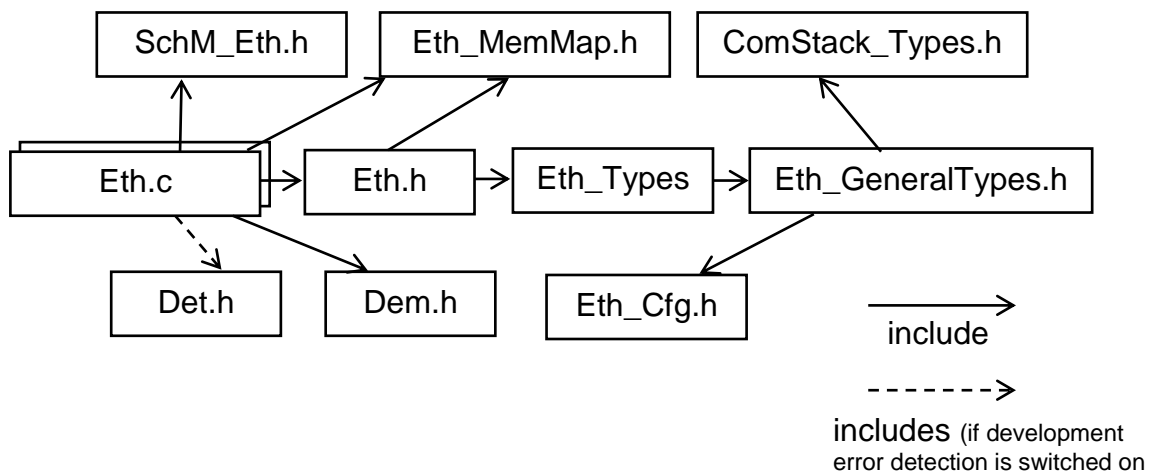


Figure 5.1 Ethernet Driver file structure

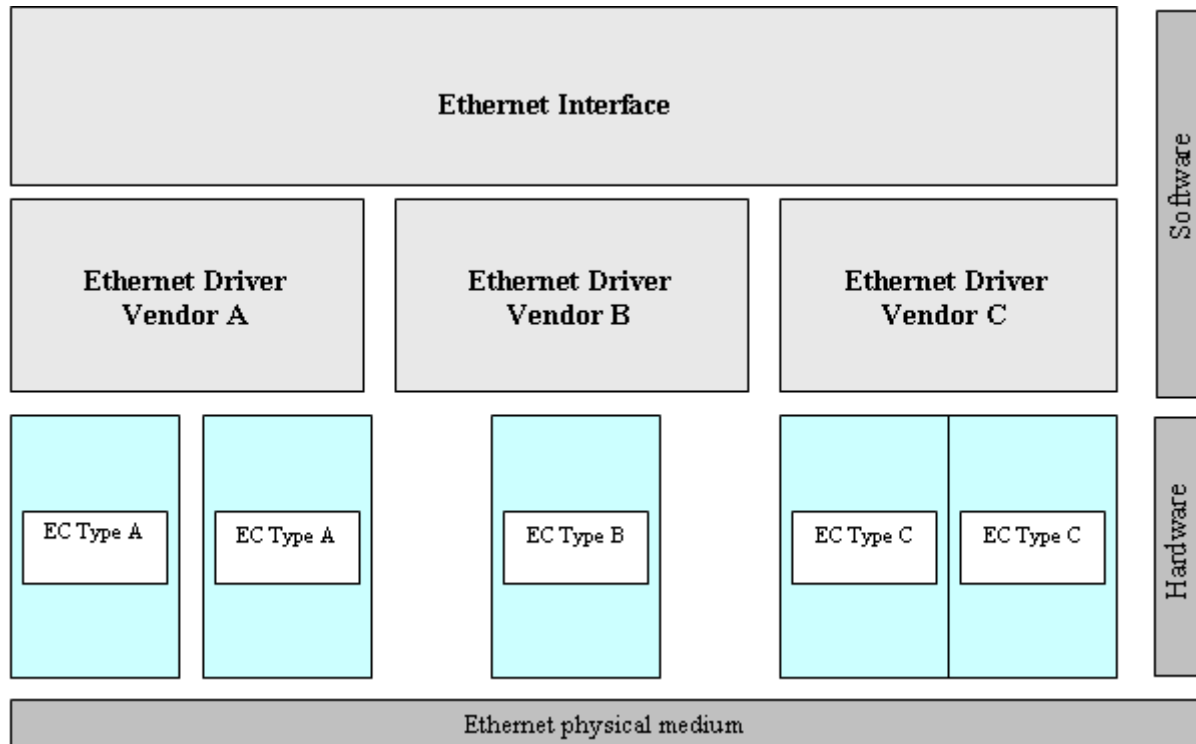
## 6 Requirements traceability

Requirement	Description	Satisfied by
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
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
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
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_Eth_00248, SWS_Eth_00252
SRS_Eth_00053	SWS shall specify configuration	SWS_Eth_00251, SWS_Eth_00255
SRS_ETH_00086	-	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

## 7 Functional specification

### 7.1 Ethernet BSW stack

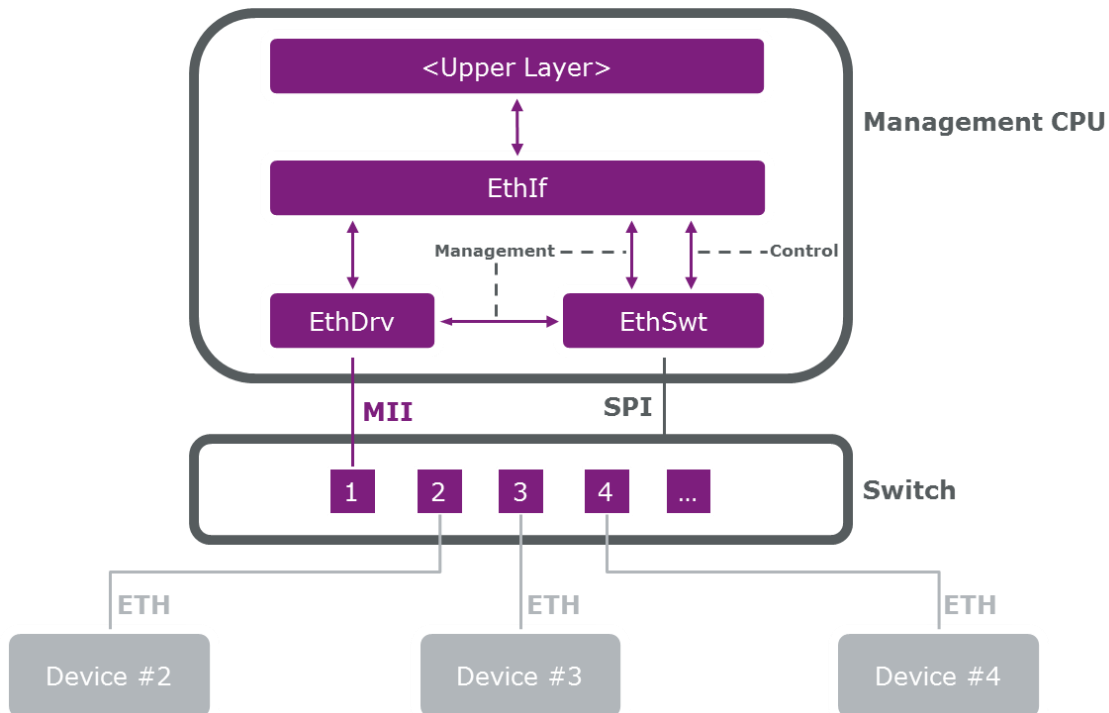
As part of the AUTOSAR Layered Software Architecture according to Figure 7.1, the Ethernet BSW modules also form a layered software stack. Figure 7.1 depicts the basic structure of this Ethernet BSW stack. The Ethernet Interface module accesses several controllers using the Ethernet Driver layer, which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



**Figure 7.1: Basic Structure of the Ethernet BSW stack**

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.

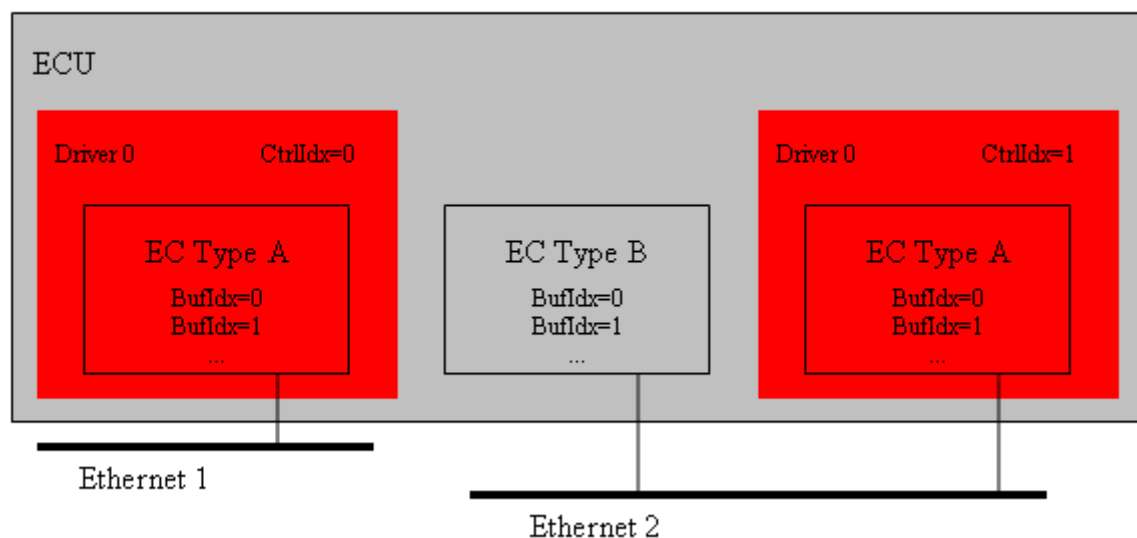




**Figure 7.2: HW/SW basic structure including Switch device**

### 7.1.1 Indexing scheme

Users of the Ethernet Driver identify controller resources using an indexing scheme as depicted in Figure 7.3.



