Document Title	Specification of Ethernet Interface
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	417
Document Classification	Standard
Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.0

Document Change History

Date	Release	Changed by	Change Description
2016-11-30	4.3.0	AUTOSAR Release Management	Diagnostics access APIs added gPTP Timestamp rework Ethernet Switch enhancements (Port Groups) Wireless Ethernet support
2015-07-31	4.2.2	AUTOSAR Release Management	EthIf_TransceiverInit and EthIf_ControllerInit removed Development Error Tracer renamed to Default Error Tracer
2014-10-31	4.2.1	AUTOSAR Release Management	Change from Synchronous to Asynchronous API gPTP Timestamp Support Ethernet Switch Support Ethernet Wakeup Support
2014-03-31	4.1.3	AUTOSAR Release Management	Extended UL_RxIndication Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	Introduction of Eth_GeneralTypes.h Support of API deviation for asynchronous implementation Changes in API of EthIf_ProvideTxBuffer and EthIf_SetPhysAddr Editorial changes Removed chapter(s) on change documentation

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2013-03-15	4.1.1	AUTOSAR Administration	Remove „Commercial Off The Shelf“ use case VLAN support 1000MBit Ethernet support
2011-12-22	4.0.3	AUTOSAR Administration	Description of payload data in EthIf_Cbk_RxIndication adapted
2010-09-30	3.1.5	AUTOSAR Administration	Further post-build configurable parameters EthIf_MainFunctionTx functional requirements improved (functionality split) 'Instance ID' removed from Version Info (concerns EthIf_GetVersionInfo API) Additional development error in EthIf_GetVersionInfo API
2010-02-02	3.1.4	AUTOSAR Administration	Initial Release

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

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 Interface.

In the AUTOSAR Layered Software Architecture, the Ethernet Interface belongs to the *ECU Abstraction Layer*, or more precisely, to the *Communication Hardware Abstraction*.

This indicates the main task of the Ethernet Interface:
Provide to upper layers a hardware independent interface to the Ethernet Communication System comprising multiple different Ethernet controllers and transceivers. This interface shall be uniform for all Ethernet controllers and transceivers. Thus, the upper layers (TCP/IP, EthSM, CDD) may access the underlying bus system in a uniform manner.

The Ethernet Interface does not directly access the Ethernet hardware (Ethernet Communication Controller and Ethernet Transceiver) but by means of one or more hardware-specific driver modules.

[SWS_EthIf_00111]⌈
In order to access the Ethernet controller(s), the Ethernet Interface shall use one or multiple Ethernet Driver modules, which abstract the specific features and interfaces of the respective Ethernet controller(s).⌋ ()

[SWS_EthIf_00123]⌈
In order to access the Ethernet transceiver(s), the Ethernet Interface shall use one or multiple Ethernet Transceiver Driver modules, which abstract the specific features and interfaces of the respective Ethernet transceiver(s).⌋ ()

[SWS_EthIf_00228]⌈
In order to access the Ethernet switch(es), the Ethernet Interface shall use one or multiple Ethernet Switch Driver modules, which abstract the specific features and interfaces of the respective Ethernet switch(es).⌋ ()

[SWS_EthIf_00112]⌈
Therefore, the Ethernet Interface executable code (however, not the configuration used during runtime) shall be completely independent of the Ethernet Communication Controller(s).⌋ ()

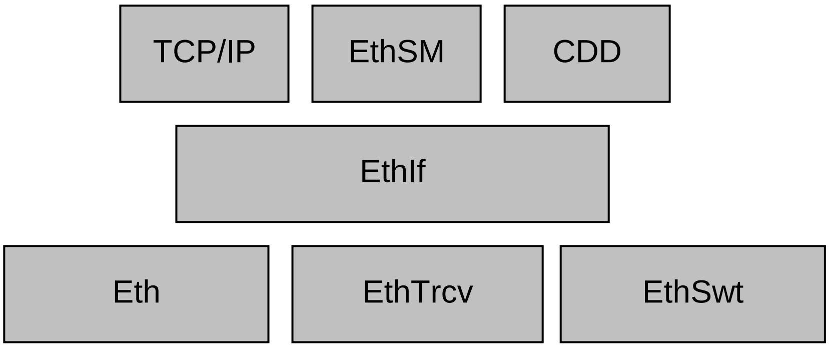

Figure 1: Ethernet stack module overview

Note: The Ethernet Interface 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 Interface can be carried out largely without detailed knowledge of the underlying hardware.

2 Acronyms and abbreviations

Abbreviation / Acronym:	***Description:***
Cascaded switch	A virtual switch comprising one Master switch and one or several slave switches.
CBR	Channel Busy Ratio
CIT	Channel Idle Time
Eth	Ethernet Controller Driver (AUTOSAR BSW module)
EthIf	Ethernet Interface (AUTOSAR BSW module)
EthSM	Ethernet State Manager (AUTOSAR BSW module)
EthTrcv	Ethernet Transceiver Driver (AUTOSAR BSW module)
Host port	A host port is a port of an automotive Ethernet switch where the data interface (e.g. MII) of the host ecu is connected to. The host port could either be an internal port or an external port. The host port has a special role from the perspective of the software. (see link accumulation and port groups)
IP	Internet Protocol
Master switch	This is a switch where the host port is located
MCG	Module Configuration Generator
MII	Media Independent Interface (standardized Interface provided by Ethernet controllers to access Ethernet transceivers)
RSSI	Received Signal Strength Indicator
Slave switch	A slave switch is connected to a master switch by one up link port
TCP	Transmission Control Protocol
TCP/IP Stack	Ethernet communication stack
Up link port	An up link port is a port of an automotive Ethernet switch which is connected to another Ethernet automotive switch (cascaded switch). An up link port could either be an internal port or an external port. One up link port is connected to another up link port. The up link port has a special role from the perspective of the software
VLAN	Virtual Local Area Network
WEth	Wireless Ethernet Driver
WEthTrcv	Wireless Ethernet Transceiver Driver

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] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[4] Requirements on Ethernet Support in AUTOSAR
AUTOSAR_SRS_Ethernet.pdf

[5] Specification of Ethernet Driver
AUTOSAR_SWS_EthernetDriver.pdf

[6] Specification of Ethernet State Manager
AUTOSAR_SWS_EthernetStateManager.pdf

[7] Specification of Ethernet Transceiver Driver
AUTOSAR_SWS_EthernetTransceiver.pdf

[8] Specification of TCP/IP
AUTOSAR_SWS_TcpIp.pdf

[9] Specification of PDU Router
AUTOSAR_SWS_PDURouter.pdf

[10] BSW Scheduler Specification
AUTOSAR_SWS_Scheduler.pdf

[11] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[12] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

[13] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[14] Specification of Default Error Tracer
AUTOSAR_SWS_DefaulttErrorTracer.pdf

[15] Specification of Diagnostics Event Manager
AUTOSAR_SWS_DiagnosticEventManager

[16] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

[17] Specification of ECU State Manager Fix
AUTOSAR_SWS_ECUStateManagerFixed.pdf

[18] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

[19] AUTOSAR Specification of Global Time Synchronization over Ethernet
AUTOSAR_SWS_TimeSyncOverEthernet.pdf

[20] AUTOSAR Specification of Ethernet Switch Driver
AUTOSAR_SWS_EthernetSwitchDriver.pdf

[21] Wireless Ethernet Driver
AUTOSAR_SWS_WirelessEthernetDriver.pdf

[22] Wireless Ethernet Transceiver Driver
AUTOSAR_SWS_WirelessEthernetTransceiverDriver.pdf

3.2 Related standards and norms

[23] IEC 7498-1 The Basic Model, IEC Norm, 1994

[24] IEEE 802.3-2006

[25] IEEE 802.1Q-2011

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [18] (SWS BSW General), which is also valid for Ethernet Interface.

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

4 Constraints and assumptions

4.1 Limitations

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

The Ethernet Interface is conceptually able to access one or more Ethernet Driver and one or more Ethernet Transceiver Driver.

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

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 Interface module.

Modules that use Ethernet Interface module:

- Ethernet Communication Stack (TCP/IP Stack)
- Ethernet State Manager (EthSM)

Modules used by the Ethernet Interface module:

-

Dependencies to other Modules:

- The Ethernet Interface module doesn’t take care of configuring Ethernet Driver but requires its preceding initialization and configuration.
- The Ethernet Interface module doesn’t take care of configuring Ethernet Transceiver Driver but requires its preceding initialization and configuration.

[SWS_EthIf_00225]⌈
The EthIf shall include the following header file:

- EthSwt_<vendorID>_<Vendor specific name><driver abbreviation>.h for services and type definitions of the EthSwt (e.g.: EthSwt_99_Ext1.h).⌋ (SRS_BSW_00436)

[SWS_EthIf_00226]⌈
The EthIf shall include the following header files which contain the configuration data used by the EthIf:

- EthSwt_<vendorID>_<Vendor specific name><driver abbreviation>_Cfg.h for configuration data of the EthSwt (e.g.: EthSwt_99_Ext1_Cfg.h).⌋ (SRS_BSW_00436)

5.1 File structure

5.1.1 Header file structure

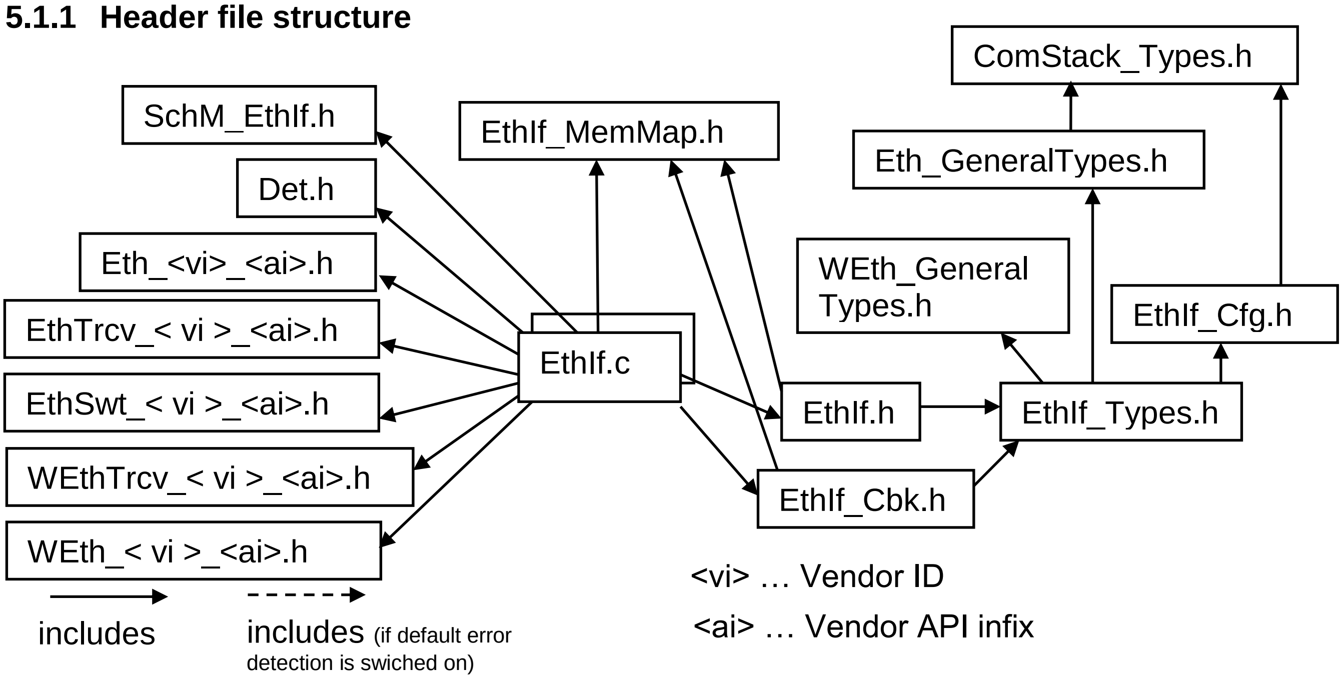

Figure 2: Ethernet Interface file structure

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_EthIf_00304, SWS_EthIf_00306
SRS_BSW_00436	-	SWS_EthIf_00225, SWS_EthIf_00226
SRS_Eth_00106	The Ethernet Transceiver Driver shall switch on/off wake up functionality at pre compile time.	SWS_EthIf_00237, SWS_EthIf_00245, SWS_EthIf_00249
SRS_Eth_00117	The Ethernet Transceiver Driver shall provide access to standardized hardware features	SWS_EthIf_91005, SWS_EthIf_91014, SWS_EthIf_91016, SWS_EthIf_91018, SWS_EthIf_91019, SWS_EthIf_91020
SRS_Eth_00125	The Ethernet Switch Driver shall support switch frame management	SWS_EthIf_91003, SWS_EthIf_91007

7 Functional specification

7.1 Ethernet BSW stack

As part of the AUTOSAR Layered Software Architecture according to [2], the Ethernet BSW modules also form a layered software stack. Figure 3 depicts the basic structure of this Ethernet BSW stack. The Ethernet Interface module accesses several Ethernet controllers using the Ethernet Driver layer, which can be made up of several Ethernet Drivers modules.

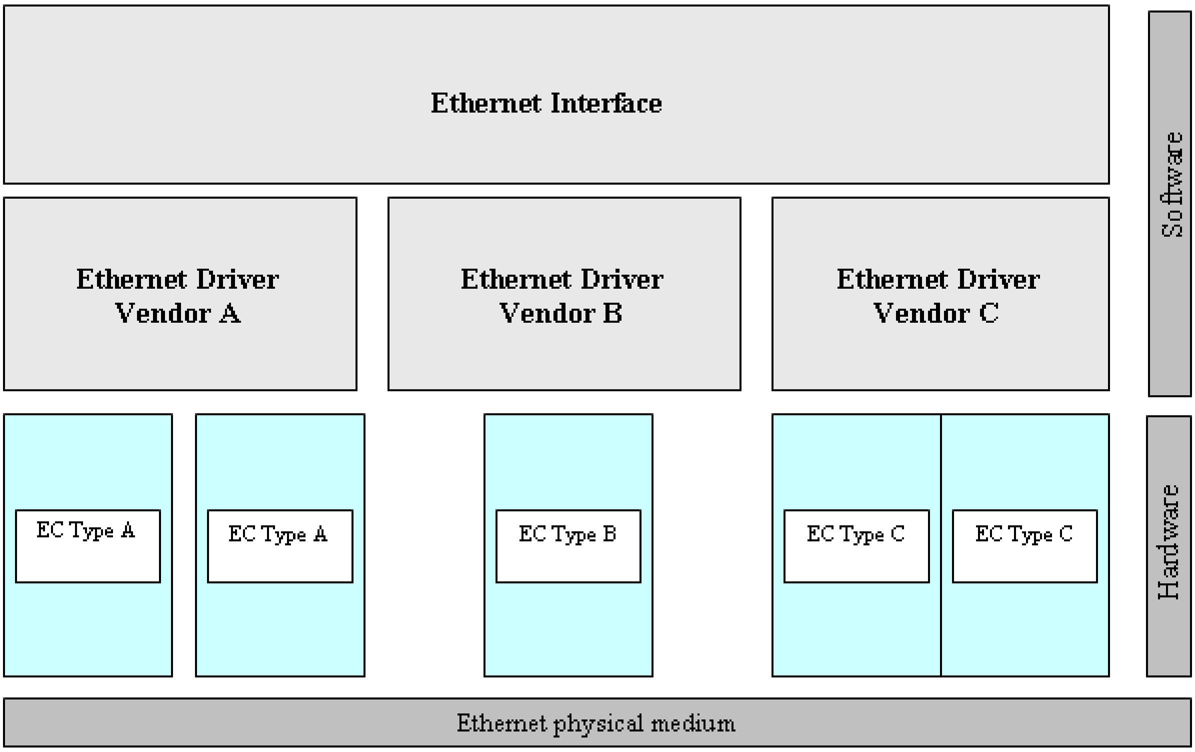

Figure 3: Basic Structure of the Ethernet BSW stack

7.1.1 Indexing scheme for Ethernet controller

Users of the Ethernet Interface identify Ethernet controller resources using an indexing scheme as depicted in Figure 4.

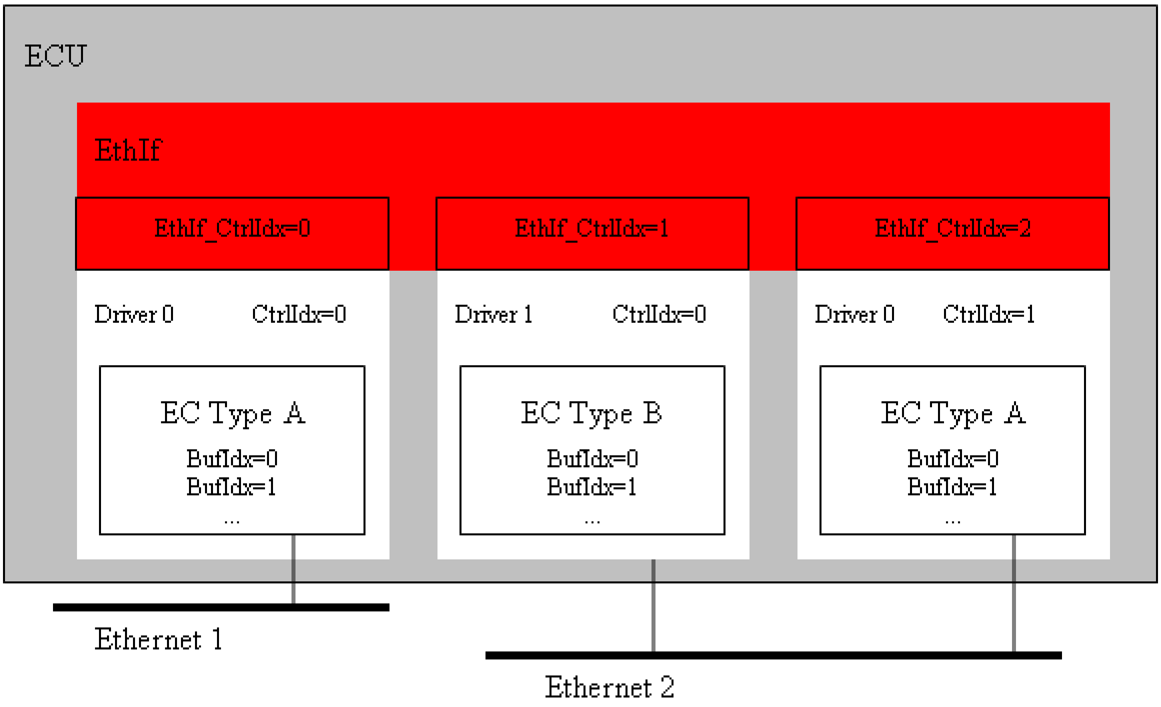

Figure 4: Ethernet Interface controller indexing scheme

[SWS_EthIf_00003] ⌈
The Ethernet Interface is using an index (EthIfCtrlIdx) to abstract the access to VLANs from the underlying communication system compromised of Ethernet Controller and Ethernet Transceiver.
Therefore the Ethernet Interface shall implement a mapping from Ethernet Interface controllers (EthIfCtrlIdx) to respective hardware ressource controllers (EthCtrlId + EthTrcvId).⌋ ()

7.1.2 Indexing scheme for Ethernet switches

Since the EthIf is not concerned with the individual EthSwtPorts which belong to the individual EthSwtes there is no indexing scheme for EthSwtPorts required in the EthIf. Any BSW module which interacts with EthSwtPorts can directly refer to the ECU configuration of the EthSwtPort for the indexing..

[SWS_EthIf_00224] ⌈
The EthIf shall dispatch all accesses by the EthIfSwitchIdx index to the respective EthSwt driver module with the EthSwtIdx value⌋ ()

7.1.3 Ethernet Interface main function

[SWS_EthIf_00004] ⌈
The Ethernet Interface shall implement main functions to be used for frame transmission confirmation and frame reception in polling mode with a calling period configurable at system configuration time.⌋()

7.1.4 Requirements

This chapter lists requirements that shall be fulfilled by Ethernet Interface module implementations.
The Ethernet Interface module environment comprises all modules which are calling interfaces of the Ethernet Interface module.

[SWS_EthIf_00005] ⌈
The Ethernet Interface module shall support pre-compile time, link time and post-build time configuration.⌋()

[SWS_EthIf_00006] ⌈
The header file *EthIf.h* shall include a software and specification version number.⌋()

[SWS_EthIf_00007] ⌈
The Ethernet Interface 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_EthIf_00008] ⌈
In case development error detection is enabled for the Ethernet Interface module: The Ethernet Interface module shall check API parameters for validity and report detected errors to the DET.⌋()

DET API functions are specified in [14].

[SWS_EthIf_00010] ⌈
The Ethernet Interface module shall implement the API functions specified by the Ethernet Interface SWS as real C-code functions and shall not implement the API as macros for object code deliveries.⌋()

[SWS_EthIf_00011] ⌈
None of the Ethernet Interface module header files shall define global variables.⌋()

7.1.5 Configuration description

[SWS_EthIf_00012] ⌈
The Ethernet Interface module shall provide an XML file that contains the data, which is required for the SW identification (it shall contain the vendor identification, module ID and software version information), configuration and integration process. This file should describe vendor specific configuration parameters as well as it should contain recommended configuration parameter values.⌋()

[SWS_EthIf_00117] ⌈
The MCG shall read the ECU configuration description of the Ethernet Driver and the Ethernet Interface module(s). While cluster related configuration parameters are contained in the Ethernet Interface module configuration description, Ethernet Driver related configuration data is contained in the Ethernet Driver module configuration description. The Ethernet Interface module specific configuration tool shall read both ECU module descriptions to derive the configuration data for all Ethernet Drivers mapped to the Ethernet Interface module.⌋()

[SWS_EthIf_00118] ⌈
The MCG shall ensure the consistency of the generated configuration data.⌋()

[SWS_EthIf_00013] ⌈
The configuration of the Ethernet Interface module shall be configured at ECU configuration time. None of the communication parameters shall be configured at runtime.⌋()

[SWS_EthIf_00014] ⌈
The start address of post-build time configuration data shall be passed during module initialization (see chapter 8.3.1).⌋()

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

A detailed description of all Ethernet Interface related configuration parameters can be found in chapter 10 of this document. Additionally, the configuration description of the Ethernet Driver (see chapter 10 of [5]) shall be evaluated for Ethernet Interface module configuration.

7.1.6 VLAN support

[SWS_EthIf_00128] ⌈
The Ethernet Interface shall support Virtual Local Area Networks (VLAN).⌋()

[SWS_EthIf_00129] ⌈
The Ethernet Interface shall encapsulate Virtual Local Area Networks (VLAN) into virtual controllers (Ethernet Interface controller) representing a dedicated VLAN.
All BSW modules above the Ethernet Interface shall interact based on those virtual controllers.

The Ethernet Driver and Transceiver deal only with real controllers and are not aware of the existence of virtual controllers.
Caveat: the virtual controller represents the untagged VLAN if no VLAN ID is set.⌋()

[SWS_EthIf_00130] ⌈
The Ethernet Interface shall use the buffers provided by the Ethernet Driver for VLAN support.⌋()

7.1.7 Wake up support

The Ethernet Interface supports wake up depending on the parameter EthIfWakeUpSupport.

Note: Enabling wake-up support in EthIf makes only sense if the underlying EthTrcv supports also wake up.

7.1.8 Switch Management support

Switch Management enables the possibility to control an Ethernet frame regarding a Switch-Port specific ingress and egress handling as well as providing a Switch-Port specific timestamp. This functionality is essential for other BSW modules, in particular for EthTSyn, which requires Port specific information associated to a time synchronization or path-delay measurement frame.

For an introduction of the basic HW architecture and interaction, please refer to [5]. For more details regarding functional sequences, please refer to [20].

Note: Switch management API's supporting the <Upper Layer> to gather / modify Switch-Port specific communication attributes.

7.1.9 Global Time support

For more details regarding time measurement with Switches, please refer to [19].

7.1.10 Switching of EthIfSwitchPortGroup

The Ethernet Interface supports wake up depending on the parameter EthIfWakeUpSupport EthIfSwitchPortGroups are requested to be ACTIVE or DOWN. The request will be handled and rated by the EthIf. EthIf has to decide either the EthIfSwichPortGroup is put to DOWN or ACTIVE state. ACTIVE-request for EthIfSwitchPortGroup will always overrule DOWN-request for EthIfSwitchPortGroups. If a DOWN-request for a EthIfSwitchPortGroup is ready for execution, the EthIf will check the EthSwtPorts which are referenced by the EthIfSwitchPortGroup and decide if the EthSwtPort can be set to DOWN state (switch off). If this is valid, the

EthSwtPort is set to DOWN state after the configured switch off delay timer has expired.

[SWS_EthIf_00256]⌈
EthIf shall delay the shutdown of an EthIfPhysController referencing a EthIfSwitch until all EthSwtPorts of the referenced switch are in state ETHTRCV_MODE_DOWN.
⌋ ()
Rationale: In case of using e.g. MDIO as control path for the EthSwt the EthIfPhysController should stay in ETH_MODE_ACTIVE until all EthSwt controlling actions (e.g. switch of EthSwtPorts) have been finished.

[SWS_EthIf_00257]⌈
If no EthIfSwitchPortGroup is configured, all EthSwtPorts shall be switched on if a least one EthIfController is requested with ETH_MODE_ACTIVE.⌋ ()

[SWS_EthIf_00258]⌈
If no EthIfSwitchPortGroup is configured, all EthSwtPorts shall be switched off if all EthIfController are requested with ETH_MODE_DOWN.⌋ ()

7.1.11 Link state accumulation of EthIfSwitchPortGroup

EthIf need to know the actual link state of the EthIfSwitchPortGroups. The link state for a EthIfSwitchPortGroup is computed over all link states of the EthSwtPorts which are referenced by the EthIfSwitchPortGroup. The execution of the computation is called the link state accumulation and the result is called the accumulated link state. The accumulated link state of the EthIfSwitchPortGroup is the actual state of the EthIfSwitchPortGroup. The actual state is reported to the EthSM state machine and additionally if the reference is type of "link-information" to the BswM by calling BswM_EthIf_PortGroupLinkStateChg.

[SWS_EthIf_00259]⌈
The link state for a EthIfSwitchPortGroup is computed over all link states of the EthSwtPorts which are referenced by the EthIfSwitchPortGroup. Its status is DOWN if one of the following conditions is met:
- Referenced EthSwtPort with the role "host port" or the role "up link port" has link down state
- All referenced EthSwtPort without a role have link down state
Otherwise its accumulated link state is link up."⌋ ()

[SWS_EthIf_00260]⌈
If the EthIfCtrl references a EthIfSwitch but no port group is configured, the EthIf shall indicate the link state of the host port to the EthSM by calling EthSM_TrcvLinkStateChg for the EthIfController when the link state changes.⌋ ()

[SWS_EthIf_00261]⌈
In case a EthIfSwitchPortGroup is not connected to any EthIfController, the EthIf shall indicate the accumulated link state of the EthIfSwitchPortGroup to the BswM by

calling BswM_EthIf_PortGroupLinkStateChg for the EthIfSwitchPortGroup when the link state changes (refer to SWS_EthIf_xxx21 for link state accumulation).⌋ ()

[SWS_EthIf_00262]⌈
In case a EthIfSwitchPortGroup is connected to a EthIfController, the EthIf shall indicate the accumulated link state of the EthIfSwitchPortGroup to the EthSM by calling EthSM_TrcvLinkStateChg for the EthIfController when the link state changes (refer to SWS_EthIf_00259 for link state accumulation).⌋ ()

7.1.12 Wireless Ethernet Support

[SWS_EthIf_00340]⌈
The Ethernet Interface shall support Wireless Ethernet specific functionality, depending on the parameter EthIfEnableWEthApi.⌋ ( )

The Wireless functions are divided in controller and transceiver specific functionality. Mainly, transmission and reception parameters are being exchanged with the EthIf upper module and the controller/transceiver.
The controller is being called only for buffer specific transmission and reception parameters by the APIs:

- EthIf_GetBufWRxParams
- EthIf_GetBufWTxParams
- EthIf_SetBufWTxParams

The Transceiver is being called for general configuration of the wireless radio and the wireless radio’s channel by:

- EthIf_SetRadioParams
- EthIf_SetChanRxParams
- EthIf_SetChanTxParams
- EthIf_GetChanRxParams

The parameter values are requested or transmitted by unique parameter identifiers. They are defined within the controller and transceiver specification [21] [22].

7.2 Error classification

7.2.1 Default Errors

[SWS_EthIf_00017] ⌈

Type or error	*Relevance*	*Related error code*	*Value [hex]*
Invalid controller index	Default Error	ETHIF_E_INV_CTRL_IDX	0x01
Invalid transceiver	Default	ETHIF_E_INV_TRCV_IDX	0x02

index	Error		
Invalid port group index	Default Error	ETHIF_E_INV_PORT_GROUP_IDX	0x03
EthIf module was not initialized	Default Error	ETHIF_E_NOT_INITIALIZED	0x04
Invalid pointer in parameter list	Default Error	ETHIF_E_PARAM_POINTER	0x05
Invalid parameter	Default Error	ETHIF_E_INV_PARAM	0x06
Initialization failure	Default Error	ETHIF_E_INIT_FAILED	0x07

⌋()

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.

7.2.1 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 Imported types

This chapter lists all types included from the following files:

[SWS_EthIf_00023] ⌈

Module	*Imported Type*
ComStack_Types	BufReq_ReturnType
EcuM	EcuM_WakeupSourceType
EthSwt	EthSwt_MgmtInfoType
Eth_GeneralTypes	EthTrcv_BaudRateType
	EthTrcv_CableDiagResultType
	EthTrcv_DuplexModeType
	EthTrcv_LinkStateType
	EthTrcv_ModeType
	EthTrcv_PhyLoopbackModeType
	EthTrcv_PhyTestModeType
	EthTrcv_PhyTxModeType
	EthTrcv_WakeupModeType
	Eth_BufIdxType
	Eth_DataType
	Eth_FilterActionType
	Eth_FrameType
	Eth_MacVlanType
	Eth_ModeType
	Eth_RateRatioType
	Eth_RxStatusType
	Eth_TimeIntDiffType
	Eth_TimeStampQualType
	Eth_TimeStampType
Std_Types	Std_ReturnType
	Std_VersionInfoType
WEth_GeneralTypes	WEthTrcv_GetChanRxParamIdType
	WEthTrcv_SetChanRxParamIdType
	WEthTrcv_SetChanTxParamIdType
	WEthTrcv_SetRadioParamIdType
	WEth_BufWRxParamIdType
	WEth_BufWTxParamIdType

⌋ ()

8.2 Type definitions

[SWS_EthIf_00152] ⌈
EthIf.h shall include Eth_GeneralTypes.h to include general Eth type declarations.⌋()

[SWS_EthIf_00153] ⌈
The types specified in SWS_EthernetInterface shall be declared in Eth_GeneralTypes.h.⌋()

8.2.1 EthIf_ConfigType

[SWS_EthIf_00149] ⌈

Name:	EthIf_ConfigType
Type:	Structure
Range:	Implementation specific.
Description:	Implementation specific structure of the post build configuration

⌋ ()

8.2.2 EthIf_StateType

[SWS_EthIf_00150] ⌈

Name:	EthIf_StateType		
Type:	Enumeration		
Range:	ETHCTRL_STATE_UNINIT	0x00	Ethernet Interface is not yet configured
	ETHCTRL_STATE_INIT	0x01	Ethernet Interface is configured
Description:	Status supervision used for Development Error Detection. The state shall be available for debugging.		

⌋ ()

8.2.3 EthIf_SwitchPortGroupIdxType

[SWS_EthIf_91101] ⌈

Name:	EthIf_SwitchPortGroupIdxType		
Type:	uint8		
Range:	0..255	--	--
Description:	Data Type that represents the Ethernet interface switch port group index. The index is zero based and unique for every configured switch port group.		