**Figure 7.3: 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 `Eth_CtrlIdx` within configuration corresponds to parameter `CtrlIdx` used in the API. `]()`

[SWS\_Eth\_00004] `[`

A buffer index (`BuIdx`) 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, BuIdx>` only. A `BuIdx` uniquely identifies the buffer used for an Ethernet Driver. `]()`

### 7.1.2 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 default 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 [16].

[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.  
J()

[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. J()

[SWS\_Eth\_00176] [

The Global Time interfaces shall be used to access the time synchronization functionalities (see document [23]). J()

[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). J ()

[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.

[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 SWC\_ETH\_0096) J()

[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).  
J()

### 7.1.3 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. J()

[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. J()

[SWS\_Eth\_00126] [

The MCG shall ensure the consistency of the generated configuration data. J()

[SWS\_Eth\_00013] J]

The configuration of the Ethernet Driver module shall be calculated at ECU configuration time. None of the communication parameters shall be calculated at runtime. J()

[SWS\_Eth\_00014] [

The start address of post-build time configuration data shall be passed during module initialization (see chapter 8.3.1). J()

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

### 7.2.1 Default Errors

[SWS\_Eth\_00016] [

<i>Type or error</i>	<i>Relevance</i>	<i>Related error code</i>	<i>Value [hex]</i>
Invalid controller index	Default error	ETH_E_INV_CTRL_IDX	0x01
Eth module or controller was not	Default error	ETH_E_NOT_INITIALIZED	0x02

initialized			
Invalid pointer in parameter list	Default error	ETH_E_PARAM_POINTER	0x03
Invalid parameter	Default error	ETH_E_INV_PARAM	0x04
Invalid mode	Default error	ETH_E_INV_MODE	0x05

()

## 7.2.2 Runtime Errors

There are no runtime errors.

## 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] [

<b>Error Name:</b>	ETH_E_ACCESS	
<b>Short Description:</b>	Ethernet Controller Access Failure.	
<b>Long Description:</b>	Monitors the access to the Ethernet Controller.	
<b>Detection Criteria:</b>	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 succeeds the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
<b>Secondary Parameters:</b>	None.	
<b>Time Required:</b>	None.	
<b>Monitor Frequency</b>	None.	

()

[SWS\_Eth\_00174] [

<b>Error Name:</b>	ETH_E_RX_FRAMES_LOST	
<b>Short Description:</b>	Ethernet Frames Lost.	
<b>Long Description:</b>	Monitors the loss of Ethernet frames during reception.	
<b>Detection Criteria:</b>	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.
<b>Secondary Parameters:</b>	None.	
<b>Time Required:</b>	None.	
<b>Monitor Frequency</b>	None.	

()

[SWS\_Eth\_00219] [

<b>Error Name:</b>	ETH_E_CRC	
<b>Short Description:</b>	CRC Failure	
<b>Long Description:</b>	Monitors invalid Ethernet frames during reception.	
<b>Detection Criteria:</b>	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.
<b>Secondary Parameters:</b>	None.	
<b>Time Required:</b>	None.	
<b>Monitor Frequency</b>	None.	

()

[SWS\_Eth\_00220] [

<b>Error Name:</b>	ETH_E_UNDERSIZEFRAME	
<b>Short Description:</b>	Frame Size Underflow	
<b>Long Description:</b>	Monitors undersize Ethernet frames during reception.	
<b>Detection Criteria:</b>	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.
<b>Secondary Parameters:</b>	None.	
<b>Time Required:</b>	None.	
<b>Monitor Frequency</b>	None.	

()

[SWS\_Eth\_00221] [

<b>Error Name:</b>	ETH_E_OVERSIZEFRAME	
<b>Short Description:</b>	Frame Size Overflow	
<b>Long Description:</b>	Monitors oversize Ethernet frames during reception.	
<b>Detection Criteria:</b>	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.
<b>Secondary Parameters:</b>	None.	
<b>Time Required:</b>	None.	
<b>Monitor Frequency</b>	None.	

()

[SWS\_Eth\_00222] [

<b>Error Name:</b>	ETH_E_ALIGNMENT	
<b>Short Description:</b>	Frame Alignment Error	
<b>Long Description:</b>	Monitors alignment errors.	
<b>Detection Criteria:</b>	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.
<b>Secondary Parameters:</b>	None.	
<b>Time Required:</b>	None.	
<b>Monitor Frequency</b>	None.	

&gt;()

[SWS\_Eth\_00223] [

<b>Error Name:</b>	ETH_E_SINGLECOLLISION	
<b>Short Description:</b>	Single Frame Collision	
<b>Long Description:</b>	Monitors Ethernet single frame collision.	
<b>Detection Criteria:</b>	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.
<b>Secondary Parameters:</b>	None.	
<b>Time Required:</b>	None.	
<b>Monitor Frequency</b>	None.	

&gt;()

[SWS\_Eth\_00224] [

<b>Error Name:</b>	ETH_E_MULTIPLECOLLISION	
<b>Short Description:</b>	Multiple Frame Collision	
<b>Long Description:</b>	Monitors Ethernet multiple frame collision.	
<b>Detection Criteria:</b>	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.
<b>Secondary Parameters:</b>	None.	
<b>Time Required:</b>	None.	
<b>Monitor Frequency</b>	None.	

&gt;()

[SWS\_Eth\_00225] [

<b>Error Name:</b>	ETH_E_LATECOLLISION	
<b>Short Description:</b>	Late Frame Collision	
<b>Long Description:</b>	Monitors Ethernet late frame collision.	
<b>Detection Criteria:</b>	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.

<b>Secondary Parameters:</b>	None.
<b>Time Required:</b>	None.
<b>Monitor Frequency</b>	None.

l()



## 8 API specification

### 8.1 Imported types

This chapter lists all types included from the following files:

[SWS\_Eth\_00026] [

Module	Imported Type
ComStack_Types	BufReq_ReturnType
Dem	Dem_EventIdType
	Dem_EventStatusType
Eth_GeneralTypes	Eth_BufIdxType
	Eth_ConfigType
	Eth_CounterType
	Eth_DataType
	Eth_FilterActionType
	Eth_FrameType
	Eth_ModeType
	Eth_RateRatioType
	Eth_RxStatsType
	Eth_RxStatusType
	Eth_TimeIntDiffType
	Eth_TimeStampQualType
	Eth_TimeStampType
	Eth_TxErrorCounterValuesType
	Eth_TxStatsType
Std_Types	Std_ReturnType
	Std_VersionInfoType

] (SRS\_Eth\_00127)

### 8.2 Type definitions

[SWS\_Eth\_00148] [

Eth.h shall include Eth\_GeneralTypes.h for the include of general Eth type declarations. ]()

[SWS\_Eth\_00149] [

The types specified in SWS\_EthernetDriver shall be declared in Eth\_GeneralTypes.h. ]()

#### 8.2.1 Eth\_ConfigType

[SWS\_Eth\_00156] [

<b>Name:</b>	Eth_ConfigType
<b>Type:</b>	Structure
<b>Range:</b>	Implementation specific.
<b>Description:</b>	Implementation specific structure of the post build configuration

] ()

### 8.2.2 Eth\_ModeType

[SWS\_Eth\_00158] [

<b>Name:</b>	Eth_ModeType		
<b>Type:</b>	Enumeration		
<b>Range:</b>	ETH_MODE_DOWN	0x00	Controller disabled
	ETH_MODE_ACTIVE	0x01	Controller enabled
<b>Description:</b>	This type defines the controller modes		

] ()

### 8.2.3 Eth\_StateType

[SWS\_Eth\_00159] [

<b>Name:</b>	Eth_StateType		
<b>Type:</b>	Enumeration		
<b>Range:</b>	ETH_STATE_UNINIT	0x00	Driver is not yet configured
	ETH_STATE_INIT	0x01	Driver is configured
<b>Description:</b>	Status supervision used for Development Error Detection. The state shall be available for debugging.		

] ()

### 8.2.4 Eth\_FrameType

[SWS\_Eth\_00160] [

<b>Name:</b>	Eth_FrameType		
<b>Type:</b>	--		
<b>Range:</b>	uint16	0x0000 - 0xFFFF	See [21]
<b>Description:</b>	This type defines the Ethernet frame type used in the Ethernet frame header		

] ()

### 8.2.5 Eth\_DataType

[SWS\_Eth\_00161] [

<b>Name:</b>	Eth_DataType		
<b>Type:</b>	--		
<b>Range:</b>	uint8	0x00 - 0xFF	8, 16 or 32 bit CPU
	uint16	0x0000 - 0xFFFF	8 or 16 bit CPU
	uint32	0x00000000 - 0xFFFFFFFF	32 bit CPU
<b>Description:</b>	This type defines the Ethernet data type used for data transmission. Its definition depends on the used CPU.		