⌋ ()

8.2.4 EthIf_MeasurementIdxType

[SWS_EthIf_91010] ⌈

Name:	EthIf_MeasurementIdxType		
Type:	uint8		
Range:	ETHIF_MEAS_DROP_CRTLIDX	0x01	Measurement index of dropped datagrams caused by invalid CrtlIdx/VLAN
	ETHIF_MEAS_RESERVED_1	0x02-0x7F	reserved by AUTOSAR
	ETHIF_MEAS_RESERVED_2	0x80-0xEF	Vendor specific range
	ETHIF_MEAS_RESERVED_3	0xF0-0xFE	reserved by AUTOSAR (future use)
	ETHIF_MEAS_ALL	0xFF	represents all measurement indexes
Description:	Index to select specific measurement data		

⌋ ()

8.3 Function definitions

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

8.3.1 EthIf_Init

[SWS_EthIf_00024] ⌈

Service name:	EthIf_Init	

Syntax:	`void EthIf_Init(`
	`    const EthIf_ConfigType* CfgPtr`
	`)`	

Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CfgPtr	Points to the implementation specific structure
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the Ethernet Interface	

⌋ () [SWS_EthIf_00025] ⌈
The function shall store the access to the configuration structure for subsequent API calls.⌋()

[SWS_EthIf_00114] ⌈
The function shall change the state of the component from ETHIF_STATE_UNINIT to ETHIF_STATE_INIT.⌋()

[SWS_EthIf_00116] ⌈
If development error detection is enabled: the function shall check the parameter CfgPtr for containing a valid configuration. If the check fails, the function shall raise the development error ETHIF_E_INIT_FAILED.⌋()

[SWS_EthIf_00027] ⌈
Caveat: The API has to be called during initialization.⌋()

8.3.2 EthIf_SetControllerMode

[SWS_EthIf_00034] ⌈

Service name:	EthIf_SetControllerMode	

Syntax:	`Std_ReturnType EthIf_SetControllerMode(`
	`    uint8 CtrlIdx,`
	`    Eth_ModeType CtrlMode`
	`)`	

Service ID[hex]:	0x03	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	CtrlMode	ETH_MODE_DOWN: disable the controller

		ETH_MODE_ACTIVE: enable the controller
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: controller mode could not be changed
Description:	Enables / disables the indexed controller	

⌋ ()

[SWS_EthIf_00035] ⌈
The function EthIf_SetControllerMode shall forward the call to function Eth_SetControllerMode of the corresponding Ethernet Controller Driver (EthIfPhysControllerIdx) if mode ETH_MODE_ACTIVE has been requested first time for the Ethernet Interface Controller referencing the Ethernet Controller.⌋()

[SWS_EthIf_00263]⌈
The function EthIf_SetControllerMode shall forward the call to function Eth_SetControllerMode of the corresponding Ethernet Controller Driver (EthIfPhysControllerIdx) if mode ETH_MODE_DOWN has been requested for all Ethernet Interface Controller referencing the Ethernet Controller.⌋ ()
Note: in case of VLAN support, it means that EthIf has to store internally the state of each EthIfController in order to filter out the requests from upper layers and disable the callouts to upper layers when the EthIfController is disabled.

[SWS_EthIf_00264]⌈
If EthIf_SetController is called for an EthIfController with ETH_MODE_ACTIVE and this EthIfController has a reference to an EthIfSwitchPortGroup of type "control" then EthIf shall forward the call to function EthSwt_SetSwitchPortMode for all ports of the respective EthIfSwitchPortGroup if the mode ETHTRCV_MODE_ACTIVE has been requested for the first EthIfSwitchPortGroup referencing the port and the current port mode is ETHTRCV_MODE_DOWN.⌋()
Note: EthIfSWitchPortGroups that shall be switched according to PNC state are handled by BswM with the call of API EthIf_SwitchPortGroupRequestMode. This can be configured within the BswM via the BswMEthIfSwitchPortGroupRequestMode action.

[SWS_EthIf_00265]⌈
If EthIf_SetController is called for an EthIfController with ETH_MODE_DOWN and this EthIfController has a reference to an EthIfSwitchPortGroup of type "control" then EthIf shall forward the call to function EthSwt_SetSwitchPortMode for all ports of the respective EthIf_SwitchPortGroup if the mode ETHTRCV_MODE_DOWN has been requested for all Switch Port Groups referencing the port and the current mode is ETHTRCV_MODE_ACTIVE.⌋ ()

A call of the EthIf_SetControllerMode cause an asynchronous indication by calling EthIf_CtrlModeIndication, if the mode of the referenced EthIfPhysController has changed.

[SWS_EthIf_00266]⌈

In the context of EthIf_CtrlModeIndication the function EthTrcv_SetTransceiverMode shall be called if the EthIfController has a reference to a EthIfTransceiver.⌋ ()

[SWS_EthIf_00267]⌈
In the context of EthIf_CtrlModeIndication the function EthSwt_SetSwitchPortMode shall be called for all EthSwtPorts of a EthIfSwitchPortGroup if the EthIfController has a reference to a EthIfSwitchPortGroup and the reference is of type "control". If ETHTRCV_MODE_DOWN is requested, the EthIf has to ensure that only those EthSwtPorts are set to ETHTRCV_MODE_DOWN which are not requested ETHTRCV_MODE_ACTIVE by another EthIfSwitchPortGroup.⌋ ()

Rationale: In case the respective EthIfController has no reference to a EthIf_SwitchPortGroup or the reference is of type "link information" the requested modes are not forwarded. This EthIf_SwitchPortGroups will be requested by a upper layer (e.g. BswM) with API EthIf_SwitchPortGroupRequestMode.

[SWS_EthIf_00036] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00037] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00038] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.3 EthIf_GetControllerMode

[SWS_EthIf_00039] ⌈

Service name:	EthIf_GetControllerMode	

Syntax:	`Std_ReturnType EthIf_GetControllerMode(`
	`    uint8 CtrlIdx,`
	`    Eth_ModeType* CtrlModePtr`
	`)`	

Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
Parameters (inout):	None	
Parameters (out):	CtrlModePtr	ETH_MODE_DOWN: the controller is disabled ETH_MODE_ACTIVE: the controller is enabled
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: controller could not be initialized
Description:	Obtains the state of the indexed controller	

⌋ ()

[SWS_EthIf_00040] ⌈
The function EthIf_GetControllerMode shall forward the call to function Eth_GetControllerMode of the corresponding Ethernet Controller Driver (EthIfPhysControllerIdx).⌋()

[SWS_EthIf_00041] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00042] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00043] ⌈
If development error detection is enabled: the function shall check the parameter CtrlModePtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00044] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.4 EthIf_SetTransceiverWakeupMode

[SWS_EthIf_00233] ⌈

Service name:	EthIf_SetTransceiverWakeupMode	

Syntax:	`Std_ReturnType EthIf_SetTransceiverWakeupMode(`
	`    uint8 TrcvIdx,`
	`    EthTrcv_WakeupModeType TrcvWakeupMode`
	`)`	

Service ID[hex]:	0x2e	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
	TrcvWakeupMode	ETHTRCV_WUM_DISABLE: disable transceiver wake up ETHTRCV_WUM_ENABLE: enable transceiver wake up ETHTRCV_WUM_CLEAR: clears transceiver wake up reason
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: transceiver wake up could not be changed or wake-up reason could not be cleared
Description:	Enables / disables the wake up mode or clear the wake-up reason of the indexed transceiver	

⌋ ()

[SWS_EthIf_00234] ⌈

The function EthIf_SetTransceiverWakeupMode shall forward the call to function EthTrcv_SetTransceiverWakeupMode of the corresponding Ethernet Transceiver Driver (EthIfTransceiverIdx) if mode ETHTRCV_WUM_ENABLE has been requested and the current state of the requested Ethernet Transceiver Driver is ETHTRCV_WUM_DISABLE⌋()

[SWS_EthIf_00268] ⌈
The function EthIf_SetTransceiverWakeupMode shall forward the call to function EthTrcv_SetTransceiverWakeupMode of the corresponding Ethernet Transceiver Driver (EthIfTransceiverIdx) if mode ETHTRCV_WUM_DISABLE has been requested and the current state of the requested Ethernet Transceiver Driver is ETHTRCV_WUM_ENABLE.⌋()

[SWS_EthIf_00269] ⌈
The function EthIf_SetTransceiverWakeupMode shall forward the call to function EthTrcv_SetTransceiverWakeupMode of the corresponding Ethernet Transceiver Driver (EthIfTransceiverIdx) if mode ETHTRCV_WUM_CLEAR has been requested and the current state of the requested Ethernet Transceiver Driver is ETHTRCV_WUM_DISABLE.⌋()

[SWS_EthIf_00235] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00236] ⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00237] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter EthIfWakeUpSupport.⌋(SRS_Eth_00106)

8.3.5 EthIf_GetTransceiverWakeupMode

[SWS_EthIf_00238] ⌈

Service name:	EthIf_GetTransceiverWakeupMode	

Syntax:	`Std_ReturnType EthIf_GetTransceiverWakeupMode(`
	`    uint8 TrcvIdx,`
	`    EthTrcv_WakeupModeType* TrcvWakeupModePtr`
	`)`	

Service ID[hex]:	0x2f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
Parameters (inout):	None	
Parameters (out):	TrcvWakeupModePtr	ETHTRCV_WUM_DISABLE: transceiver wake up is disabled

		ETHTRCV_WUM_ENABLE: transceiver wake up is enabled
Return value:	Std_ReturnType	E_NOT_OK: transceiver wake up mode could not be obtained
Description:	Returns the wake up mode of the indexed transceiver	

⌋ ()
[SWS_EthIf_00239] ⌈
The function EthIf_GetTransceiverWakeupMode shall forward the call to function EthTrcv_GetTransceiverWakeupMode of the corresponding Ethernet Transceiver Driver (EthIfTransceiverIdx).⌋()

[SWS_EthIf_00240] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00241] ⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00242] ⌈
If development error detection is enabled: the function shall check the parameter TrcvWakeupModePtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00243] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter EthIfGetTransceiverWakeupModeApi.⌋()

8.3.6 EthIf_CheckWakeup

[SWS_EthIf_00244] ⌈

Service name:	EthIf_CheckWakeup	

Syntax:	`Std_ReturnType EthIf_CheckWakeup(`
	`    EcuM_WakeupSourceType WakeupSource`
	`)`	

Service ID[hex]:	0x30	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	WakeupSource	source (transceiver) which initiated the wake up event
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK when function has been successfully executed E_NOT_OK when function could not be successfully executed
Description:	Service is called by integration code to check a wakeup source.	

⌋ () [SWS_EthIf_00245] ⌈

The function EthIf_CheckWakeup shall forward the call to function EthTrcv_CheckWakeup of the respective Ethernet Transceiver Driver.⌋(SRS_Eth_00106)

[SWS_EthIf_00246] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00247] ⌈
If development error detection is enabled: the function shall check the parameter WakeupSource for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00248] ⌈
The function EthIf_CheckWakeup() shall be pre-compile time configurable On/Off by the configuration parameter EthIfWakeUpSupport.⌋()

[SWS_EthIf_00249] ⌈
Caveat: The function EthIf_CheckWakeup() requires previous transceiver initialization (EthIf_Init).⌋(SRS_Eth_00106)

8.3.7 EthIf_GetPhysAddr

[SWS_EthIf_00061] ⌈

Service name:	EthIf_GetPhysAddr	

Syntax:	`void EthIf_GetPhysAddr(`
	`    uint8 CtrlIdx,`
	`    uint8* PhysAddrPtr`
	`)`	

Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
Parameters (inout):	None	
Parameters (out):	PhysAddrPtr	Physical source address (MAC address) in network byte order.
Return value:	None	
Description:	Obtains the physical source address used by the indexed controller	

⌋ () [SWS_EthIf_00062] ⌈
The function EthIf_GetPhysAddr shall forward the call to the respective Ethernet Controller Driver.⌋()

[SWS_EthIf_00063] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00064] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00065] ⌈
If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00066] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.8 EthIf_SetPhysAddr

[SWS_EthIf_00132] ⌈

Service name:	EthIf_SetPhysAddr	

Syntax:	`void EthIf_SetPhysAddr(`
	`    uint8 CtrlIdx,`
	`    const uint8* PhysAddrPtr`
	`)`	

Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in):	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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Sets the physical source address used by the indexed controller.	

⌋ () [SWS_EthIf_00134] ⌈
The function EthIf_SetPhysAddr shall forward the call to the respective Ethernet Controller Driver.⌋()

[SWS_EthIf_00135] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00136] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00137] ⌈
If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00138] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.9 EthIf_UpdatePhysAddrFilter

[SWS_EthIf_00139] ⌈

Service name:	EthIf_UpdatePhysAddrFilter	

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

Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in):	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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: filter was successfully changed E_NOT_OK: filter could not be changed
Description:	Update the physical source address to/from the indexed controller filter. If the Ethernet Controller is not capable to do the filtering, the software has to do this.	

⌋ () [SWS_EthIf_00140] ⌈
The function EthIf_SetPhysAddrFilter shall forward the call to the respective Ethernet Controller Driver.⌋()

[SWS_EthIf_00141] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00142] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00143] ⌈
If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00144] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.10 EthIf_GetPortMacAddr

[SWS_EthIf_00190] ⌈

Service name:	EthIf_GetPortMacAddr	

Syntax:	`Std_ReturnType EthIf_GetPortMacAddr(`
	`    const uint8* MacAddrPtr,`
	`    uint8* SwitchIdxPtr,`
	`    uint8* PortIdxPtr`
	`)`	

Service ID[hex]:	0x28	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	MacAddrPtr	MAC-address for which a switch port is searched over which the node with this MAC-address can be reached.
Parameters (inout):	None	
Parameters (out):	SwitchIdxPtr	Pointer to the switch index
	PortIdxPtr	Pointer to the port index
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: switch port could not be initialized
Description:	Obtains the port over which this MAC-address can be reached	

⌋ () [SWS_EthIf_00191] ⌈
The function EthIf_GetPortMacAddr shall return the switch and port index over which the given MAC-address is reachable. If multiple or no ports are possible, this API call will return an error value. The API call will be forwarded to the Ethernet Switch Driver which shall have a corresponding API call.⌋()

[SWS_EthIf_00192] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGetPortMacAddrApi.⌋()

[SWS_EthIf_00193] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00194] ⌈
If development error detection is enabled: the function shall check the parameter MacAddrPtr, SwitchIdxPtr and PortIdxPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00195] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.11 EthIf_GetArlTable

[SWS_EthIf_00196] ⌈

Service name:	EthIf_GetArlTable

Syntax:	`Std_ReturnType EthIf_GetArlTable(`
	`    uint8 switchIdx,`
	`    uint16* numberOfElements,`
	`    Eth_MacVlanType* arlTableListPointer`

	)	
Service ID[hex]:	0x29	
Sync/Async:	Synchronous /Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	switchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout):	numberOfElements	In: Maximum number of elements which can be written into the arlTable Out: Number of elements which are currently available in the EthSwitch module.
Parameters (out):	arlTableListPointer	Returns a pointer to the memory where the ARL table of the switch consisting of a list of structs with MAC-address, VLAN-ID and port shall be stored.
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: requested switchIdx is not valid or inactive
Description:	Obtains the address resolution table of a switch and copies the list into a user provided buffer. The function will copy all or numberOfElements into the output list. If input value of numberOfElements is 0 the function will not copy any data but only return the number of valid entries in the cache. arlTableListPointer may be NULL_PTR in this case.	

⌋ ()

[SWS_EthIf_00197] ⌈
The function EthIf_GetArlTable shall return a list of structs with MAC-address, VLAN-ID and port for the indexed switch.⌋()

[SWS_EthIf_00254] ⌈
The function EthIf_GetArlTable shall forward the call to function EthSwt_GetArlTable of the respective Ethernet Switch Driver.⌋()

[SWS_EthIf_00198] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGetArlTable.⌋()

[SWS_EthIf_00199] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00200] ⌈
If development error detection is enabled: the function shall check the parameter ArlTable for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00201] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.12 EthIf_GetBufferLevel

[SWS_EthIf_00202] ⌈

Service name:	EthIf_GetBufferLevel

Syntax:	`Std_ReturnType EthIf_GetBufferLevel(`
	`    uint8 SwitchIdx,`

	uint32* SwitchBufferLevelPtr)	
Service ID[hex]:	0x2a	
Sync/Async:	Synchronous /Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout):	None	
Parameters (out):	SwitchBufferLevelPtr	The interpretation of this value is switch dependent
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: switch port could not be initialized
Description:	Reads the buffer level of the corresponding switch. Whether this buffer level is one value for the entire switch (shared memory) or one value for each port at a switch is technology dependent.	

⌋ ()

[SWS_EthIf_00203] ⌈
The function EthIf_GetBufferLevel shall read the buffer level of the currently used buffer of the switch.⌋()

[SWS_EthIf_00204] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGetBufferLevelApi.⌋()

[SWS_EthIf_00205] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00206] ⌈
If development error detection is enabled: the function shall check the parameter SwitchBufferLevelPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00207] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.13 EthIf_GetCtrlIdxList

[SWS_EthIf_91053] ⌈

Service name:	EthIf_GetCtrlIdxList	
Syntax:	Std_ReturnType EthIf_GetCtrlIdxList(uint8* NumberOfCtrlIdx, uint8* CtrlIdxListPtr)	
Service ID[hex]:	0x44	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	NumberOfCtrlIdx	in: maximum number of controllers in CtrlIdxListPtr, 0 to return the number of controllers but without filling CtrlIdxListPtr. out: number of active controllers.

Parameters (out):	CtrlIdxListPtr	List of active controller indexes
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: failure
Description:	Returns the number and index of all active Ethernet controllers.	

⌋ ()

[SWS_EthIf_00298]⌈
The optional EthIf_GetCtrlIdxList API shall return only the NumberOfCtrlIdx which are active.⌋ ()

[SWS_EthIf_00299]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋ ()

[SWS_EthIf_00300]⌈
If development error detection is enabled: the function shall check the OUT parameter CtrlIdxListPtr for being valid only if the the OUT parameter NumberOfCtrlIdx is greater 0x00. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

8.3.14 EthIf_GetVlanId

[SWS_EthIf_91052] ⌈

Service name:	EthIf_GetVlanId	

Syntax:	`Std_ReturnType EthIf_GetVlanId(`
	`    uint8 CtrlIdx,`
	`    uint16* VlanIdPtr`
	`)`	

Service ID[hex]:	0x43	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
Parameters (inout):	None	
Parameters (out):	VlanIdPtr	Pointer to store the VLAN identifier (VID) of the Ethernet controller. 0 if the the Ethernet controller represents no virtual network (VLAN).
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: failure
Description:	Returns the VLAN identifier of the requested Ethernet controller.	

⌋ ()

[SWS_EthIf_00301]⌈
The optional EthIf_GetVlanId API shall return the VlanId of the requested CtrlIdx.⌋ ()

[SWS_EthIf_00302]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋ ()

[SWS_EthIf_00303]⌈
If development error detection is enabled: the function shall check the parameter VlanId for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

8.3.15 EthIf_GetAndResetMeasurementData

[SWS_EthIf_91011] ⌈

Service name:	EthIf_GetAndResetMeasurementData	

Syntax:	`Std_ReturnType EthIf_GetAndResetMeasurementData(`
	`    EthIf_MeasurementIdxType MeasurementIdx,`
	`    boolean MeasurementResetNeeded,`
	`    uint32* MeasurementDataPtr`
	`)`	

Service ID[hex]:	0x45	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	MeasurementIdx	Data index of measurement data
	MeasurementResetNeeded	Flag to trigger a reset of the measurement data
Parameters (inout):	None	
Parameters (out):	MeasurementDataPtr	Reference to data buffer, where to copy measurement data
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Allows to read and reset detailed measurement data for diagnostic purposes. Get all MeasurementIdx's at once is not supported. ETHIF_MEAS_ALL shall only be used to reset all MeasurementIdx's at once. A NULL_PTR shall be provided for MeasurementDataPtr in this case.	

⌋ ()

[SWS_EthIf_00308]⌈
EthIf_GetAndResetMeasurementData shall return measurement data for selected measurement index.⌋ ()

[SWS_EthIf_00309]⌈
For measurement index ETHIF_MEAS_DROP_CRTLIDX the function shall return the number of all dropped datagrams, caused by invalid CrtlIdx/VLAN. If the VLAN is not enabled, all received VLAN tagged datagrams are invalid and shall be counted also.
⌋ ()

[SWS_EthIf_00310]⌈
The function shall return E_NOT_OK if the requested measurement index is not supported.⌋ ()

[SWS_EthIf_00312]⌈

The function shall reset all existing measurement data to 0, if MeasurementResetNeeded is true and measurement index is set to ETHIF_MEAS_ALL.⌋ ()

[SWS_EthIf_00313]⌈
All measurement data which counts data shall not overrun.⌋ ()

[SWS_EthIf_00314]⌈
The function shall accept NULL_PTR. In this case the measurement data shall not be copied.⌋ ()

[SWS_EthIf_00316]⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGetAndResetMeasurementDataApi.⌋ ()

[SWS_EthIf_00317]⌈
If the VLAN is not active the Ethernet Interface shall increment the corresponding measurement data and filter the message.⌋ ()

[SWS_EthIf_00319]⌈
If development error detection is enabled: The function shall check that the service EthIf_Init () was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOTINIT and return E_NOT_OK.⌋ ()

8.3.16 EthIf_StoreConfiguration

[SWS_EthIf_00214] ⌈

Service name:	EthIf_StoreConfiguration	

Syntax:	`Std_ReturnType EthIf_StoreConfiguration(`
	`    uint8 SwitchIdx`
	`)`	

Service ID[hex]:	0x2c	
Sync/Async:	Synchronous /Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: switch port could not be initialized or unknown index
Description:	Stores the configuration of the learned MAC/Port tables of a switch in a persistent manner and will be used by e.g. CDD.	

⌋ ()
[SWS_EthIf_00215] ⌈
The function EthIf_StoreConfiguration shall read a list of values of the switch.⌋()

[SWS_EthIf_00216] ⌈

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

[SWS_EthIf_00217] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00218] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.17 EthIf_ResetConfiguration

[SWS_EthIf_00219] ⌈

Service name:	EthIf_ResetConfiguration	

Syntax:	`Std_ReturnType EthIf_ResetConfiguration(`
	`    uint8 SwitchIdx`
	`)`	

Service ID[hex]:	0x2d	
Sync/Async:	Synchronous /Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: switch port could not be initialized
Description:	Resets the configuration of the learned MAC/Port tables of a switch in a persistent manner and will be used by e.g. CDD. The statically configured entries shall still remain.	

⌋ ()

[SWS_EthIf_00220] ⌈
The function EthIf_ResetConfiguration shall read a list of values of the switch.⌋()

[SWS_EthIf_00221] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfResetConfigurationApi.⌋()

[SWS_EthIf_00222] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00223] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.18 EthIf_GetCurrentTime

[SWS_EthIf_00154] ⌈

Service name:	EthIf_GetCurrentTime	

Syntax:	`Std_ReturnType EthIf_GetCurrentTime(`
	`    uint8 CtrlIdx,`
	`    Eth_TimeStampQualType* timeQualPtr,`
	`    Eth_TimeStampType* timeStampPtr`
	`)`	

Service ID[hex]:	0x22	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the addresses ETH controller.
Parameters (inout):	None	
Parameters (out):	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Returns a time value out of the HW registers according to the capability of the HW. Is the HW resolution is lower than the Eth_TimeStampType resolution resp. range, the remaining bits will be filled with 0.	

⌋ ()

[SWS_EthIf_00155] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00156] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00157] ⌈
If development error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00158] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGlobalTimeSupport.⌋()

[SWS_EthIf_00159] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.19 EthIf_EnableEgressTimeStamp

[SWS_EthIf_00160] ⌈

Service name:	EthIf_EnableEgressTimeStamp

Syntax:	`void EthIf_EnableEgressTimeStamp(`
	`    uint8 CtrlIdx,`
	`    Eth_BufIdxType BufIdx`
	`)`

Service ID[hex]:	0x23
Sync/Async:	Synchronous

Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the addresses ETH controller.
	BufIdx	Index of the message buffer, where Application expects egress time stamping
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Activates egress time stamping on a dedicated message object. Some HW does store once the egress time stamp marker and some HW needs it always before transmission. There will be no "disable" functionality, due to the fact, that the message type is always "time stamped" by network design.	

⌋ ()

[SWS_EthIf_00161] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00162] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00164] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGlobalTimeSupport.⌋()

[SWS_EthIf_00165] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.20 EthIf_GetEgressTimeStamp

[SWS_EthIf_00166] ⌈

Service name:	EthIf_GetEgressTimeStamp	

Syntax:	`Std_ReturnType EthIf_GetEgressTimeStamp(`
	`    uint8 CtrlIdx,`
	`    Eth_BufIdxType BufIdx,`
	`    Eth_TimeStampQualType* timeQualPtr,`
	`    Eth_TimeStampType* timeStampPtr`
	`)`	

Service ID[hex]:	0x24	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the address ETH controller.
	BufIdx	Index of the message buffer, where the Upper Layer expects egress time stamping
Parameters (inout):	None	
Parameters (out):	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Return value:	Std_ReturnType	--
Description:	Reads back the egress time stamp on a dedicated message object. It must be called within the TxConfirmation() function.	

⌋ ()
[SWS_EthIf_00167] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00168] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00169] ⌈
If development error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00170] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGlobalTimeSupport.⌋()

[SWS_EthIf_00171] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.21 EthIf_GetIngressTimeStamp

[SWS_EthIf_00172] ⌈

Service name:	EthIf_GetIngressTimeStamp	

Syntax:	`Std_ReturnType EthIf_GetIngressTimeStamp(`
	`    uint8 CtrlIdx,`
	`    Eth_DataType* DataPtr,`
	`    Eth_TimeStampQualType* timeQualPtr,`
	`    Eth_TimeStampType* timeStampPtr`
	`)`	

Service ID[hex]:	0x25	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the addresses ETH controller.
	DataPtr	Pointer to the message buffer, where Application expects ingress time stamping
Parameters (inout):	None	
Parameters (out):	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Return value:	Std_ReturnType	--
Description:	Reads back the ingress time stamp on a dedicated message object. It must be called within the RxIndication() function.	

⌋ ()
[SWS_EthIf_00173] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00174] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00175] ⌈
If development error detection is enabled: the function shall check the parameter DataPtr, timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00176] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGlobalTimeSupport.⌋()

[SWS_EthIf_00177] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.22 EthIf_SwitchPortGroupRequestMode

[SWS_EthIf_91102] ⌈

Service name:	EthIf_SwitchPortGroupRequestMode	

Syntax:	`Std_ReturnType EthIf_SwitchPortGroupRequestMode(`
	`    EthIf_SwitchPortGroupIdxType PortGroupIdx,`
	`    EthTrcv_ModeType PortMode`
	`)`	

Service ID[hex]:	0x06	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	PortGroupIdx	Index of the port group within the context of the Ethernet Interface
	PortMode	ETHTRCV_MODE_DOWN: disable the port group ETHTRCV_MODE_ACTIVE: enable the port group
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: port group mode could not be changed
Description:	Request a mode for the EthIfSwtPortGroup. The call shall be forwarded to EthSwt by calling EthSwt_SetSwitchPortMode for all EthSwtPorts referenced by the port group.	

⌋ ()

[SWS_EthIf_00270]⌈
If EthIf_SwitchPortGroupRequestMode is called with ETHTRCV_MODE_DOWN EthIf shall start a timer with EthIfSwitchOffPortTimedelay for all ports of the respective EthIf_SwitchPortGroup if the mode ETHTRCV_MODE_DOWN has been requested for all EthIfSwitchPortGroups referencing the port and the current mode is ETHTRCV_MODE_ACTIVE.⌋ ()

[SWS_EthIf_00271]⌈

If the timer to switch off ports (see SWS_EthIf_00270) elapses for a port EthIf shall call EthSwt_SetSwitchPortMode with ETHTRCV_MODE_DOWN for the corresponding EthSwtPort.⌋ ()

Rationale: "Delaying with EthIfSwitchOffPortTimedelay is needed to ensure that if port is connected to an ECU without switch this ECU has shut down its transceiver and if port is connected to a port of a further switch that this port is shutdown at a similar point in time.

Rationale: The implementation has to ensure that EthSwtPorts within EthIfSwitchPortGroups are only disabled if all prior activation requests have been withdrawn. This could be realized e.g. by a counter mechanism.

[SWS_EthIf_00272]⌈
If EthIf_SwitchPortGroupRequestMode is called with ETHTRCV_MODE_ACTIVE, EthIf shall forward the call to function EthSwt_SetSwitchPortMode for all EthSwtPorts of the respective EthIfSwitchPortGroup if the requested mode and the current mode are different.⌋ ()

[SWS_EthIf_00273]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED and return E_NOT_OK.⌋ ()

[SWS_EthIf_00274]⌈
If development error detection is enabled: the function shall check that the provided parameter PortGroupIdx addresses a port group not referenced by any EthIfController. If the check fails, the function shall raise the development error ETHIF_E_INV_PORT_GROUP_IDX and return E_NOT_OK.⌋ ()

Rationale: Avoid that a EthIfSwitchPortGroup which shall be controlled by EthIfController is incidentally called by BswM

Note: The function requires previous initialization (EthIf_Init).

8.3.23 EthIf_StartAllPorts

[SWS_EthIf_91103] ⌈

Service name:	EthIf_StartAllPorts	

Syntax:	`Std_ReturnType EthIf_StartAllPorts(`
	`    void`
	`)`	

Service ID[hex]:	0x07	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success

		E_NOT_OK: port mode could not be started
Description:	Request to start all configured ports	

⌋ ()

[SWS_EthIf_00275]⌈
If EthIf_StartAllPorts is called, EthIf shall set the mode to ETHTRCV_MODE_ACTIVE of all ports which are not in a port group referenced by EthIfController and start a timer with EthIfPortStartupActiveTime for all this ports.⌋ ()

[SWS_EthIf_00276]⌈
If the timer to switch off all ports (see SWS_EthIf_00275) elapses EthIf shall call EthSwt_SetSwitchPortMode with ETHTRCV_MODE_DOWN for EthSwtPort which are not requested with ETHTRCV_MODE_ACTIVE via EthIf_SwitchPortGroupRequestMode.⌋ ()

Rationale: "Delaying with EthIfPortStartupActiveTime is needed to ensure that NM messages with PNC information are received and the requested PNCs are activated.

Note: EthIf_StartAllPorts could be called in context of BswM_EcuM_CurrentWakeup. After a wakeup occurred on the wakeup line all EthIfPortGroups shall be activated to enable communication stack to receive NM messages (PNC information). With this it is possible to start the EthIfSwitchPortGroups without starting a PNC.

[SWS_EthIf_00277]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED and return E_NOT_OK.⌋ ()

8.3.24 EthIf_SetSwitchMgmtInfo

[SWS_EthIf_91003] ⌈

Service name:	EthIf_SetSwitchMgmtInfo	

Syntax:	`Std_ReturnType EthIf_SetSwitchMgmtInfo(`
	`    uint8 CtrlIdx,`
	`    Eth_BufIdxType BufIdx,`
	`    EthSwt_MgmtInfoType* MgmtInfoPtr`
	`)`	

Service ID[hex]:	0x38	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of an Ethernet Interface controller
	BufIdx	Ethernet Tx Buffer index
	MgmtInfoPtr	Pointer to the management information
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Management infos successfully set E_NOT_OK: Setting of management infos failed
Description:	Provides additional management information along to an Ethernet frame that requires special treatment within the Switch. It has to be called between EthIf_ProvideTxBuffer() and EthIf_Transmit() of the related frame.	

⌋ (SRS_Eth_00125)

[SWS_EthIf_00279]⌈
The function shall be pre compile time configurable ON/OFF by the configuration parameter: EthIfSwitchManagementSupport.⌋ ()

[SWS_EthIf_00280]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init() was previously called.
If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋ ()

[SWS_EthIf_00281]⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid.
If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00282]⌈
If development error detection is enabled: the function shall check the parameter BufIdx for being valid.
If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM.⌋()

[SWS_EthIf_00283]⌈
If development error detection is enabled: the function shall check the parameter MgmtInfoPtr for being valid.
If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00284]⌈
Caveat: The function requires previous Switch initialization (EthSwt_Init()).⌋ ()

8.3.25 EthIf_SwitchEnableTimeStamping

[SWS_EthIf_91007] ⌈

Service name:	EthIf_SwitchEnableTimeStamping	

Syntax:	`Std_ReturnType EthIf_SwitchEnableTimeStamping(`
	`    uint8 CtrlIdx,`
	`    Eth_BufIdxType BufIdx,`
	`    EthSwt_MgmtInfoType* MgmtInfo`
	`)`	

Service ID[hex]:	0x39	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	BufIdx	Index of the message buffer, where Application expects egress time stamping
Parameters (inout):	None	
Parameters (out):	MgmtInfo	Management information

Return value:	Std_ReturnType	E_OK: Time stamping on egress successfully enabled E_NOT_OK: Enabling of time stamping on egress has been failed
Description:	Activates egress time stamping on a dedicated message object, addressed by CtrlIdx and BufIdx.	

⌋ (SRS_Eth_00125)

[SWS_EthIf_00387]⌈
If EthIf_SwitchEnableTimeStamping is called, the EthIf shall call EthSwt_EnableTimeStamping, if the controller index refers to an EthSwt_Config. If the CtrlIdx refers to EthIfSwitchPortGroup, the EthIf shall call the API EthSwt_PortEnableTimeStamp for every port in the group.⌋ ()

[SWS_EthIf_00285]⌈
The function shall be pre compile time configurable ON/OFF by the configuration parameter: EthIfGlobalTimeSupport.⌋ ()

[SWS_EthIf_00286]⌈
If development error detection is enabled: the function shall check that the service Eth_Init() was previously called.
If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋ ()

[SWS_EthIf_00287]⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid.
If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋ ()

[SWS_EthIf_00288]⌈
If development error detection is enabled: the function shall check the parameter BufIdx for being valid.
If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM.⌋ ()

[SWS_EthIf_00289]⌈
If development error detection is enabled: the function shall check the parameter BufIdx for being valid.
If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM.⌋ ()

[SWS_EthIf_00290]⌈
If development error detection is enabled: the function shall check the parameter BufIdx for being valid.
If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM.⌋ ()

8.3.26 EthIf_VerifyConfig

[SWS_EthIf_91012] ⌈

Service name:	EthIf_VerifyConfig	
Syntax:	Std_ReturnType EthIf_VerifyConfig(uint8 SwitchIdx, boolean* Result)	
Service ID[hex]:	0x40	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout):	None	
Parameters (out):	Result	Result of verification, TRUE: configureation verified ok, FALSE: configuraton values found corrupted
Return value:	Std_ReturnType	E_OK: Configuration verificaton succeeded, E_NOT_OK: Configuration verification not succeeded.
Description:	Forwarded to EthSwt_VerifyConfig. EthSwt_VerifyConfig verifies the Switch Configuration depending on the HW-Architecture, HW-capability and the intended accuracy of this verification.	