] ()

### 8.2.6 Eth\_BufIdxType

[SWS\_Eth\_00175] [

<b>Name:</b>	Eth_BufIdxType		
<b>Type:</b>	uint32		
<b>Description:</b>	Ethernet buffer identifier type.		

] ()

## 8.2.7 Eth\_RxStatusType

[SWS\_Eth\_00162] [

<b>Name:</b>	Eth_RxStatusType		
<b>Type:</b>	Enumeration		
<b>Range:</b>	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
<b>Description:</b>	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.		

] ()

## 8.2.8 Eth\_FilterActionType

[SWS\_Eth\_00163] [

<b>Name:</b>	Eth_FilterActionType		
<b>Type:</b>	Enumeration		
<b>Range:</b>	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
<b>Description:</b>	The Enumeration Type Eth_FilterActionType describes the action to be taken for the MAC address given in *PhysAddrPtr.		

] ()

## 8.2.9 Eth\_TimeStampQualType

[SWS\_Eth\_00177] [

<b>Name:</b>	Eth_TimeStampQualType		
<b>Type:</b>	--		
<b>Range:</b>	ETH_VALID	0	--
	ETH_INVALID	1	--
	ETH_UNCERTAIN	2	--
<b>Description:</b>	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.		

] ()

## 8.2.10 Eth\_TimeStampType

[SWS\_Eth\_00178] [

<b>Name:</b>	Eth_TimeStampType		
<b>Type:</b>	Structure		
<b>Element:</b>	uint32	nanoseconds	Nanoseconds part of the time
	uint32	seconds	32 bit LSB of the 48 bits Seconds part of

			the time
	uint16	secondsHi	16 bit MSB of the 48 bits Seconds part of the time
<b>Description:</b>	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		

] ()

### 8.2.11 Eth\_TimeIntDiffType

[SWS\_Eth\_00179] [

<b>Name:</b>	Eth TimeIntDiffType		
<b>Type:</b>	Structure		
<b>Element:</b>	Eth_TimeStampType	diff	time difference
	boolean	sign	Positive (True) / negative (False) time
<b>Description:</b>	Variables of this type are used to express time differences.		

] ()

### 8.2.12 Eth\_RateRatioType

[SWS\_Eth\_00180] [

<b>Name:</b>	Eth RateRatioType		
<b>Type:</b>	Structure		
<b>Element:</b>	Eth_TimeIntDiffType	IngressTimeStampDelta	IngressTimeStampSync2 - IngressTimeStampSync1
	Eth_TimeIntDiffType	OriginTimeStampDelta	OriginTimeStampSync2[FUP2] - OriginTimeStampSync1[FUP1]
<b>Description:</b>	Variables of this type are used to express frequency ratios.		

] ()

### 8.2.13 Eth\_MacVlanType

[SWS\_Eth\_91001] [

<b>Name:</b>	Eth_MacVlanType		
<b>Type:</b>	Structure		
<b>Element:</b>	uint8[6]	MacAddr	Specifies the MAC address [0..255,0..255,0..255,0..255,0..255,0..255]
	uint16	VlanId	Specifies the VLAN address 0..65535
	uint8	SwitchPort	Port of the switch 0..255
	uint16	HashValue	Hash value that is generated by the Ethernet switch according the given

			MacAddr. If the HashValue is not available the value shall be set to 0b1111xxxxxxxxxxx == 0xFxxx (invalid value). If the HashValue is available the value shall be set to 0b0000xxxxxxxxxxx == 0x0xxx (valid value)
<b>Description:</b>	The interpretation of this value is not specified, i.e. whether it is number of used bytes or number of used memory cells, etc.		

] (SRS\_ETH\_00086)

## 8.2.14 Eth\_CounterType

[SWS\_Eth\_91002] [

<b>Name:</b>	Eth_CounterType		
<b>Type:</b>	Structure		
<b>Element:</b>	uint32	DropPktBufOverrun	dropped packets due to buffer overrun
	uint32	DropPktCrc	dropped packets due to CRC errors
	uint32	UndersizePkt	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)
	uint32	OversizePkt	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)
	uint32	AlgnmtErr	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.
	uint32	SqeTestErr	SQE test error according to IETF RFC1643 dot3StatsSQETestErrors
	uint32	DiscInbdPkt	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)
	uint32	ErrInbdPkt	total number of erroneous inbound packets
	uint32	DiscOtbdPkt	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)
	uint32	ErrOtbdPkt	total number of erroneous outbound packets
	uint32	SnglCollPkt	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)
	uint32	MultCollPkt	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)
	uint32	DfrdPkt	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 dot3StatsDeferredTransmissions)
	uint32	LatCollPkt	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)
	uint32	HwDepCtr0	hardware dependent counter value
	uint32	HwDepCtr1	hardware dependent counter value
	uint32	HwDepCtr2	hardware dependent counter value
	uint32	HwDepCtr3	hardware dependent counter value
<b>Description:</b>		Statistic counter for diagnostics.	

] ()

## 8.2.15 Eth\_RxStatsType

[SWS\_Eth\_91002] [

<b>Name:</b>	Eth_RxStatsType		
<b>Type:</b>	Structure		
<b>Element:</b>	uint32	RxStatsDropEvents	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.
	uint32	RxStatsOctets	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.
	uint32	RxStatsPkts	The total number of packets (including bad packets, broadcast packets, and multicast packets) received. Also described in IETF RFC 2819 MIB etherStatsPkts
	uint32	RxStatsBroadcastPkts	The total number of good packets received that were directed to the broadcast address. Also described in IETF RFC 2819 MIB etherStatsBroadcastPkts
	uint32	RxStatsMulticastPkts	The total number of good packets received that were directed to a

			multicast address. Also described in IETF RFC 2819 MIB etherStatsMulticastPkts.
	uint32	RxStatsCrcAlignErrors	The total number of packets received that had a length of between 64 and 1518 octets that had either a bad Frame Check Sequence (FCS) with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error). Also described in IETF RFC 2819 MIB etherStatsCRCAlignErrors
	uint32	RxStatsUndersizePkts	The total number of packets received that were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. Also described in IETF RFC 2819 MIB etherStatsUndersizePkts.
	uint32	RxStatsOversizePkts	The total number of packets received that were longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed. Also described in IETF RFC 2819 MIB etherStatsOversizePkts
	uint32	RxStatsFragments	The total number of packets received that were less than 64 octets in length (excluding framing bits but including FCS octets) and had either a bad Frame Check Sequence (FCS) with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error). Also described in IETF RFC 2819 MIB etherStatsFragments.
	uint32	RxStatsJabbers	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.
	uint32	RxStatsCollisions	The best estimate of the total number of collisions on this Ethernet segment. Also described in IETF RFC 2819 MIB etherStatsCollisions
	uint32	RxStatsPkts64Octets	The total number of packets (including bad packets) received that were 64 octets in length. Also described in IETF RFC 2819 MIB etherStatsPkts64Octets
	uint32	RxStatsPkts65to127Octets	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
	uint32	RxStatsPkts128to255Octets	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
	uint32	RxStatsPkts256to511Octets	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
	uint32	RxStatsPkts512to1023Octets	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
	uint32	RxStatsPkts1024to1518Octets	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
	uint32	RxUnicastFrames	The number of subnetwork-unicast packets delivered to a higher-layer protocol. Also described in IETF RFC1213 MIB ifInUcastPkts
<b>Description:</b> Statistic counter for diagnostics.			

] (SRS\_Eth\_00127)

## 8.2.16 Eth\_TxStatsType

[SWS\_Eth\_91003] [

<b>Name:</b>	Eth_TxStatsType		
<b>Type:</b>	Structure		
<b>Element:</b>	uint32	TxNumberOfOctets	The total number of octets transmitted out of the interface, including framing characters. Also described in IETF RFC1213 MIB ifOutOctets.
	uint32	TxNUcastPkts	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
	uint32	TxUniCastPkts	The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or



			not sent. Also described in IETF RFC1213 MIB ifOutUcastPkts.
<b>Description:</b>	Statistic counter for diagnostics.		

] (SRS\_Eth\_00127)

## 8.2.17 Eth\_TxErrorCounterValuesType

[SWS\_Eth\_91004] [

<b>Name:</b>	Eth TxErrorCounterValuesType		
<b>Type:</b>	Structure		
<b>Element:</b>	uint32	TxDroppedNoErrorPkts	The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. Also described in IETF RFC1213 MIB ifOutDiscards
	uint32	TxDroppedErrorPkts	transmitted because of errors. Also described in IETF RFC1213 MIB ifOutErrors
	uint32	TxDeferredTrans	A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy. The count represented by an instance of this object does not include frames involved in collisions. Also described in IETF RFC1643 MIB dot3StatsDeferredTransmissions
	uint32	TxSingleCollision	A count of successfully transmitted frames on a particular interface for which transmission is inhibited by exactly one collision. A frame that is counted by an instance of this object is also counted by the corresponding instance of either the TxUniCastPkts and TxNUcastPkts and is not counted by the corresponding instance of the TxMultipleCollision object. Also described in IETF RFC1643 MIB dot3StatsSingleCollisionFrames
	uint32	TxMultipleCollision	A count of successfully transmitted frames on a particular interface for which transmission is inhibited by more than one collision. A frame that is counted by an instance of this object is also counted by the corresponding instance of either the TxUniCastPkts and TxNUcastPkts and is not counted by the corresponding instance of the TxSingleCollision object. Also described in IETF RFC1643 MIB dot3StatsMultipleCollisionFrames.
	uint32	TxLateCollision	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
	uint32	TxExcessiveCollison	A count of frames for which transmission on a particular interface fails due to excessive collisions. Also described in IETF RFC1643 MIB dot3StatsExcessiveCollisions
<b>Description:</b>		Statistic counters for diagnostics.	

] (SRS\_Eth\_00127)

## 8.3 Function definitions

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

### 8.3.1 Eth\_Init

[SWS\_Eth\_00027] [

Service name:	Eth_Init		
Syntax:	<pre>void Eth_Init(     const Eth_ConfigType* CfgPtr )</pre>		
Service ID[hex]:	0x01		
Sync/Async:	Synchronous		
Reentrancy:	Non Reentrant		
Parameters (in):	CfgPtr	Points to the implementation specific structure	
Parameters (inout):	None		
Parameters (out):	None		
Return value:	None		
Description:	Initializes the Ethernet Driver		

] () [SWS\_Eth\_00028]

The function shall store the access to the configuration structure for subsequent API calls. ]()

[SWS\_Eth\_00034] [

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

- Disable all controller
- 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]()

[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 otherwise (if DET is disabled) return E\_NOT\_OK, otherwise pass the production error ETH\_E\_ACCESS and return E\_OK. ]()

[SWS\_Eth\_00031] [

Caveat: The API has to be called during initialization. ]()

### 8.3.2 Eth\_SetControllerMode

[SWS\_Eth\_00041] [

<b>Service name:</b>	Eth_SetControllerMode	
<b>Syntax:</b>	<pre>Std_ReturnType Eth_SetControllerMode(     uint8 CtrlIdx,     Eth_ModeType CtrlMode )</pre>	
<b>Service ID[hex]:</b>	0x03	
<b>Sync/Async:</b>	Asynchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	CtrlMode	ETH_MODE_DOWN: disable the controller ETH_MODE_ACTIVE: enable the controller
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: success E_NOT_OK: controller mode could not be changed
<b>Description:</b>	Enables / disables the indexed controller	

] () [SWS\_Eth\_00042] [

The function shall:

- Put the controller in the specified mode given in the parameter 'CtrlMode'
  - Upon mode ETH\_MODE\_DOWN the driver shall:
    - Disable 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
    - Enable the Ethernet controller ]()

[SWS\_Eth\_00043] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED otherwise (if DET is disabled) return E\_NOT\_OK. ]()

[SWS\_Eth\_00044] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX otherwise (if DET is disabled) return E\_NOT\_OK. J()

[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, otherwise pass the production error ETH\_E\_ACCESS and return E\_OK. J())

[SWS\_Eth\_00045] [

Caveat: The function requires previous controller initialization (Eth\_Init). J())

### 8.3.3 Eth\_GetControllerMode

[SWS\_Eth\_00046] [

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

] () [SWS\_Eth\_00047] [

The function shall read the current controller mode. J())

[SWS\_Eth\_00048] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED otherwise (if DET is disabled) return E\_NOT\_OK. J())

[SWS\_Eth\_00049] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX otherwise (if DET is disabled) return E\_NOT\_OK. J())

[SWS\_Eth\_00050] [

If default error detection is enabled: the function shall check the parameter CtrlModePtr for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER otherwise (if DET is disabled) return E\_NOT\_OK. J())

[SWS\_Eth\_00051] [

Caveat: The function requires previous controller initialization (Eth\_Init). J())

### 8.3.4 Eth\_GetPhysAddr

#### [SWS\_Eth\_00052] [

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

] ()

#### [SWS\_Eth\_00053] [

The function shall read the source address used by the indexed controller. ]()

#### [SWS\_Eth\_00054] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED. ]()

#### [SWS\_Eth\_00055] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX. ]()

#### [SWS\_Eth\_00056] [

If default error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER. ]()

#### [SWS\_Eth\_00057] [

Caveat: The function requires previous controller initialization (Eth\_Init). ]()

### 8.3.5 Eth\_SetPhysAddr

#### [SWS\_Eth\_00151] [

<b>Service name:</b>	Eth_SetPhysAddr	
<b>Syntax:</b>	<pre>void Eth_SetPhysAddr(     uint8 CtrlIdx,     const uint8* PhysAddrPtr )</pre>	
<b>Service ID[hex]:</b>	0x13	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant for the same CtrlIdx, reentrant for different	
<b>Parameters (in):</b>	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Driver.

	PhysAddrPtr	Pointer to memory containing the physical source address (MAC address) in network byte order.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Sets the physical source address used by the indexed controller	

] () [SWS\_Eth\_00139] [

The function shall update the source address used by the indexed controller. ]()

[SWS\_Eth\_00140] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED. ]()

[SWS\_Eth\_00141] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX. ]()

[SWS\_Eth\_00142] [

If default error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER. ]()

[SWS\_Eth\_00143] [

Caveat: The function requires previous controller initialization (Eth\_Init). ]()

### 8.3.6 Eth\_UpdatePhysAddrFilter

[SWS\_Eth\_00152] [

<b>Service name:</b>	Eth_UpdatePhysAddrFilter	
<b>Syntax:</b>	<pre>Std_ReturnType Eth_UpdatePhysAddrFilter(     uint8 CtrlIdx,     const uint8* PhysAddrPtr,     Eth_FilterActionType Action )</pre>	
<b>Service ID[hex]:</b>	0x12	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant for the same CtrlIdx, reentrant for different	
<b>Parameters (in):</b>	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet 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 Ethernet packet.
	Action	Add or remove the address from the Ethernet controllers filter.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: filter was successfully changed E_NOT_OK: filter could not be changed
<b>Description:</b>	Update the physical source address to/from the indexed controller filter. If the	

	Ethernet Controller is not capable to do the filtering, the software has to do this.
--	--

] () [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 default 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 default error ETH\_E\_NOT\_INITIALIZED. ]()

[SWS\_Eth\_00165] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX. ]()

[SWS\_Eth\_00166] [

If default error detection is enabled the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER. ]()

[SWS\_Eth\_00167] [

Caveat: The function requires previous controller initialization (Eth\_Init). ]()

[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] [

<b>Service name:</b>	Eth_WriteMii	
<b>Syntax:</b>	<pre>Std_ReturnType Eth_WriteMii(     uint8 CtrlIdx,     uint8 TrcvIdx,     uint8 RegIdx,     uint16 RegVal )</pre>	
<b>Service ID[hex]:</b>	0x05	
<b>Sync/Async:</b>	Asynchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	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)
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
<b>Description:</b>	Configures a transceiver register or triggers a function offered by the receiver	

#### ] () [SWS\_Eth\_00059] [

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

#### [SWS\_Eth\_00241]]

The function shall call EthTrcv\_WriteMiiIndication when the MII access finished. ] ()

#### [SWS\_Eth\_00060] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED. ]()

#### [SWS\_Eth\_00061] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX. ]()

#### [SWS\_Eth\_00062] [

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

#### [SWS\_Eth\_00063] [

Caveat: The function requires previous controller initialization (Eth\_Init). ]()

### 8.3.8 Eth\_ReadMii

#### [SWS\_Eth\_00064] [

<b>Service name:</b>	Eth_ReadMii
----------------------	-------------



<b>Syntax:</b>	<pre>Std_ReturnType Eth_ReadMii (     uint8 CtrlIdx,     uint8 TrcvIdx,     uint8 RegIdx,     uint16* RegValPtr )</pre>	
<b>Service ID[hex]:</b>	0x06	
<b>Sync/Async:</b>	Asynchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	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)
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	RegValPtr	Filled with the register content of the indexed register (see [21] for details)
	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
<b>Description:</b>	Reads a transceiver register	

] () [SWS\_Eth\_00065] [

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

[SWS\_Eth\_00242] [

The function shall call EthTrcv\_ReadMiiIndication when the MII access finished. ] ()

[SWS\_Eth\_00066] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED. ]()

[SWS\_Eth\_00067] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX. ]()

[SWS\_Eth\_00068] [

If default error detection is enabled: the function shall check the parameter RegValPtr for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER. ]()

[SWS\_Eth\_00069] [

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

[SWS\_Eth\_00070] [

Caveat: The function requires previous controller initialization (Eth\_Init). ]()

### 8.3.9 Eth\_GetCounterValues

[SWS\_Eth\_00226] [

<b>Service name:</b>	Eth_GetCounterValues
----------------------	----------------------

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

] (SRS\_Eth\_00127) [SWS\_Eth\_00227] [

The function shall read a list of values from the indexed controller. ]()

[SWS\_Eth\_00228] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED otherwise (if DET is disabled) return E\_NOT\_OK. ]()

[SWS\_Eth\_00229] [

If dev default elopment error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX otherwise (if DET is disabled) return E\_NOT\_OK. ]()

[SWS\_Eth\_00230] [

If default error detection is enabled: the function shall check the parameter CounterPtr for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER otherwise (if DET is disabled) return E\_NOT\_OK. ]()

[SWS\_Eth\_00231] [

The function Eth\_GetCounterValues shall be pre compile time configurable On/Off by the configuration parameter: EthGetCounterValuesApi. ]()

[SWS\_Eth\_00232] [

Caveat: The function requires previous controller initialization (Eth\_Init). ]()

### 8.3.10 Eth\_GetRxStats

[SWS\_Eth\_00233] [

<b>Service name:</b>	Eth_GetRxStats	
<b>Syntax:</b>	Std_ReturnType Eth_GetRxStats( uint8 CtrlIdx, Eth_RxStatsType* RxStats )	
<b>Service ID[hex]:</b>	0x15	

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

] (SRS\_Eth\_00127) [SWS\_Eth\_00234] [

The function shall read a list of values from the indexed controller according to [22].