⌋ ()

[SWS_EthIf_00304]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋
(SRS_BSW_00101)(SRS_BSW_00369)

[SWS_EthIf_00305]⌈
The function shall be compile time configurable On/Off by the configuration parameter: EthIfVerifyConfigApi.⌋ ()

8.3.27 EthIf_SetForwardingMode

[SWS_EthIf_91013] ⌈

Service name:	EthIf_SetForwardingMode	
Syntax:	Std_ReturnType EthIf_SetForwardingMode(uint8 SwitchIdx, boolean mode)	
Service ID[hex]:	0x41	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	mode	True Forwarding enabled, False Forwarding disabled
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: stopping of frame forwarding succeeded, E_NOT_OK: stopping of frame forwarding not succeeded.

Description:	Verifies the Switch Configuration. If Configuration is not valid, Switch is reconfigured.

⌋ ()

[SWS_EthIf_00306]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋ (SRS_BSW_00101)(SRS_BSW_00369)

[SWS_EthIf_00307]⌈
The function shall be compile time configurable On/Off by the configuration parameter: EthIfSetForwardingModeApi.⌋ ()

8.3.28 EthIf_GetPhySignalQuality

[SWS_EthIf_91019] ⌈

Service name:	EthIf_GetPhySignalQuality	

Syntax:	`Std_ReturnType EthIf_GetPhySignalQuality(`
	`    uint8 TrcvIdx,`
	`    uint8* SignalQualityPtr`
	`)`	

Service ID[hex]:	0x16	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in):	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
Parameters (inout):	None	
Parameters (out):	SignalQualityPtr	Pointer to the memory where the signal quality in percent shall be stored.
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted.
Description:	Obtains the current signal quality of the link of the indexed transceiver	

⌋ (SRS_Eth_00117)

[SWS_EthIf_00320]⌈
The function EthIf_GetPhySignalQuality shall forward the call to function EthTrcv_GetPhySignalQuality of the corresponding Ethernet Transceiver Driver (EthIfTransceiverIdx).⌋ ()

[SWS_EthIf_00321]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00322]⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development

error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00323]⌈
If development error detection is enabled: the function shall check the parameter SignalQualityPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

8.3.29 EthIf_SetPhyTestMode

[SWS_EthIf_91016] ⌈

Service name:	EthIf_SetPhyTestMode	

Syntax:	`Std_ReturnType EthIf_SetPhyTestMode(`
	`    uint8 TrcvIdx,`
	`    EthTrcv_PhyTestModeType Mode`
	`)`	

Service ID[hex]:	0x17	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in):	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
	Mode	Test mode to be activated
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted.
Description:	Activates a given test mode.	

⌋ (SRS_Eth_00117)

[SWS_EthIf_00324]⌈
The function EthIf_SetPhyTestMode shall forward the call to function EthTrcv_SetPhyTestMode of the corresponding Ethernet Transceiver Driver (EthIfTransceiverIdx).⌋ ()

[SWS_EthIf_00325]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00326]⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

8.3.30 EthIf_SetPhyLoopbackMode

[SWS_EthIf_91018] ⌈

Service name:	EthIf_SetPhyLoopbackMode	

Syntax:	`Std_ReturnType EthIf_SetPhyLoopbackMode(`
	`    uint8 TrcvIdx,`
	`    EthTrcv_PhyLoopbackModeType Mode`
	`)`	

Service ID[hex]:	0x12	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in):	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
	Mode	Loopback mode to be activated
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted.
Description:	Activates a given loopback mode.	

⌋ (SRS_Eth_00117)

[SWS_EthIf_00327]⌈
The function EthIf_SetPhyLoopbackMode shall forward the call to function EthTrcv_SetPhyLoopbackMode of the corresponding Ethernet Transceiver Driver (EthIfTransceiverIdx).⌋ ()

[SWS_EthIf_00328]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00329]⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

8.3.31 EthIf_SetPhyTxMode

[SWS_EthIf_91005] ⌈

Service name:	EthIf_SetPhyTxMode

Syntax:	`Std_ReturnType EthIf_SetPhyTxMode(`
	`    uint8 TrcvIdx,`
	`    EthTrcv_PhyTxModeType Mode`
	`)`

Service ID[hex]:	0x13
Sync/Async:	Synchronous
Reentrancy:	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.

Parameters (in):	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
	Mode	Transmission mode to be activated
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
Description:	Activates a given transmission mode.	

⌋ (SRS_Eth_00117)

[SWS_EthIf_00327]⌈
The function EthIf_SetPhyTxMode shall forward the call to function EthTrcv_SetPhyTxMode of the corresponding Ethernet Transceiver Driver (EthIfTransceiverIdx).⌋ ()

[SWS_EthIf_00328]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00329]⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

8.3.32 EthIf_GetCableDiagnosticsResult

[SWS_EthIf_91014] ⌈

Service name:	EthIf_GetCableDiagnosticsResult	

Syntax:	`Std_ReturnType EthIf_GetCableDiagnosticsResult(`
	`    uint8 TrcvIdx,`
	`    EthTrcv_CableDiagResultType* ResultPtr`
	`)`	

Service ID[hex]:	0x14	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in):	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
Parameters (inout):	None	
Parameters (out):	ResultPtr	Pointer to the location where the cable diagnostics result shall be stored
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
Description:	Retrieves the cable diagnostics result of a given transceiver.	

⌋ (SRS_Eth_00117)

[SWS_EthIf_00330]⌈
The function EthIf_GetCableDiagnosticsResult shall forward the call to function EthTrcv_GetCableDiagnosticsResult of the corresponding Ethernet Transceiver Driver (EthIfTransceiverIdx).⌋ ()

[SWS_EthIf_00331]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00332][⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00333]⌈
If development error detection is enabled: the function shall check the parameter ResultPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

8.3.33 EthIf_GetPhyIdentifier

[SWS_EthIf_91020] ⌈

Service name:	EthIf_GetPhyIdentifier	

Syntax:	`Std_ReturnType EthIf_GetPhyIdentifier(`
	`    uint8 TrcvIdx,`
	`    uint32* OrgUniqueIdPtr,`
	`    uint8* ModelNrPtr,`
	`    uint8* RevisionNrPtr`
	`)`	

Service ID[hex]:	0x15	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in):	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
Parameters (inout):	None	
Parameters (out):	OrgUniqueIdPtr	Pointer to the memory where the Organizationally Unique Identifier shall be stored.
	ModelNrPtr	Pointer to the memory where the Manufacturer's Model Number shall be stored.
	RevisionNrPtr	Pointer to the memory where the Revision Number shall be stored.
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
Description:	Obtains the PHY identifier of the Ethernet Interface according to IEEE 802.3-2015 chapter 22.2.4.3.1 PHY Identifer.	

⌋ (SRS_Eth_00117)

[SWS_EthIf_00334]⌈
The function EthIf_GetPhyIdentifier shall forward the call to function EthTrcv_GetPhyIdentifier of the corresponding Ethernet Transceiver Driver (EthIfTransceiverIdx).⌋ ()

[SWS_EthIf_00335]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00336]⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00337]⌈
If development error detection is enabled: the function shall check the parameter OrgUniqueIdPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00338]⌈
If development error detection is enabled: the function shall check the parameter ModelNrPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00339]⌈
If development error detection is enabled: the function shall check the parameter RevisionNrPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

8.3.34 EthIf_GetBufWRxParams

[SWS_EthIf_91002] ⌈

Service name:	EthIf_GetBufWRxParams

Syntax:	`Std_ReturnType EthIf_GetBufWRxParams(`
	`    uint8 CtrlIdx,`
	`    const WEth_BufWRxParamIdType* RxParamIds,`
	`    uint32* ParamValues,`
	`    uint8 NumParams`
	`)`

Service ID[hex]:	0x32
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	RxParamIds	IDs of the Parameters to read
	NumParams	Number of Parameters
Parameters (inout):	None	
Parameters (out):	ParamValues	Values of the Parameters requested
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: failed reading parameters
Description:	Read out values related to the receive direction of the transceiver for a received packet. For example, this could be RSSI or Channel belonging to one single packet.	

⌋ ()

[SWS_EthIf_00341]⌈
The function EthIf_GetBufWRxParams shall forward the call to function WEth_GetBufWRxParams of the respective Wireless Ethernet Controller Driver.⌋ ()

[SWS_EthIf_00342]⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfEnableWEthApi.⌋ ()

[SWS_EthIf_00343]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋ ()

[SWS_EthIf_00344]⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00345]⌈
If development error detection is enabled: the function shall check the parameter RxParamIds for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

[SWS_EthIf_00346]⌈
If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

Note: The function requires previous reception (EthIf_RxIndication).

8.3.35 EthIf_GetBufWTxParams

[SWS_EthIf_91009] ⌈

Service name:	EthIf_GetBufWTxParams

Syntax:	`Std_ReturnType EthIf_GetBufWTxParams(`
	`    uint8 CtrlIdx,`
	`    const WEth_BufWTxParamIdType* TxParamIds,`

	uint32* ParamValues, uint8 NumParams)	
Service ID[hex]:	0x31	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	TxParamIds	IDs of the Parameter that are requested
	NumParams	Number of Parameters that are requested
Parameters (inout):	None	
Parameters (out):	ParamValues	Values of the Parameters requested
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: failed reading parameters
Description:	Read out values related to the transmit direction of the transceiver for a transmitted packet. For example, this could be transaction ID belonging to one single packet.	

⌋ ()

[SWS_EthIf_00347]⌈
The function EthIf_GetBufWTxParams shall forward the call to function WEth_GetBufWTxParams of the respective Wireless Ethernet Controller Driver.⌋ ()

[SWS_EthIf_00348]⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfEnableWEthApi.⌋ ()

[SWS_EthIf_00349]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋ ()

[SWS_EthIf_00350]⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00351]⌈
If development error detection is enabled: the function shall check the parameter TxParamIds for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

[SWS_EthIf_00352]⌈
If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

Note: The function requires previous transmission (EthIf_Transmit).

8.3.36 EthIf_SetBufWTxParams

[SWS_EthIf_91017] ⌈

Service name:	EthIf_SetBufWTxParams	

Syntax:	`Std_ReturnType EthIf_SetBufWTxParams(`
	`    uint8 CtrlIdx,`
	`    Eth_BufIdxType BufIdx,`
	`    const WEth_BufWTxParamIdType* TxParamIds,`
	`    const uint32* ParamValues,`
	`    uint8 NumParams`
	`)`	

Service ID[hex]:	0x33	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	BufIdx	Index of the buffer resource
	TxParamIds	IDs of the Parameter that are provided to the transmit radio
	ParamValues	Values of the Parameters that are provided to the transmit radio
	NumParams	Number of Parameters that are provided to the transmit radio
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: failed setting parameter
Description:	Set values related to the transmit direction of the transceiver for a specific buffer (packet to be sent). For example, this can be the desired transmit power or the channel belonging to one single packet.	

⌋ ()

[SWS_EthIf_00353]⌈
The function EthIf_SetBufWTxParams shall forward the call to function WEth_SetBufWTxParams of the respective Wireless Ethernet Controller Driver.⌋ ()

[SWS_EthIf_00354]⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfEnableWEthApi.⌋ ()

[SWS_EthIf_00355]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋ ()

[SWS_EthIf_00356]⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00357]⌈

If development error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00358]⌈
If development error detection is enabled: the function shall check the parameter TxParamIds for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

[SWS_EthIf_00359]⌈
If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

Note: The function requires previous buffer request (EthIf_ProvideTxBuffer).

8.3.37 EthIf_SetRadioParams

[SWS_EthIf_91026] ⌈

Service name:	EthIf_SetRadioParams	

Syntax:	`Std_ReturnType EthIf_SetRadioParams(`
	`    uint8 TrcvId,`
	`    const WEthTrcv_SetRadioParamIdType* ParamIds,`
	`    const uint32* ParamValue,`
	`    uint8 NumParams`
	`)`	

Service ID[hex]:	0x34	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TrcvId	Index of the transceiver
	ParamIds	IDs of the Parameters to set
	ParamValue	Values of the Parameters to set
	NumParams	Number of Parameters to set
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: failed writing parameters
Description:	Set values related to a transceiver's wireless radio. For example, this could be the selection of the radio settings (channel, ...).	

⌋ ()

[SWS_EthIf_00360]⌈
The function EthIf_SetRadioParams shall forward the call to function WEthTrcv_SetRadioParams of the respective Wireless Ethernet Transceiver Driver.
⌋ ()

[SWS_EthIf_00361]⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfEnableWEthApi.⌋ ()

[SWS_EthIf_00362]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋ ()

[SWS_EthIf_00363]⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00364]⌈
If development error detection is enabled: the function shall check the parameter ParamIds for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

[SWS_EthIf_00365]⌈
If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

8.3.38 EthIf_SetChanRxParams

[SWS_EthIf_91034] ⌈

Service name:	EthIf_SetChanRxParams	

Syntax:	`Std_ReturnType EthIf_SetChanRxParams(`
	`uint8 TrcvId,`
	`uint8 RadioId,`
	`const WEthTrcv_SetChanRxParamIdType* ParamIds,`
	`const uint32* ParamValues,`
	`uint8 NumParams`
	`)`	

Service ID[hex]:	0x35	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TrcvId	Index of the transceiver
	RadioId	Index of the Transceiver's Radio (including channel)
	ParamIds	IDs of the Parameters to set
	ParamValues	Values of the Parameters to set
	NumParams	Number of Parameters to set
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: failed writing parameters
Description:	Set values related to the receive direction of a transceiver's wireless channel. For example, this could be a channel parameter like the frequency.	

⌋ ()

[SWS_EthIf_00366]⌈

The function EthIf_SetChanRxParams shall forward the call to function WEthTrcv_SetChanRxParams of the respective Wireless Ethernet Transceiver Driver.⌋ ()

[SWS_EthIf_00367]⌈
The function EthIf_SetChanRxParams shall be pre compile time configurable On/Off by the configuration parameter: EthIfEnableWEthApi.⌋ ()

[SWS_EthIf_00368]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋ ()

[SWS_EthIf_00369]⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00370]⌈
If development error detection is enabled: the function shall check the parameter RadioIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00371]⌈
If development error detection is enabled: the function shall check the parameter RxParamIds for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

[SWS_EthIf_00372]⌈
If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

8.3.39 EthIf_SetChanTxParams

[SWS_EthIf_91042] ⌈

Service name:	EthIf_SetChanTxParams	

Syntax:	`Std_ReturnType EthIf_SetChanTxParams(`
	`    uint8 TrcvId,`
	`    uint8 RadioId,`
	`    const WEthTrcv_SetChanTxParamIdType* TxParamIds,`
	`    const uint32* ParamValues,`
	`    uint8 NumParams`
	`)`	

Service ID[hex]:	0x36	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TrcvId	Index of the transceiver

	RadioId	Index of the Transceiver's Radio (including channel)
	TxParamIds	IDs of the Parameters to set
	ParamValues	Values of the Parameters to set
	NumParams	Number of Parameters to set
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: failed writing parameters
Description:	Set values related to the transmit direction of a transceiver's wireless channel. For example, this could be the bitrate of a channel.	