]()

[SWS\_Eth\_00235] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED otherwise (if DET is disabled) return E\_NOT\_OK. ]()

[SWS\_Eth\_00236] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX otherwise (if DET is disabled) return E\_NOT\_OK. ]()

[SWS\_Eth\_00237] [

If default error detection is enabled: the function shall check the parameter RxStats for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER otherwise (if DET is disabled) return E\_NOT\_OK. ]()

[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] [

<b>Service name:</b>	Eth_GetTxStats	
<b>Syntax:</b>	<pre>Std_ReturnType Eth_GetTxStats(     uint8 CtrlIdx,     Eth_TxStatsType* TxStats )</pre>	
<b>Service ID[hex]:</b>	0x1c	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	CtrlIdx	Index of the controller within the context of the Ethernet Driver
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	TxStats	List of values to read statistic values for transmission.
<b>Return value:</b>	Std_ReturnType	E_OK: success, E_NOTOK: Tx-statistics could not be obtained
<b>Description:</b>	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.	

] (SRS\_Eth\_00127)

#### [SWS\_Eth\_00248][

If default 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 default error ETH\_E\_NOT\_INITIALIZED otherwise (if DET is disabled) return E\_NOT\_OK.]  
(SRS\_BSW\_00101, SRS\_BSW\_00416)

#### [SWS\_Eth\_00249][

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX otherwise (if DET is disabled) return E\_NOT\_OK.]  
(SRS\_BSW\_00323, SRS\_BSW\_00369)

#### [SWS\_Eth\_00250][

If default error detection is enabled: the function shall check the parameter TxStats for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER otherwise (if DET is disabled) return E\_NOT\_OK.]  
(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.] (SRS\_Eth\_00053)

### 8.3.12 Eth\_GetTxErrorCounterValues

#### [SWS\_Eth\_91006] [

<b>Service name:</b>	Eth_GetTxErrorCounterValues	
<b>Syntax:</b>	Std_ReturnType Eth_GetTxErrorCounterValues (	

	<pre>uint8 CtrlIdx, Eth_TxErrorCounterValuesType* TxErrorCounterValues )</pre>	
<b>Service ID[hex]:</b>	0x1d	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	CtrlIdx	Index of the controller within the context of the Ethernet Driver
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	TxErrorCounterValues	List of values to read statistic error counter values for transmission.
<b>Return value:</b>	Std_ReturnType	E_OK: success, E_NOTOK: Tx-statistics could not be obtained
<b>Description:</b>	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.	

] (SRS\_Eth\_00127)

[SWS\_Eth\_00252]

If default 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 default error ETH\_E\_NOT\_INITIALIZED otherwise (if DET is disabled) return E\_NOT\_OK.]  
(SRS\_BSW\_00101, SRS\_BSW\_00416)

[SWS\_Eth\_00253]

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX otherwise (if DET is disabled) return E\_NOT\_OK.]  
(SRS\_BSW\_00323, SRS\_BSW\_00369)

[SWS\_Eth\_00254]

If default error detection is enabled: the function shall check the parameter TxStats for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER otherwise (if DET is disabled) return E\_NOT\_OK.]  
(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: EthGetTxErrorCounterValuesApi.]  
(SRS\_Eth\_00053)

### 8.3.13 Eth\_GetCurrentTime

[SWS\_Eth\_00181]

<b>Service name:</b>	Eth_GetCurrentTime
<b>Syntax:</b>	<pre>Std_ReturnType Eth_GetCurrentTime(     uint8 CtrlIdx,     Eth_TimeStampQualType* timeQualPtr,     Eth_TimeStampType* timeStampPtr )</pre>

	)	
<b>Service ID[hex]:</b>	0x16	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	CtrlIdx	Index of the addresses ETH controller.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
<b>Return value:</b>	Std_ReturnType	E_OK: successful
		E_NOT_OK: failed
<b>Description:</b>	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.	

] ()

[SWS\_Eth\_00182] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED. ]()

[SWS\_Eth\_00183] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX. ]()

[SWS\_Eth\_00184] [

If default 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 default error ETH\_E\_PARAM\_POINTER. ]()

[SWS\_Eth\_00210] [

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

[SWS\_Eth\_00185] [

Caveat: The function requires previous controller initialization (Eth\_Init). ]()

### 8.3.14 Eth\_EnableEgressTimeStamp

[SWS\_Eth\_00186] [

<b>Service name:</b>	Eth_EnableEgressTimeStamp	
<b>Syntax:</b>	<pre>void Eth_EnableEgressTimeStamp(     uint8 CtrlIdx,     Eth_BufIdxType BufIdx )</pre>	
<b>Service ID[hex]:</b>	0x17	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	CtrlIdx	Index of the addresses ETH controller.
	BufIdx	Index of the message buffer, where Application expects egress time stamping

<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	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.

] ()

[SWS\_Eth\_00187] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED. ]()

[SWS\_Eth\_00188] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX. ]()

[SWS\_Eth\_00211] [

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

[SWS\_Eth\_00189] [

Caveat: The function requires previous controller initialization (Eth\_Init). ]()

### 8.3.15 Eth\_GetEgressTimeStamp

[SWS\_Eth\_00190] [

<b>Service name:</b>	Eth_GetEgressTimeStamp	
<b>Syntax:</b>	<pre>void Eth_GetEgressTimeStamp(     uint8 CtrlIdx,     Eth_BufIdxType BufIdx,     Eth_TimeStampQualType* timeQualPtr,     Eth_TimeStampType* timeStampPtr )</pre>	
<b>Service ID[hex]:</b>	0x18	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	CtrlIdx	Index of the addresses ETH controller.
	BufIdx	Index of the message buffer, where Application expects egress time stamping
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
<b>Return value:</b>	None	
<b>Description:</b>	Reads back the egress time stamp on a dedicated message object. It must be called within the TxConfirmation() function.	

] ()

[SWS\_Eth\_00191] [



If default 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 default error ETH\_E\_NOT\_INITIALIZED. ]()

[SWS\_Eth\_00192] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX. ]()

[SWS\_Eth\_00193] [

If default 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 default error ETH\_E\_PARAM\_POINTER. ]()

[SWS\_Eth\_00212] [

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

[SWS\_Eth\_00194] [

Caveat: The function requires previous controller initialization (Eth\_Init). ]()

### 8.3.16 Eth\_GetIngressTimeStamp

[SWS\_Eth\_00195] [

<b>Service name:</b>	Eth_GetIngressTimeStamp	
<b>Syntax:</b>	<pre>void Eth_GetIngressTimeStamp(     uint8 CtrlIdx,     Eth_DataType* DataPtr,     Eth_TimeStampQualType* timeQualPtr,     Eth_TimeStampType* timeStampPtr )</pre>	
<b>Service ID[hex]:</b>	0x19	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	CtrlIdx	Index of the addresses ETH controller.
	DataPtr	Pointer to the message buffer, where Application expects ingress time stamping
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
<b>Return value:</b>	None	
<b>Description:</b>	Reads back the ingress time stamp on a dedicated message object. It must be called within the RxIndication() function.	

] ()

[SWS\_Eth\_00196] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED. ]()

[SWS\_Eth\_00197] [



If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX. ]()

[SWS\_Eth\_00198] [

If default 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 default error ETH\_E\_PARAM\_POINTER. ]()

[SWS\_Eth\_00213] [

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

[SWS\_Eth\_00199] [

Caveat: The function requires previous controller initialization (Eth\_Init). ]()

### 8.3.17 Eth\_ProvideTxBuffer

[SWS\_Eth\_00077] [

<b>Service name:</b>	Eth_ProvideTxBuffer	
<b>Syntax:</b>	<pre>BufReq_ReturnType Eth_ProvideTxBuffer(     uint8 CtrlIdx,     uint8 Priority,     Eth_BufIdxType* BufIdxPtr,     uint8** BufPtr,     uint16* LenBytePtr )</pre>	
<b>Service ID[hex]:</b>	0x09	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	Priority	Frame priority for transmit buffer FIFO selection
<b>Parameters (inout):</b>	LenBytePtr	In: desired length in bytes, out: granted length in bytes
<b>Parameters (out):</b>	BufIdxPtr	Index to the granted buffer resource. To be used for subsequent requests
	BufPtr	Pointer to the granted buffer
<b>Return value:</b>	BufReq_ReturnType	BUFREQ_OK: success BUFREQ_E_NOT_OK: development error detected BUFREQ_E_BUSY: all buffers in use BUFREQ_E_OVFL: requested buffer too large
<b>Description:</b>	Provides access to a transmit buffer of the FIFO related to the specified priority	

] () [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\_00137] [

All locked transmit buffers shall be released if the 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 default 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 default error ETH\_E\_NOT\_INITIALIZED and return BUFREQ\_E\_NOT\_OK. `()`

[SWS\_Eth\_00082]   
If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX and return BUFREQ\_E\_NOT\_OK. `()`

[SWS\_Eth\_00083]   
If default error detection is enabled: the function shall check the parameter BufIdxPtr for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER and return BUFREQ\_E\_NOT\_OK. `()`

[SWS\_Eth\_00084]   
If default error detection is enabled: the function shall check the parameter BufPtr for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER and return BUFREQ\_E\_NOT\_OK. `()`

[SWS\_Eth\_00085]   
If default error detection is enabled: the function shall check the parameter LenBytePtr for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER and return BUFREQ\_E\_NOT\_OK. `()`

[SWS\_Eth\_00086]   
Caveat: The function requires previous controller initialization (Eth\_Init). `()`

### 8.3.18 Eth\_Transmit

[SWS\_Eth\_00087]

<b>Service name:</b>	Eth_Transmit
<b>Syntax:</b>	<pre>Std_ReturnType Eth_Transmit(     uint8 CtrlIdx,     Eth_BufIdxType BufIdx,     Eth_FrameType FrameType,     boolean TxConfirmation,     uint16 LenByte,     const uint8* PhysAddrPtr )</pre>
<b>Service ID[hex]:</b>	0xA
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant for different buffer indexes and Ctrl indexes

<b>Parameters (in):</b>	CtrlIdx	Index of the controller within the context of the Ethernet 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
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: success E_NOT_OK: transmission failed
<b>Description:</b>	Triggers transmission of a previously filled transmit buffer	

] () [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. ]()

[SWS\_Eth\_00089] [

If TxConfirmation is false, the function shall release the buffer resource. ]()

[SWS\_Eth\_00138] [

All pending transmit buffers shall be released if the controller is disabled via Eth\_SetControllerMode. ]()

[SWS\_Eth\_00090] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED otherwise (if DET is disabled) return E\_NOT\_OK. ]()

[SWS\_Eth\_00091] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX otherwise (if DET is disabled) return E\_NOT\_OK. ]()

[SWS\_Eth\_00092] [

If default error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_PARAM otherwise (if DET is disabled) return E\_NOT\_OK. ]()

[SWS\_Eth\_00093] [

If default error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER otherwise (if DET is disabled) return E\_NOT\_OK. ]()

[SWS\_Eth\_00129] [

If default 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 default error ETH\_E\_INV\_MODE otherwise (if DET is disabled) return E\_NOT\_OK. ]()

[SWS\_Eth\_00094] [

Caveat: The function requires previous buffer request (Eth\_ProvideTxBuffer). ]()

### 8.3.19 Eth\_Receive

#### [SWS\_Eth\_00095] [

<b>Service name:</b>	Eth_Receive
<b>Syntax:</b>	<pre>void Eth_Receive(     uint8 CtrlIdx,     uint8 FifoIdx,     Eth_RxStatusType* RxStatusPtr )</pre>
<b>Service ID[hex]:</b>	0xB
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant for different FIFOs. Non Reentrant for the same FIFO.
<b>Parameters (in):</b>	CtrlIdx      Index of the controller within the context of the Ethernet Driver
	FifoIdx      Specifies the related fifo
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	RxStatusPtr      Indicates whether a frame has been received and if so, whether more frames are available for the related fifo.
<b>Return value:</b>	None
<b>Description:</b>	Receive a frame from the related fifo.

#### ] () [SWS\_Eth\_00096] [

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\_00097] [

If default 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 default error ETH\_E\_NOT\_INITIALIZED. ]()

#### [SWS\_Eth\_00098] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX. ]()

#### [SWS\_Eth\_00132] [

If default 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 default error ETH\_E\_INV\_MODE. ]()

#### [SWS\_Eth\_00153] [

When calling the callback function EthIf\_RxIndication broadcast frames shall be indicated to the Ethernet Interface (see [6]). ]()

#### [SWS\_Eth\_00099] [

Caveat: The function requires previous controller initialization (Eth\_Init). ]()

### 8.3.20 Eth\_TxConfirmation

#### [SWS\_Eth\_00100] [

<b>Service name:</b>	Eth_TxConfirmation
----------------------	--------------------

<b>Syntax:</b>	void Eth_TxConfirmation( uint8 CtrlIdx )
<b>Service ID[hex]:</b>	0xC
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	CtrlIdx   Index of the controller within the context of the Ethernet Driver
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	void   None
<b>Description:</b>	Triggers frame transmission confirmation

] () [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 default 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 default error ETH\_E\_NOT\_INITIALIZED. ]()

[SWS\_Eth\_00104] [

If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETH\_E\_INV\_CTRL\_IDX. ]()

[SWS\_Eth\_00134] [

If default 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 default error ETH\_E\_INV\_MODE. ]()

[SWS\_Eth\_00105] [

Caveat: The function requires previous initialization (Eth\_Init). ]()

### 8.3.21 Eth\_GetVersionInfo

[SWS\_Eth\_00106] [

<b>Service name:</b>	Eth_GetVersionInfo	
<b>Syntax:</b>	void Eth_GetVersionInfo( Std_VersionInfoType* VersionInfoPtr )	
<b>Service ID[hex]:</b>	0xD	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	VersionInfoPtr	Version information of this module

<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	void   None
<b>Description:</b>	Returns the version information of this module

] ()

[SWS\_Eth\_00136] [

If default error detection is enabled: the function shall check the parameter VersionInfoPtr for being valid. If the check fails, the function shall raise the default error ETH\_E\_PARAM\_POINTER. ]()

## 8.4 Callback notifications

The Ethernet Driver does not provide any callback functions.

## 8.5 Scheduled functions

### 8.5.1 Eth\_MainFunction

[SWS\_Eth\_00171] [

<b>Service name:</b>	Eth_MainFunction
<b>Syntax:</b>	void Eth_MainFunction( void )
<b>Service ID[hex]:</b>	0x0a
<b>Description:</b>	The function checks for controller errors and lost frames. Used for polling state changes. Calls EthIf_CtrlModeIndication when the controller mode changed.

] ()

[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 7.2.2 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] [

API function	Description
Dem_SetEventStatus	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.
EthIf_CtrlModeIndication	Called asynchronously when mode has been read out. Triggered by previous Eth_SetControllerMode call. Can directly be called within the trigger functions.
EthIf_RxIndication	Handles a received frame received by the indexed controller
EthIf_TxConfirmation	Confirms frame transmission by the indexed controller
SchM_Enter_Eth	Invokes the SchM_Enter function to enter a module local exclusive area.
SchM_Exit_Eth	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] [

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

] ()

### 8.6.3 Configurable interfaces

The Ethernet Driver does not use configurable interfaces.

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).



## 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 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

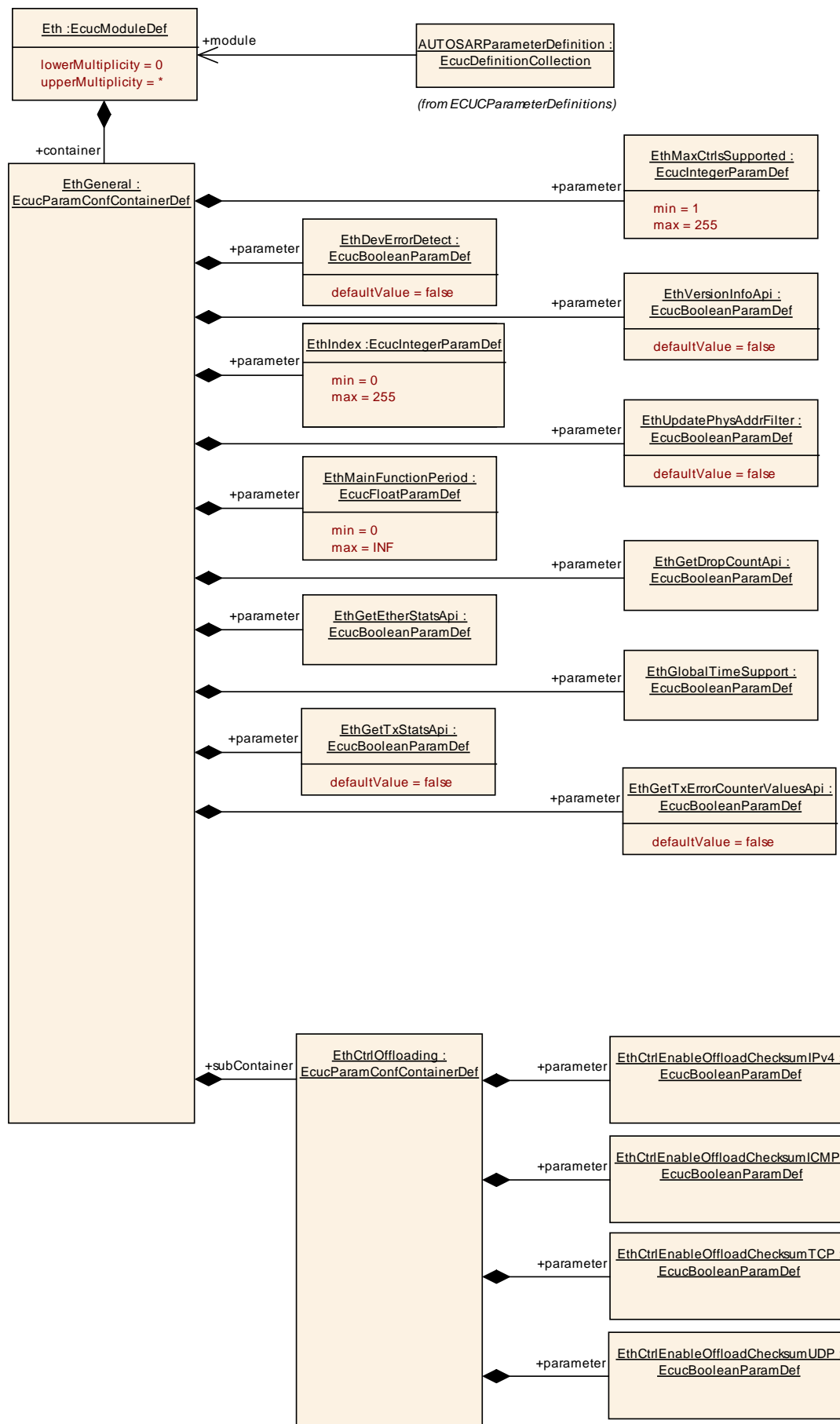
Chapter 10.2 specifies the structure (containers) and the parameters of the module Ethernet Driver.