⌋ ()

[SWS_EthIf_00373]⌈
The function EthIf_SetChanTxParams shall forward the call to function WEthTrcv_SetChanTxParams of the respective Wireless Ethernet Transceiver Driver.⌋ ()

[SWS_EthIf_00374]⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfEnableWEthApi.⌋ ()

[SWS_EthIf_00375]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋ ()

[SWS_EthIf_00376]⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00377]⌈
If development error detection is enabled: the function shall check the parameter RadioIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00378]⌈
If development error detection is enabled: the function shall check the parameter TxParamIds for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

[SWS_EthIf_00379]⌈
If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

8.3.40 EthIf_GetChanRxParams

[SWS_EthIf_91050] ⌈

Service name:	EthIf_GetChanRxParams	

Syntax:	`Std_ReturnType EthIf_GetChanRxParams(`
	`    uint8 TrcvId,`
	`    uint8 RadioId,`
	`    const WEthTrcv_GetChanRxParamIdType* ParamIds,`
	`    uint32* ParamValues,`
	`    uint8 NumParams`
	`)`	

Service ID[hex]:	0x37	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TrcvId	Index of the transceiver
	RadioId	Index of the Transceiver's Radio (including channel)
	ParamIds	IDs of the Parameters to read
	NumParams	Number of Parameters to read
Parameters (inout):	None	
Parameters (out):	ParamValues	Values of the requested Parameters
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: failed reading parameters
Description:	Read values related to the receive direction of the transceiver. For example, this could be a Channel Busy Ratio (CBR) or the average Channel Idle Time (CIT).	

⌋ ()

[SWS_EthIf_00380]⌈
The function EthIf_GetChanRxParams shall forward the call to function WEthTrcv_GetChanRxParams of the respective Wireless Ethernet Transceiver Driver.⌋ ()

[SWS_EthIf_00381]⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfEnableWEthApi.⌋ ()

[SWS_EthIf_00382]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋ ()

[SWS_EthIf_00383]⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00384]⌈
If development error detection is enabled: the function shall check the parameter RadioIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM otherwise (if DET is disabled) return E_NOT_OK.⌋ ()

[SWS_EthIf_00385]⌈
If development error detection is enabled: the function shall check the parameter RxParamIds for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

[SWS_EthIf_00386]⌈
If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋ ()

8.3.41 EthIf_ProvideTxBuffer

[SWS_EthIf_00067] ⌈

Service name:	EthIf_ProvideTxBuffer	

Syntax:	`BufReq_ReturnType EthIf_ProvideTxBuffer(`
	`uint8 CtrlIdx,`
	`Eth_FrameType FrameType,`
	`uint8 Priority,`
	`Eth_BufIdxType* BufIdxPtr,`
	`uint8** BufPtr,`
	`uint16* LenBytePtr`
	`)`	

Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	FrameType	Ethernet Frame Type (EtherType)
	Priority	Priority value which shall be used for the 3-bit PCP field of the VLAN tag
Parameters (inout):	LenBytePtr	in: desired length in bytes, out: granted length in bytes
Parameters (out):	BufIdxPtr	Index to the granted buffer resource. To be used for subsequent requests
	BufPtr	Pointer to the granted buffer
Return value:	BufReq_ReturnType	BUFREQ_OK: success BUFREQ_E_NOT_OK: development error detected BUFREQ_E_BUSY: all buffers in use
Description:	Provides access to a transmit buffer of the specified Ethernet controller.	

⌋ ()

[SWS_EthIf_00146] ⌈
If CtrlIdx refers to an EthIfCtrl where no EthIfVlanID is configured, the parameters FrameType and Priority are not used.⌋()

[SWS_EthIf_00147] ⌈
If VLAN is used

- EthIf shall increment the input desired length by 4 bytes before calling the Ethernet Driver module
- EthIf shall store the PCP (Priority parameter), CFI (always 0), VID (configured VLAN ID) and value of the FrameType parameter at the beginning of the buffer received from Eth_ProvideTxBuffer).
- EthIf shall increment the BufPtr by 4 bytes when returning the granted buffer

- EthIf shall decrement the output granted length by 4 bytes⌋()

[SWS_EthIf_00068] ⌈
The function EthIf_ProvideTxBuffer shall forward the call to the respective Ethernet Controller Driver.⌋()

[SWS_EthIf_00069] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED and return BUFREQ_E_NOT_OK.⌋()

[SWS_EthIf_00070] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX and return BUFREQ_E_NOT_OK.⌋()

[SWS_EthIf_00071] ⌈
If development error detection is enabled: the function shall check the parameter BufIdxPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER and return BUFREQ_E_NOT_OK.⌋()

[SWS_EthIf_00072] ⌈
If development error detection is enabled: the function shall check the parameter BufPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER and return BUFREQ_E_NOT_OK.⌋()

[SWS_EthIf_00073] ⌈
If development error detection is enabled: the function shall check the parameter LenBytePtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER and return BUFREQ_E_NOT_OK.⌋()

[SWS_EthIf_00074] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.42 EthIf_Transmit

[SWS_EthIf_00075] ⌈

Service name:	EthIf_Transmit	

Syntax:	`Std_ReturnType EthIf_Transmit(`
	`    uint8 CtrlIdx,`
	`    Eth_BufIdxType* BufIdx,`
	`    Eth_FrameType FrameType,`
	`    boolean TxConfirmation,`
	`    uint16* LenByte,`
	`    const uint8* PhysAddrPtr`
	`)`	

Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different buffer indexes and Ctrl indexes	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	FrameType	Ethernet frame type
	TxConfirmation	Activates transmission confirmation
	PhysAddrPtr	Physical target address (MAC address) in network byte order
Parameters (inout):	LenByte	Data length in byte
Parameters (out):	BufIdx	Index of the buffer resource
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: transmission failed
Description:	Triggers transmission of a previously filled transmit buffer	

⌋ () [SWS_EthIf_00250] ⌈
If CtrlIdx refers to an EthIfCtrl where an EthIfVlanID is configured, the parameters FrameType is not used, and 0x8100 is provided to Eth_Transmit instead.⌋()

[SWS_EthIf_00076] ⌈
The function EthIf_Transmit shall forward the call to the respective Ethernet Controller Driver.⌋()

[SWS_EthIf_00077] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00078] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00079] ⌈
If development error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00080] ⌈
If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00081] ⌈
Caveat: The function requires previous buffer request (EthIf_ProvideTxBuffer).⌋()

8.3.43 EthIf_GetVersionInfo

[SWS_EthIf_00082] ⌈

Service name:	EthIf_GetVersionInfo	

Syntax:	`void EthIf_GetVersionInfo(`
	`    Std_VersionInfoType* VersionInfoPtr`
	`)`	

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	VersionInfoPtr	Version information of this module
Return value:	None	
Description:	Returns the version information of this module	

⌋ ()

[SWS_EthIf_00127] ⌈
If development error detection is enabled: the function shall check the parameter VersionInfoPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

8.4 Callback notifications

This is a list of functions provided for other modules. File EthIf_Cbk.h shall provide the function prototypes of the callback functions.

8.4.1 EthIf_RxIndication

[SWS_EthIf_00085] ⌈

Service name:	EthIf_RxIndication

Syntax:	`void EthIf_RxIndication(`
	`    uint8 CtrlIdx,`
	`    Eth_FrameType FrameType,`
	`    boolean IsBroadcast,`
	`    const uint8* PhysAddrPtr,`
	`    Eth_DataType* DataPtr,`
	`    uint16 LenByte`
	`)`

Service ID[hex]:	0x10
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	CtrlIdx	Index of the physical Ethernet controller within the context of the Ethernet Interface
	FrameType	Frame type of received Ethernet frame
	IsBroadcast	parameter to indicate a broadcast frame
	PhysAddrPtr	Pointer to Physical source address (MAC address in network byte order) of received Ethernet frame
	DataPtr	Pointer to payload of received Ethernet frame.
	LenByte	Length of the received frame bytes
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Handles a received frame received by the indexed controller	

⌋ ()

[SWS_EthIf_00086] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00087] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00088] ⌈
If development error detection is enabled: the function shall check the parameter DataPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00151] ⌈
The Ethernet Driver shall indicate broadcast message with the parameter 'IsBroadcast' to the Ethernet Interface.⌋()

[SWS_EthIf_00145] ⌈
If the VLAN is not active the Ethernet Interface shall increment the corresponding measurement data and filter the message⌋()

[SWS_EthIf_00089] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

[SWS_EthIf_00090] ⌈
Caveat: The function shall be callable on interrupt level.⌋()

8.4.2 EthIf_TxConfirmation

[SWS_EthIf_00091] ⌈

Service name:	EthIf_TxConfirmation

Syntax:	`void EthIf_TxConfirmation(`
	`    uint8 CtrlIdx,`
	`    Eth_BufIdxType BufIdx,`
	`    Std_ReturnType Result`

	)	
Service ID[hex]:	0x11	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the physical Ethernet controller within the context of the Ethernet Interface
	BufIdx	Index of the transmitted buffer
	Result	E_OK: The transmission was successful, E_NOT_OK: The transmission failed.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Confirms frame transmission by the indexed controller	

⌋ ()

[SWS_EthIf_00255]⌈
EthIf_TxConfirmation shall pass the Result received within EthIf_TxConfirmation to the configured upper layer via <UL>_TxConfirmation.⌋ ()

[SWS_EthIf_00092]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00093]⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00094]⌈
If development error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM.⌋()

[SWS_EthIf_00095]⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

[SWS_EthIf_00096]⌈
Caveat: The function shall be callable on interrupt level.⌋()

8.4.3 EthIf_CtrlModeIndication

[SWS_EthIf_00231] ⌈

Service name:	EthIf_CtrlModeIndication

Syntax:	`void EthIf_CtrlModeIndication(`
	`    uint8 CtrlIdx,`
	`    Eth_ModeType CtrlMode`
	`)`

Service ID[hex]:	0x0e
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant for the same CtrlIdx, reentrant for different

Parameters (in):	CtrlIdx	Index of the physical Ethernet controller within the context of the Ethernet Interface
	CtrlMode	Notified Ethernet controller mode
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Called asynchronously when mode has been read out. Triggered by previous Eth_SetControllerMode call. Can directly be called within the trigger functions.	

⌋ ()

[SWS_EthIf_00252] ⌈
The function shall call EthSM_CtrlModeIndication.⌋()

8.4.4 EthIf_TrcvModeIndication

[SWS_EthIf_00232] ⌈

Service name:	EthIf_TrcvModeIndication	

Syntax:	`void EthIf_TrcvModeIndication(`
	`    uint8 TrcvIdx,`
	`    EthTrcv_ModeType TrcvMode`
	`)`	

Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in):	TrcvIdx	Index of the Ethernet transceiver within the context of the Ethernet Interface
	TrcvMode	Notified Ethernet transceiver mode
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Called asynchronously when mode has been read out. Triggered by previous Eth_SetTransceiverMode call. Can directly be called within the trigger functions.	

⌋ ()

8.4.5 EthIf_SwitchMgmtInfoIndication

[SWS_EthIf_91006] ⌈

Service name:	EthIf_SwitchMgmtInfoIndication	

Syntax:	`void EthIf_SwitchMgmtInfoIndication(`
	`    uint8 CtrlIdx,`
	`    Eth_DataType* DataPtr,`
	`    EthSwt_MgmtInfoType* MgmtInfoPtr`
	`)`	

Service ID[hex]:	0x3a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of an Ethernet Interface controller
Parameters (inout):	DataPtr	Ethernet data pointer where the management information belongs
Parameters (out):	MgmtInfoPtr	Management information
Return value:	None	
Description:	Ingress Switch management info indication redirected call to upper layers who	

	registered for the call.

⌋ ()

[SWS_EthIf_00291]⌈
The function shall be pre compile time configurable ON/OFF by the configuration parameter: EthIfSwitchManagementSupport.⌋()

[SWS_EthIf_00292]⌈
Caveat: The function requires previous Switch initialization (EthSwt_Init()).⌋()

8.4.6 EthIf_SwitchEgressTimeStampIndication

[SWS_EthIf_91009] ⌈

Service name:	EthIf_SwitchEgressTimeStampIndication	

Syntax:	`void EthIf_SwitchEgressTimeStampIndication(`
	`uint8 CtrlIdx,`
	`Eth_DataType* DataPtr,`
	`EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`Eth_TimeStampType* timeStampPtr`
	`)`	

Service ID[hex]:	0x3b	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of an Ethernet Interface controller
Parameters (inout):	DataPtr	Ethernet data pointer
Parameters (out):	MgmtInfoPtr	Management information
	timeStampPtr	Current timestamp
Return value:	None	
Description:	Returns an egress timestamp value out of the Switch. If the HW resolution is lower than the Eth_TimeStampType resolution resp. range, than the remaining bits will be filled with 0.	

⌋ ()

[SWS_EthIf_00293]⌈
The function shall be pre compile time configurable ON/OFF by the configuration parameter: EthIfGlobalTimeSupport.⌋()

8.4.7 EthIf_SwitchIngressTimeStampIndication

[SWS_EthIf_91008] ⌈

Service name:	EthIf_SwitchIngressTimeStampIndication

Syntax:	`void EthIf_SwitchIngressTimeStampIndication(`
	`uint8 CtrlIdx,`
	`Eth_DataType* DataPtr,`
	`EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`Eth_TimeStampType* timeStampPtr`
	`)`

Service ID[hex]:	0x3c
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	CtrlIdx	Index of an Ethernet Interface controller
	MgmtInfoPtr	Management information
	timeStampPtr	Current timestamp
Parameters (inout):	None	
Parameters (out):	DataPtr	Ethernet data pointer
Return value:	None	
Description:	Returns an ingress timestamp value out of the Switch. If the HW resolution is lower than the Eth_TimeStampType resolution resp. range, than the remaining bits will be filled with 0.	

⌋ ()

[SWS_EthIf_00294]⌈
The function shall be pre compile time configurable ON/OFF by the configuration parameter: EthIfGlobalTimeSupport.⌋ ()

8.5 Scheduled functions

8.5.1 EthIf_MainFunctionRx

[SWS_EthIf_00097] ⌈

Service name:	EthIf_MainFunctionRx

Syntax:	`void EthIf_MainFunctionRx(`
	`    void`
	`)`

Service ID[hex]:	0x20
Description:	The function checks for new received frames and issues transmission confirmations in polling mode. It checks also for transceiver state changes.

⌋ ()
[SWS_EthIf_00098] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00099] ⌈
The receive frame check shall be pre compile time configurable On/Off by the configuration parameter: ETHIF_ENABLE_RX_INTERRUPT.⌋()

8.5.2 EthIf_MainFunctionRx_<PriorityProcessing ShortName>

[SWS_EthIf_91051] ⌈

Service name:	EthIf_MainFunctionRx_<PriorityProcessing ShortName>

Syntax:	`void EthIf_MainFunctionRx_<PriorityProcessing ShortName>(`
	`    void`
	`)`

Service ID[hex]:	0x42
Sync/Async:	Asynchronous
Reentrancy:	Non Reentrant
Parameters (in):	None

Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	The function checks for new received frames at the related Ethernet controller and reception queue by calling Eth_Receive() with the respective FifoIdx. EthIf_MainFunctionRx shall receive frames from all FIFOs that are not assigned for processing via EthIfPhysCtrlRxMainFunctionPriorityProcessing.

⌋ ()

8.5.3 EthIf_MainFunctionTx

[SWS_EthIf_00113] ⌈

Service name:	EthIf_MainFunctionTx

Syntax:	`void EthIf_MainFunctionTx(`
	`    void`
	`)`

Service ID[hex]:	0x21
Description:	The function issues transmission confirmations in polling mode. It checks also for transceiver state changes.

⌋ ()

[SWS_EthIf_00124] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[SWS_EthIf_00100] ⌈
The transmission confirmation check shall be pre compile time configurable On/Off by the configuration parameter: ETHIF_ENABLE_TX_INTERRUPT.⌋()

[SWS_EthIf_00101] ⌈
The frequency of polling the transceiver state change shall be configurable by the configuration parameter: EthIfTrcvLinkStateChgMainReload.⌋()

8.5.4 EthIf_MainFunctionState

[SWS_EthIf_91104] ⌈

Service name:	EthIf_MainFunctionState

Syntax:	`void EthIf_MainFunctionState(`
	`    void`
	`)`

Service ID[hex]:	0x05
Sync/Async:	Asynchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	The function is polling the link state of the used communication hardware (Ethernet transceiver, Ethernet switch ports).