Chapter 10.3 specifies published information of the module Ethernet Driver.

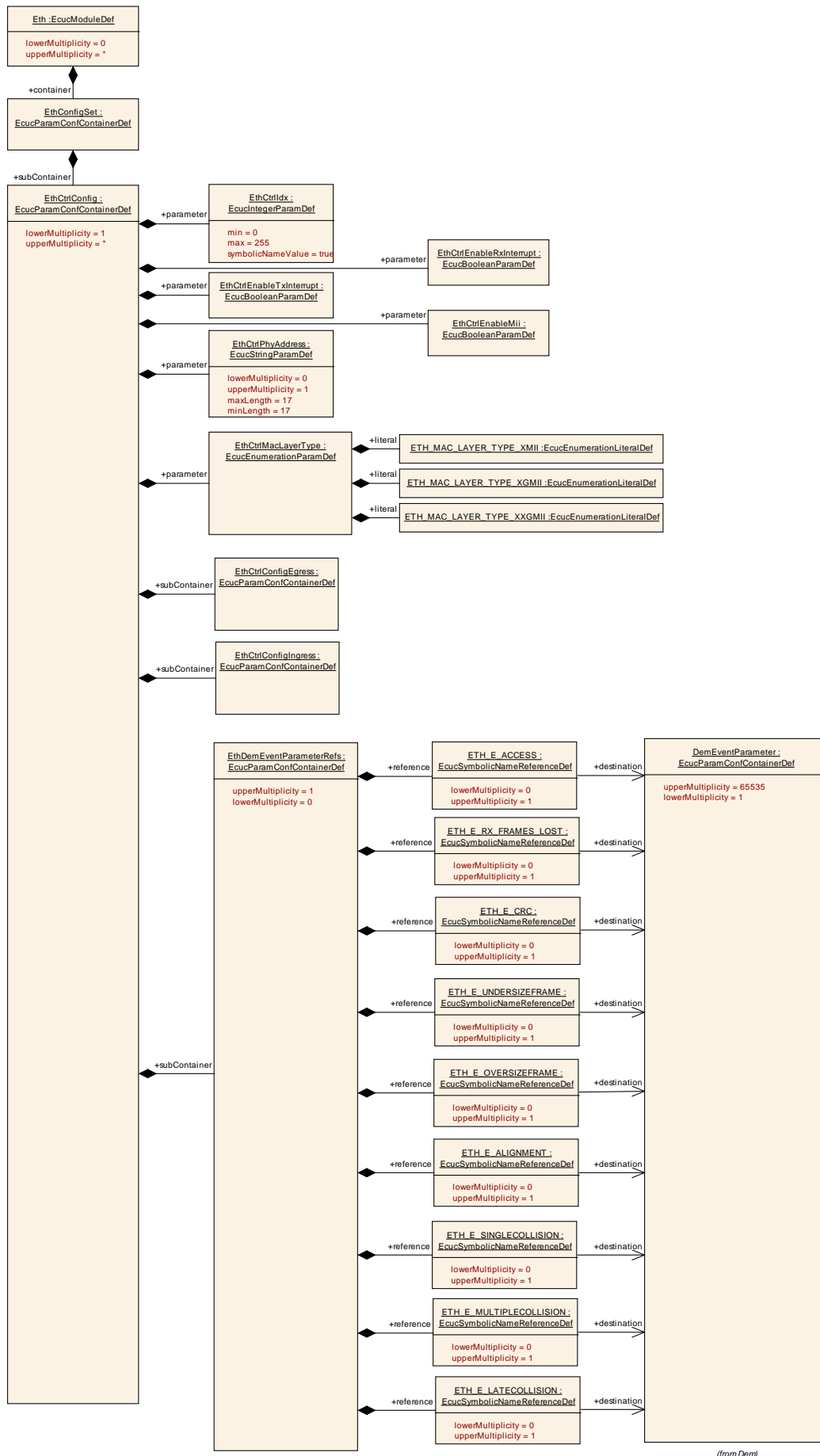
.

## 10.1 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 8.



**Figure 10.1: Ethernet Driver configuration structure**



(from Dem)

Figure 10.2: Ethernet Driver Controller configuration structure

### 10.1.1 Eth

<b>SWS Item</b>	<b>ECUC_Eth_00038 :</b>
<b>Module Name</b>	<i>Eth</i>
<b>Module Description</b>	Configuration of the Eth (Ethernet Driver) module.
<b>Post-Build Variant Support</b>	true
<b>Supported Config Variants</b>	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

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

### 10.1.2 EthConfigSet

<b>SWS Item</b>	<b>ECUC_Eth_00015 :</b>
<b>Container Name</b>	EthConfigSet
<b>Description</b>	This container contains the configuration parameters and sub containers of the AUTOSAR Eth module.
<b>Configuration Parameters</b>	

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

### 10.1.3 EthCtrlConfig

<b>SWS Item</b>	<b>ECUC_Eth_00006 :</b>
<b>Container Name</b>	EthCtrlConfig
<b>Description</b>	Configuration of the individual controller
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Eth_00012 :</b>		
<b>Name</b>	EthCtrlEnableMii		
<b>Description</b>	Enables / Disables Media Independent Interface (MII) for transceiver access		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00010 :</b>
-----------------	-------------------------

<b>Name</b>	EthCtrlEnableRxInterrupt		
<b>Description</b>	Enables / Disables receive interrupt		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00011 :</b>		
<b>Name</b>	EthCtrlEnableTxInterrupt		
<b>Description</b>	Enables / Disables transmit interrupt		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00007 :</b>		
<b>Name</b>	EthCtrlIdx		
<b>Description</b>	Specifies the instance ID of the configured controller.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>ECUC_Eth_00039 :</b>		
<b>Name</b>	EthCtrlMacLayerType		
<b>Description</b>	Defines the MAC layer type of the ethernet controller.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	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 100Mbit/s (e.g. RMII, RvMII, SMII, RvMII)	
	ETH_MAC_LAYER_TYPE_XXGMII	MAC layer interface (data) bandwith class 10Gbit/s	
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		



<b>SWS Item</b>	<b>ECUC_Eth_00020 :</b>		
<b>Name</b>	EthCtrlPhyAddress		
<b>Description</b>	Specifies the unique 48-bit physical address (MAC address) of the controller in network byte order. Regular Expression: [0-9a-fA-F]{2}[:-][0-9a-fA-F]{2}{5}		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucStringParamDef		
<b>Default value</b>	--		
<b>maxLength</b>	17		
<b>minLength</b>	17		
<b>regularExpression</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
EthCtrlConfigEgress	1	Configuration of one Ethernet controller egress behavior.
EthCtrlConfigIngress	1	Configuration of one Ethernet controller ingress behavior.
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.

#### 10.1.4 EthCtrlConfigEgress

<b>SWS Item</b>	<b>ECUC_Eth_00046 :</b>
<b>Container Name</b>	EthCtrlConfigEgress
<b>Description</b>	Configuration of one Ethernet controller egress behavior.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Eth_00052 :</b>		
<b>Name</b>	EthCtrlConfigEgressLastSchedulerRef		
<b>Description</b>	Reference to the scheduler which is the last in the egress structure.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ EthCtrlConfigScheduler ]		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>

EthCtrlConfigEgressFifo	0..*	Represents a Fifo at the egress side.
EthCtrlConfigScheduler	1..*	Represents a Scheduler on the egress side.
EthCtrlConfigShaper	0..*	Represents a Shaper an the egress side.