⌋ ()

[SWS_EthIf_00278] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED. ⌋()

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_EthIf_00102] ⌈

API function	*Description*

⌋ ()

8.6.2 Optional Interfaces

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

[SWS_EthIf_00103] ⌈

API function	*Description*
Eth_GetControllerMode	Obtains the state of the indexed controller
Eth_GetPhysAddr	Obtains the physical source address used by the indexed controller
Eth_ProvideTxBuffer	Provides access to a transmit buffer of the FIFO related to the specified priority
Eth_ReadMii	Reads a transceiver register
Eth_Receive	Receive a frame from the related fifo.
Eth_SetControllerMode	Enables / disables the indexed controller
Eth_Transmit	Triggers transmission of a previously filled transmit buffer
Eth_TxConfirmation	Triggers frame transmission confirmation
Eth_WriteMii	Configures a transceiver register or triggers a function offered by the receiver
EthSM_CtrlModeIndication	Called when mode has been read out. Either triggered by previous EthIf_GetControllerMode or by EthIf_SetControllerMode call. Can directly be called within the trigger functions.
EthSwt_EnableTimeStamping	Activates egress time stamping on a dedicated message object on all ports of a Switch but on uplink ports between cascaded switches. The selective activation of dedicated message objects for time stamping reduces the number of notification calls only to the required calls. Some HW does store once the egress time stamp marker and some HW needs it always before transmission. There will be no disabled functionality, due to the fact, that the message type is always "time stamped" by network design.
EthSwt_SetMgmtInfo	Extends the Ethernet frame prepared previously by EthSwt_EthTxPrepareFrame() with the management information to achieve transmission only on specific ports.
EthTrcv_GetBaudRate	Obtains the baud rate of the indexed transceiver

EthTrcv_GetDuplexMode	Obtains the duplex mode of the indexed transceiver
EthTrcv_GetLinkState	Obtains the link state of the indexed transceiver
EthTrcv_GetTransceiverMode	Obtains the state of the indexed transceiver
EthTrcv_SetTransceiverMode	Enables / disables the indexed transceiver
EthTrcv_StartAutoNegotiation	Restarts the negotiation of the transmission parameters used by the indexed transceiver
WEth_GetBufWRxParams	Read out values related to the receive direction for a received packet. For example, this could be RSSI or Channel belonging to one single packet.This API is valid only within the context of WEth_Receive
WEth_GetBufWTxParams	Read out values related to the transmit direction for a transmitted packet. For example, this could be transaction ID belonging to one single packet.This API is valid only within the context of WEth_TxConfirmation.
WEth_SetBufWTxParams	Set values related to the transmit direction for a specific buffer (packet to be sent). For example, this can be the desired transmit power or the channel belonging to one single packet.
WEthTrcv_GetChanRxParams	Read values related to the receive direction of the transceiver. For example, this could be a Channel Busy Ratio (CBR) or the average Channel Idle Time (CIT).
WEthTrcv_SetChanRxParams	Set values related to the receive direction of a transceiver's wireless channel.For example, this could be a channel parameter like the frequency.
WEthTrcv_SetChanTxParams	Set values related to the transmit direction of a transceiver's wireless channel. For example, this could be the bitrate of a channel.
WEthTrcv_SetRadioParams	Set values related to a transceiver's wireless radio. For example, this could be the selection of the radio settings (channel, ...).

⌋ ()

8.6.3 Configurable interfaces

This chapter lists all interfaces with configurable target functions. The target function is usually a callback function. The function names are configurable.

[SWS_EthIf_00104] ⌈

Service name:	<User>_RxIndication	

Syntax:	`void <User>_RxIndication(`
	`    uint8 CtrlIdx,`
	`    Eth_FrameType FrameType,`
	`    boolean IsBroadcast,`
	`    const uint8* PhysAddrPtr,`
	`    uint8* DataPtr,`
	`    uint16 LenByte`
	`)`	

Service ID[hex]:	--	
Sync/Async:	--	
Reentrancy:	Dont care	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	FrameType	frame type of received Ethernet frame
	IsBroadcast	parameter to indicate a broadcast frame
	PhysAddrPtr	pointer to Physical source address (MAC address in network byte order) of received Ethernet frame
	DataPtr	Pointer to payload of the received Ethernet frame (i.e. Ethernet header is not provided).

	LenByte	Length of received data.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indicates the reception of an Ethernet frame	

⌋ () [SWS_EthIf_00105] ⌈
The callback function shall be configurable by the configuration parameter: EthIfRxIndicationFunction.⌋()

[SWS_EthIf_00106] ⌈

Service name:	<UL>_TxConfirmation	

Syntax:	`void <UL>_TxConfirmation(`
	`    uint8 CtrlIdx,`
	`    Eth_BufIdxType BufIdx,`
	`    Std_ReturnType Result`
	`)`	

Service ID[hex]:	--	
Sync/Async:	--	
Reentrancy:	Dont care	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	BufIdx	Index of the buffer resource
	Result	--
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Confirms the transmission of an Ethernet frame	

⌋ () [SWS_EthIf_00107] ⌈
The callback function shall be configurable by the configuration parameter: EthIfTxConfirmationFunction.⌋()

[SWS_EthIf_00108] ⌈

Service name:	<User>_TrcvLinkStateChg	

Syntax:	`void <User>_TrcvLinkStateChg(`
	`    uint8 CtrlIdx,`
	`    EthTrcv_LinkStateType TrcvLinkState`
	`)`	

Service ID[hex]:	--	
Sync/Async:	--	
Reentrancy:	Don’t care	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	TrcvLinkState	ETHTRCV_LINK_STATE_DOWN transceiver link is down ETHTRCV_LINK_STATE_ACTIVE transceiver link is up
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indicates the change of a transceiver state	

⌋ () [SWS_EthIf_00109] ⌈
The callback function shall be configurable by the configuration parameter: EthIfTrcvLinkStateChgFunction.⌋()

[SWS_EthIf_00229] ⌈
EthIfControllers not referring to an Ethernet Transceiver, i.e. no valid EthIfEthTrcvRef is configured, shall act as if the transceiver was present and the transceiver status was ETHTRCV_LINK_STATE_ACTIVE.⌋()

[SWS_EthIf_00230] ⌈
Upon change of link state <User>_TrcvLinkStateChg shall be invoked for every affected EthIfController.⌋()

[SWS_EthIf_91006] ⌈

Service name:	<User>_SwitchMgmtInfoIndication	

Syntax:	`void <User>_SwitchMgmtInfoIndication(`
	`    uint8 CtrlIdx,`
	`    uint8* DataPtr,`
	`    EthSwt_MgmtInfoType* MgmtInfoPtr`
	`)`	

Service ID[hex]:	0x3f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
Parameters (inout):	DataPtr	Ethernet data pointer where the management information belongs
Parameters (out):	MgmtInfoPtr	Management information
Return value:	None	
Description:	Ingress Switch management info indication redirected call to upper layers who registered for the call.	

⌋ () [SWS_EthIf_00295]⌈
The function <User>_SwitchMgmtInfoIndication() shall be pre compile time configurable by the configuration parameter: SwitchMgmtInfoIndicationFunction.⌋()

[SWS_EthIf_91009] ⌈

Service name:	EthIf_SwitchEgressTimeStampIndication	

Syntax:	`void EthIf_SwitchEgressTimeStampIndication(`
	`    uint8 CtrlIdx,`
	`    Eth_DataType* DataPtr,`
	`    EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`    Eth_TimeStampType* timeStampPtr`
	`)`	

Service ID[hex]:	0x3b	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of an Ethernet Interface controller
Parameters (inout):	DataPtr	Ethernet data pointer
Parameters (out):	MgmtInfoPtr	Management information
	timeStampPtr	Current timestamp
Return value:	None	
Description:	Returns an egress timestamp value out of the Switch. If the HW resolution is lower than the Eth_TimeStampType resolution resp. range, than the remaining bits will be filled with 0.	

⌋ ()
[SWS_EthIf_00296]⌈

The function <User>_SwitchEgressTimeStampIndication() shall be pre compile time configurable by the configuration parameter: SwitchEgressTimeStampIndicationFunction.⌋()

[SWS_EthIf_91008] ⌈

Service name:	EthIf_SwitchIngressTimeStampIndication	

Syntax:	`void EthIf_SwitchIngressTimeStampIndication(`
	`    uint8 CtrlIdx,`
	`    Eth_DataType* DataPtr,`
	`    EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`    Eth_TimeStampType* timeStampPtr`
	`)`	

Service ID[hex]:	0x3c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of an Ethernet Interface controller
	MgmtInfoPtr	Management information
	timeStampPtr	Current timestamp
Parameters (inout):	None	
Parameters (out):	DataPtr	Ethernet data pointer
Return value:	None	
Description:	Returns an ingress timestamp value out of the Switch. If the HW resolution is lower than the Eth_TimeStampType resolution resp. range, than the remaining bits will be filled with 0.	

⌋ ()

[SWS_EthIf_00297]⌈

The function <User>_SwitchEgressTimeStampIndication() shall be pre compile time configurable by the configuration parameter: SwitchEgressTimeStampIndicationFunction.⌋()

Terms and definitions:

Reentrant: interface is 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 sequence diagrams show the basic operations carried out during operation. They show the interaction of the Ethernet Interface with upper layer BSW module and the underlying Ethernet Controller Driver.

Please note that the sequence diagrams are an extension for illustrational purposes to ease understanding of the specification.

9.1 Initialization

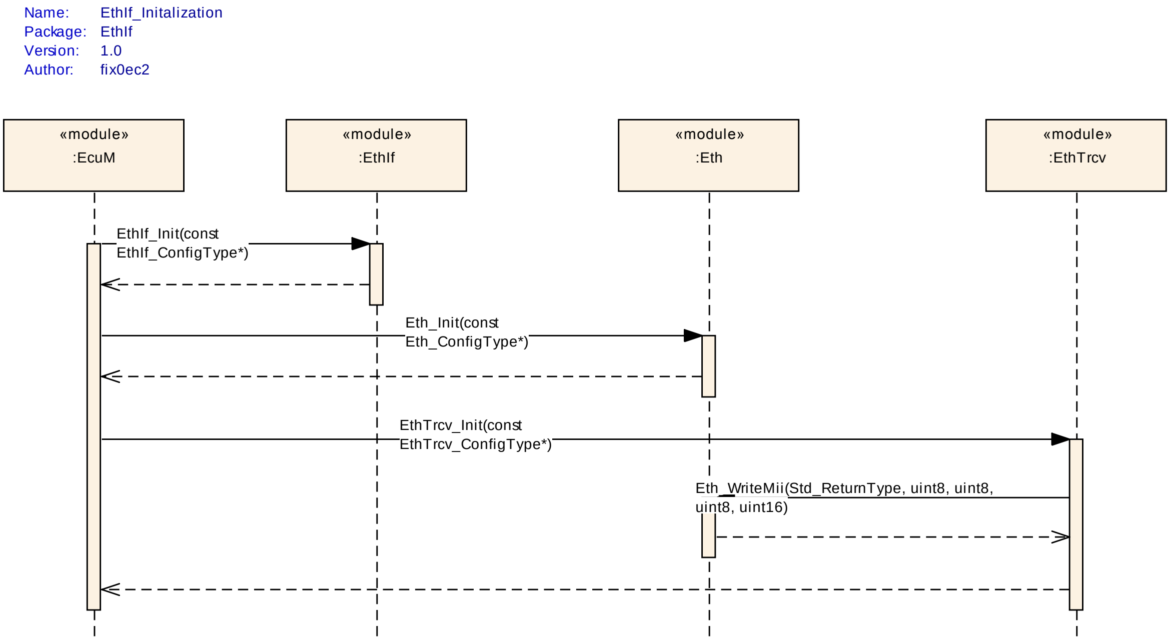
Name: EthIf_Initalization
Package: EthIf
Version: 1.0
Author: fix0ec2
«module» :EcuM
«module» :EthIf
«module» :Eth
«module» :EthTrcv
EthIf_Init(const EthIf_ConfigType*)
Eth_Init(const Eth_ConfigType*)
EthTrcv_Init(const EthTrcv_ConfigType*)
Eth_WriteMii(Std_ReturnType, uint8, uint8, uint8, uint16)

Figure 5: Initialization

9.2 Communication Initialization

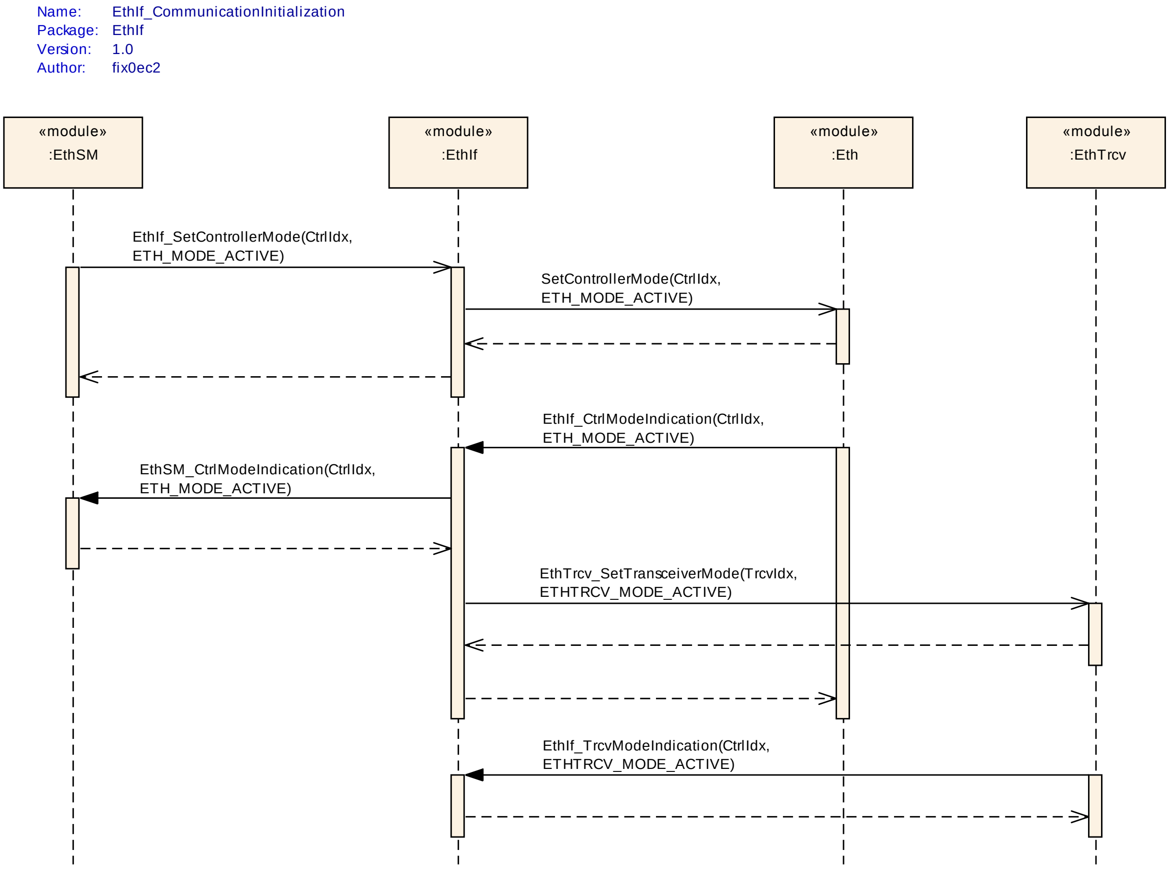
Name: EthIf_CommunicationInitialization
Package: EthIf
Version: 1.0
Author: fix0ec2
«module» :EthSM
«module» :EthIf
«module» :Eth
«module» :EthTrcv
EthIf_SetControllerMode(CtrlIdx, ETH_MODE_ACTIVE)
SetControllerMode(CtrlIdx, ETH_MODE_ACTIVE)
EthIf_CtrlModeIndication(CtrlIdx, ETH_MODE_ACTIVE)
EthSM_CtrlModeIndication(CtrlIdx, ETH_MODE_ACTIVE)
EthTrcv_SetTransceiverMode(TrcvIdx, ETHTRCV_MODE_ACTIVE)
EthIf_TrcvModeIndication(CtrlIdx, ETHTRCV_MODE_ACTIVE)