### 10.1.5 EthCtrlConfigEgressFifo

<b>SWS Item</b>	<b>ECUC_Eth_00047 :</b>		
<b>Container Name</b>	EthCtrlConfigEgressFifo		
<b>Description</b>	Represents a Fifo at the egress side.		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_Eth_00051 :</b>		
<b>Name</b>	EthCtrlConfigEgressFifoBufLenByte		
<b>Description</b>	Fifo buffer length in bytes.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00050 :</b>		
<b>Name</b>	EthCtrlConfigEgressFifoBufTotal		
<b>Description</b>	Fifo buffer count.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00048 :</b>		
<b>Name</b>	EthCtrlConfigEgressFifIdx		
<b>Description</b>	Egress Fifo index.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME

	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00049 :</b>		
<b>Name</b>	EthCtrlConfigEgressFifoPriorityAssignment		
<b>Description</b>	Message egress priority assignment.		
<b>Multiplicity</b>	0..*		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 7		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

### 10.1.6 EthCtrlConfigScheduler

<b>SWS Item</b>	<b>ECUC_Eth_00053 :</b>		
<b>Container Name</b>	EthCtrlConfigScheduler		
<b>Description</b>	Represents a Scheduler on the egress side.		
<b>Configuration Parameters</b>			

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
EthCtrlConfigSchedulerPredecessor	1..*	Defines an ordered list of predecessors for this scheduler.

### 10.1.7 EthCtrlConfigSchedulerPredecessor

<b>SWS Item</b>	<b>ECUC_Eth_00054 :</b>		
<b>Container Name</b>	EthCtrlConfigSchedulerPredecessor		
<b>Description</b>	Defines an ordered list of predecessors for this scheduler.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_Eth_00055 :</b>		
<b>Name</b>	EthCtrlConfigSchedulerPredecessorOrder		
<b>Description</b>	Defines the order of the scheduler predecessors.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 18446744073709551615		
<b>Default value</b>	--		

<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00056 :</b>		
<b>Name</b>	EthCtrlConfigSchedulerPredecessorRef		
<b>Description</b>	Choice reference to the scheduler predecessor.		
<b>Multiplicity</b>	1		
<b>Type</b>	Choice reference to [ EthCtrlConfigEgressFifo , EthCtrlConfigScheduler , EthCtrlConfigShaper ]		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

### 10.1.8 EthCtrlConfigShaper

<b>SWS Item</b>	<b>ECUC_Eth_00057 :</b>		
<b>Container Name</b>	EthCtrlConfigShaper		
<b>Description</b>	Represents a Shaper an the egress side.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_Eth_00058 :</b>		
<b>Name</b>	EthCtrlConfigShaperIdleSlope		
<b>Description</b>	Defines the increase of credit in bits per second for the AVB shaper.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 18446744073709551615		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00059 :</b>		
<b>Name</b>	EthCtrlConfigShaperPredecessorFifoRef		
<b>Description</b>	Reference to the fifo which is the predecessor for this shaper.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ EthCtrlConfigEgressFifo ]		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

### 10.1.9 EthCtrlConfigIngress

<b>SWS Item</b>	<b>ECUC_Eth_00040 :</b>
<b>Container Name</b>	EthCtrlConfigIngress
<b>Description</b>	Configuration of one Ethernet controller ingress behavior.
<b>Configuration Parameters</b>	

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
EthCtrlConfigIngressFifo	0..*	Represents a Fifo at the ingress side.

### 10.1.10 EthCtrlConfigIngressFifo

<b>SWS Item</b>	<b>ECUC_Eth_00041 :</b>		
<b>Container Name</b>	EthCtrlConfigIngressFifo		
<b>Description</b>	Represents a Fifo at the ingress side.		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_Eth_00045 :</b>		
<b>Name</b>	EthCtrlConfigIngressFifoBufLenByte		
<b>Description</b>	Fifo buffer length in bytes.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00044 :</b>		
<b>Name</b>	EthCtrlConfigIngressFifoBufTotal		
<b>Description</b>	Fifo buffer count.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00043 :</b>		
<b>Name</b>	EthCtrlConfigIngressFifoldx		
<b>Description</b>	Ingress Fifo index.		

<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00042 :</b>		
<b>Name</b>	EthCtrlConfigIngressFifoPriorityAssignment		
<b>Description</b>	Message ingress priority assignment.		
<b>Multiplicity</b>	0..*		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 7		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

### 10.1.11 EthDemEventParameterRefs

<b>SWS Item</b>	<b>ECUC_Eth_00016 :</b>
<b>Container Name</b>	EthDemEventParameterRefs
<b>Description</b>	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.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Eth_00017 :</b>		
<b>Name</b>	ETH_E_ACCESS		
<b>Description</b>	Reference to the DemEventParameter which shall be issued when the error "Controller access failed" has occurred.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME

	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00026 :</b>		
<b>Name</b>	ETH_E_ALIGNMENT		
<b>Description</b>	Reference to the DemEventParameter which shall be issued when the error "Alignment Error" has occurred.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00023 :</b>		
<b>Name</b>	ETH_E_CRC		
<b>Description</b>	Reference to the DemEventParameter which shall be issued when the error "CRC Failure" has occurred.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00029 :</b>		
<b>Name</b>	ETH_E_LATECOLLISION		
<b>Description</b>	Reference to the DemEventParameter which shall be issued when the error "Late Collisions" has occurred.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		



<b>SWS Item</b>	<b>ECUC_Eth_00028 :</b>		
<b>Name</b>	ETH_E_MULTIPLECOLLISION		
<b>Description</b>	Reference to the DemEventParameter which shall be issued when the error "Multiple Collisions" has occurred.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00025 :</b>		
<b>Name</b>	ETH_E_OVERSIZEFRAME		
<b>Description</b>	Reference to the DemEventParameter which shall be issued when the error "Oversized Frame" has occurred.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00021 :</b>		
<b>Name</b>	ETH_E_RX_FRAMES_LOST		
<b>Description</b>	Reference to the DemEventParameter which shall be issued when the error "receive frames lost" has occurred.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00027 :</b>		
<b>Name</b>	ETH_E_SINGLECOLLISION		
<b>Description</b>	Reference to the DemEventParameter which shall be issued when the error "Single Collisions" has occurred.		



<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00024 :</b>		
<b>Name</b>	ETH_E_UNDERSIZEFRAME		
<b>Description</b>	Reference to the DemEventParameter which shall be issued when the error "Undersized Frame" has occurred.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

### 10.1.12 EthGeneral

<b>SWS Item</b>	<b>ECUC_Eth_00001 :</b>
<b>Container Name</b>	EthGeneral
<b>Description</b>	General configuration of Ethernet Driver module
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Eth_00003 :</b>		
<b>Name</b>	EthDevErrorDetect		
<b>Description</b>	Switches the development error detection and notification on or off. <ul style="list-style-type: none"> <li>true: detection and notification is enabled.</li> <li>false: detection and notification is disabled.</li> </ul>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local
---------------------------	--------------

<b>SWS Item</b>	<b>ECUC_Eth_00035 :</b>		
<b>Name</b>	EthGetDropCountApi		
<b>Description</b>	Enables / Disables Eth_GetCounterValues API.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00036 :</b>		
<b>Name</b>	EthGetEtherStatsApi		
<b>Description</b>	Enables / Disables Eth_GetEtherStats API.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00061 :</b>		
<b>Name</b>	EthGetTxErrorCounterValuesApi		
<b>Description</b>	Enables/Disables Eth_GetTxErrorCounterValues API.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00060 :</b>		
<b>Name</b>	EthGetTxStatsApi		
<b>Description</b>	Enables/Disables Eth_GetTxStats API.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00037 :</b>		
<b>Name</b>	EthGlobalTimeSupport		
<b>Description</b>	Enables/Disables the GlobalTime APIs used amongst others by Global Time Synchronization over Ethernet.		
<b>Multiplicity</b>	1		

<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00018 :</b>		
<b>Name</b>	EthIndex		
<b>Description</b>	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00022 :</b>		
<b>Name</b>	EthMainFunctionPeriod		
<b>Description</b>	Specifies the period of main function Eth_MainFunction in seconds. Ethernet driver does not require this information but the BSW scheduler.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	]0 .. INF[		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00002 :</b>		
<b>Name</b>	EthMaxCtrlsSupported		
<b>Description</b>	Limits the total number of supported controllers.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00019 :</b>		
<b>Name</b>	EthUpdatePhysAddrFilter		
<b>Description</b>	Enables/Disables optional API Eth_UpdatePhysAddrFilter.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		

<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00004 :</b>		
<b>Name</b>	EthVersionInfoApi		
<b>Description</b>	Enables / Disables version info API		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

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

### 10.1.13 EthCtrlOffloading

<b>SWS Item</b>	<b>ECUC_Eth_00030 :</b>
<b>Container Name</b>	EthCtrlOffloading
<b>Description</b>	Configuration of hardware offloading features.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_Eth_00032 :</b>		
<b>Name</b>	EthCtrlEnableOffloadChecksumICMP		
<b>Description</b>	Enables / Disables hardware offloading for ICMP checksums.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00031 :</b>		
<b>Name</b>	EthCtrlEnableOffloadChecksumIPv4		
<b>Description</b>	Enables / Disables hardware offloading for IPv4 checksums.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00033 :</b>		
<b>Name</b>	EthCtrlEnableOffloadChecksumTCP		
<b>Description</b>	Enables / Disables hardware offloading for TCP checksums.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_Eth_00034 :</b>		
<b>Name</b>	EthCtrlEnableOffloadChecksumUDP		
<b>Description</b>	Enables / Disables hardware offloading for UDP checksums.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>No Included Containers</b>
-------------------------------

## 11 Not applicable requirements

### [SWS\_Eth\_00999]

These requirements are not applicable to this specification (BSW00170).