Figure 6: Communication Initialization

9.3 Data Transmission

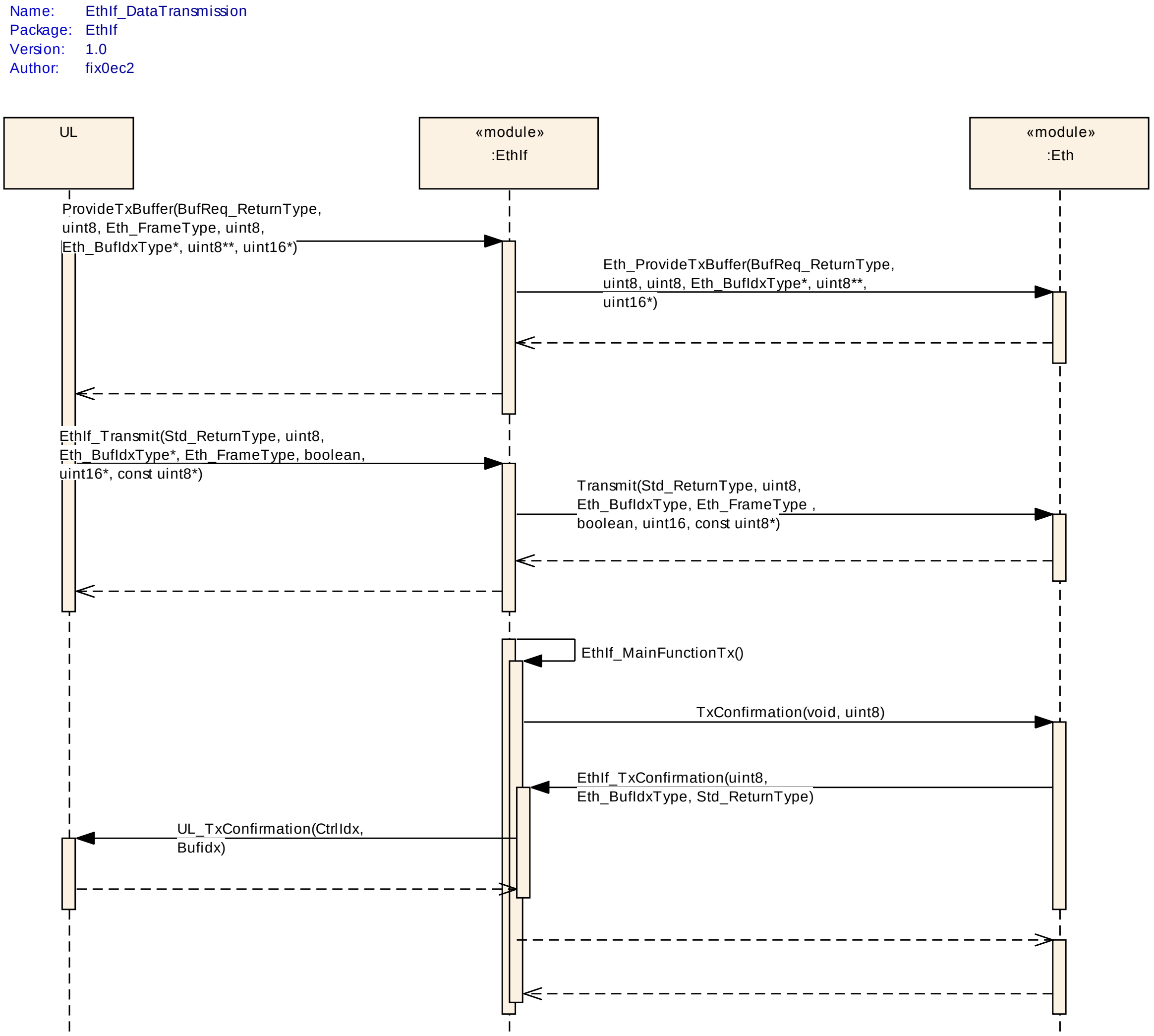
Name: EthIf_DataTransmission
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module» :EthIf
«module» :Eth
ProvideTxBuffer(BufReq_ReturnType, uint8, Eth_FrameType, uint8, Eth_BufIdxType*, uint8**, uint16*)
Eth_ProvideTxBuffer(BufReq_ReturnType, uint8, uint8, Eth_BufIdxType*, uint8**, uint16*)
EthIf_Transmit(Std_ReturnType, uint8, Eth_BufIdxType*, Eth_FrameType, boolean, uint16*, const uint8*)
Transmit(Std_ReturnType, uint8, Eth_BufIdxType, Eth_FrameType , boolean, uint16, const uint8*)
EthIf_MainFunctionTx()
TxConfirmation(void, uint8)
EthIf_TxConfirmation(uint8, Eth_BufIdxType, Std_ReturnType)
UL_TxConfirmation(CtrlIdx, Bufidx)

Figure 7: Frame Transmission in Polling Mode

[SWS_EthIf_00115]
In each call of EthIf_MainFunctionTx the component shall call Eth_TxConfirmation for all Ethernet Controller Drivers.
Note: The Ethernet Interface expects that each Ethernet Controller Driver issues confirmations for all transmitted frames using the call-back function EthIf_TxConfirmation.

[SWS_EthIf_00125]
EthIf_ TxConfirmation shall forward the confirmation to the registered call-back functions <User>_TxConfirmation.

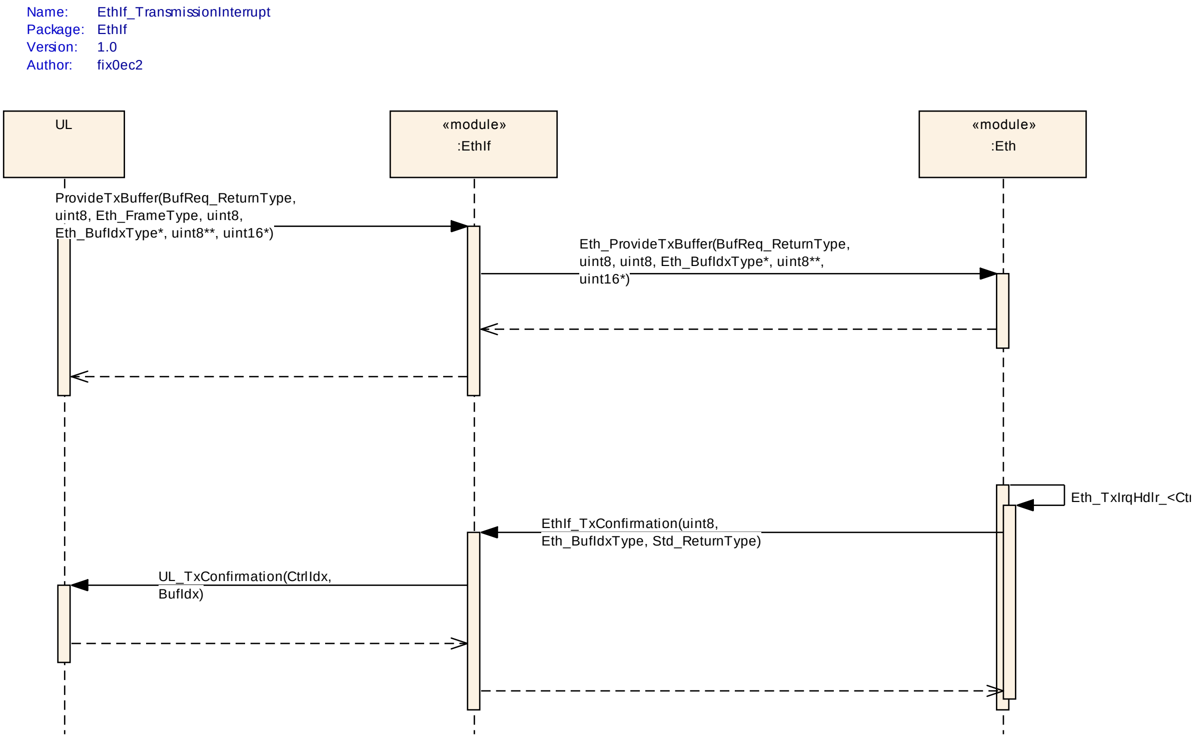
Name: EthIf_TransmissionInterrupt
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module»
:EthIf
«module»
:Eth
ProvideTxBuffer(BufReq_ReturnType, uint8, Eth_FrameType, uint8, Eth_BufIdxType*, uint8**, uint16*)
Eth_ProvideTxBuffer(BufReq_ReturnType, uint8, uint8, Eth_BufIdxType*, uint8**, uint16*)
Eth_TxIrqHdlr_<Ct
EthIf_TxConfirmation(uint8, Eth_BufIdxType, Std_ReturnType)
UL_TxConfirmation(CtrlIdx, BufIdx)

Figure 8: Frame Transmission in Interrupt Mode

9.4 Data Reception

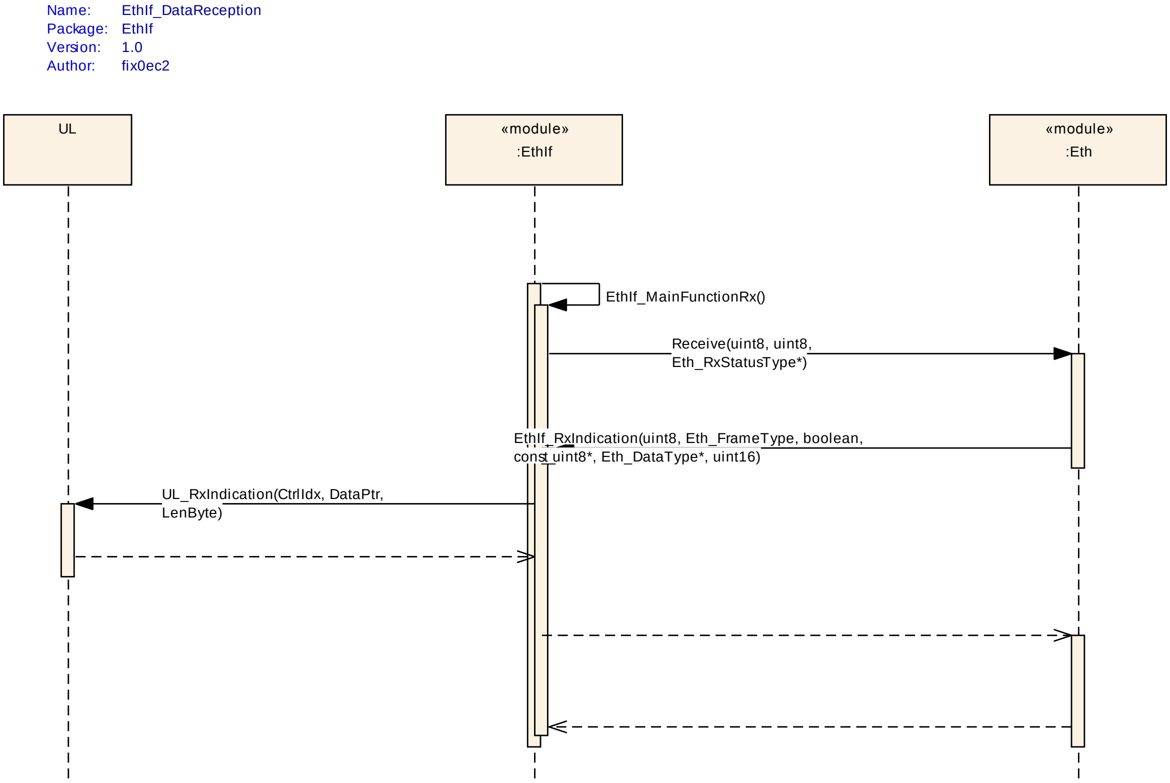
Name: EthIf_DataReception
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module»
:EthIf
«module»
:Eth
EthIf_MainFunctionRx()
Receive(uint8, uint8, Eth_RxStatusType*)
EthIf_RxIndication(uint8, Eth_FrameType, boolean, const uint8*, Eth_DataType*, uint16)
UL_RxIndication(CtrlIdx, DataPtr, LenByte)

Figure 9: Frame Reception in Polling Mode

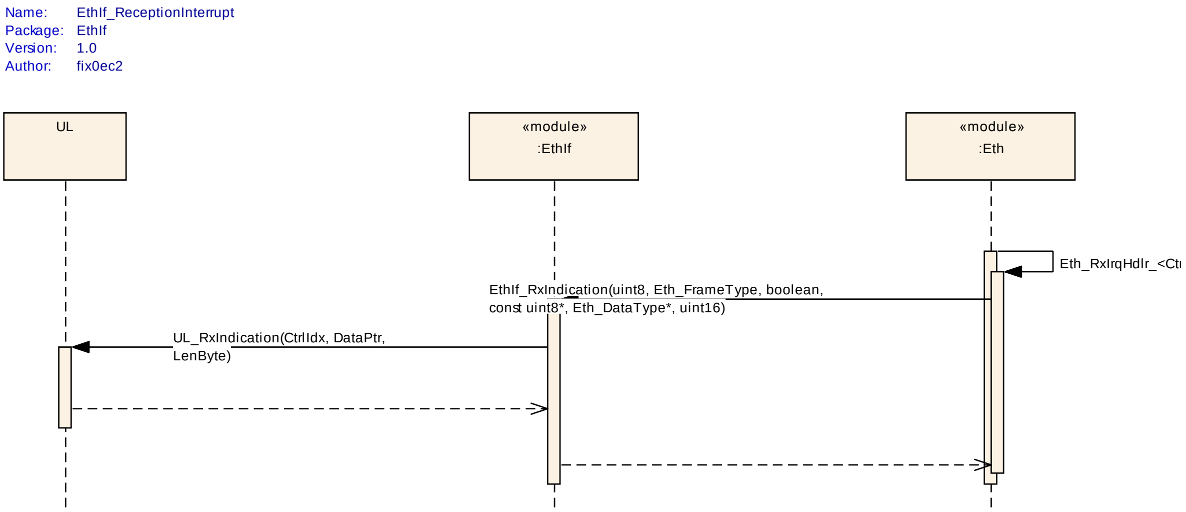
Name: EthIf_ReceptionInterrupt
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module»
:EthIf
«module»
:Eth
Eth_RxIrqHdlr_<Ctı
EthIf_RxIndication(uint8, Eth_FrameType, boolean, const uint8*, Eth_DataType*, uint16)
UL_RxIndication(CtrlIdx, DataPtr, LenByte)

Figure 10: Frame Reception in Interrupt Mode

9.5 Link State Change

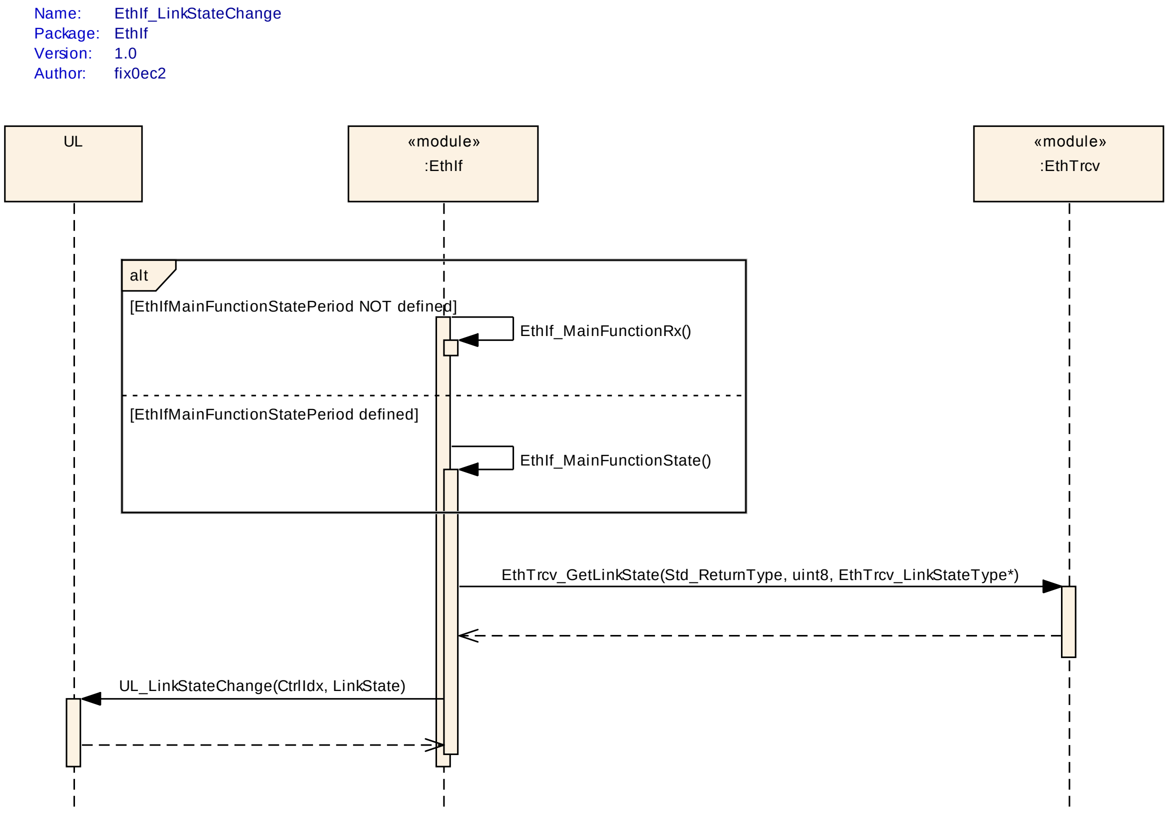
Name: EthIf_LinkStateChange
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module»
:EthIf
«module»
:EthTrcv
alt
[EthIfMainFunctionStatePeriod NOT defined]
EthIf_MainFunctionRx()
[EthIfMainFunctionStatePeriod defined]
EthIf_MainFunctionState()
EthTrcv_GetLinkState(Std_ReturnType, uint8, EthTrcv_LinkStateType*)
UL_LinkStateChange(CtrlIdx, LinkState)

Figure 11: Link State Change

9.6 Switch Management support

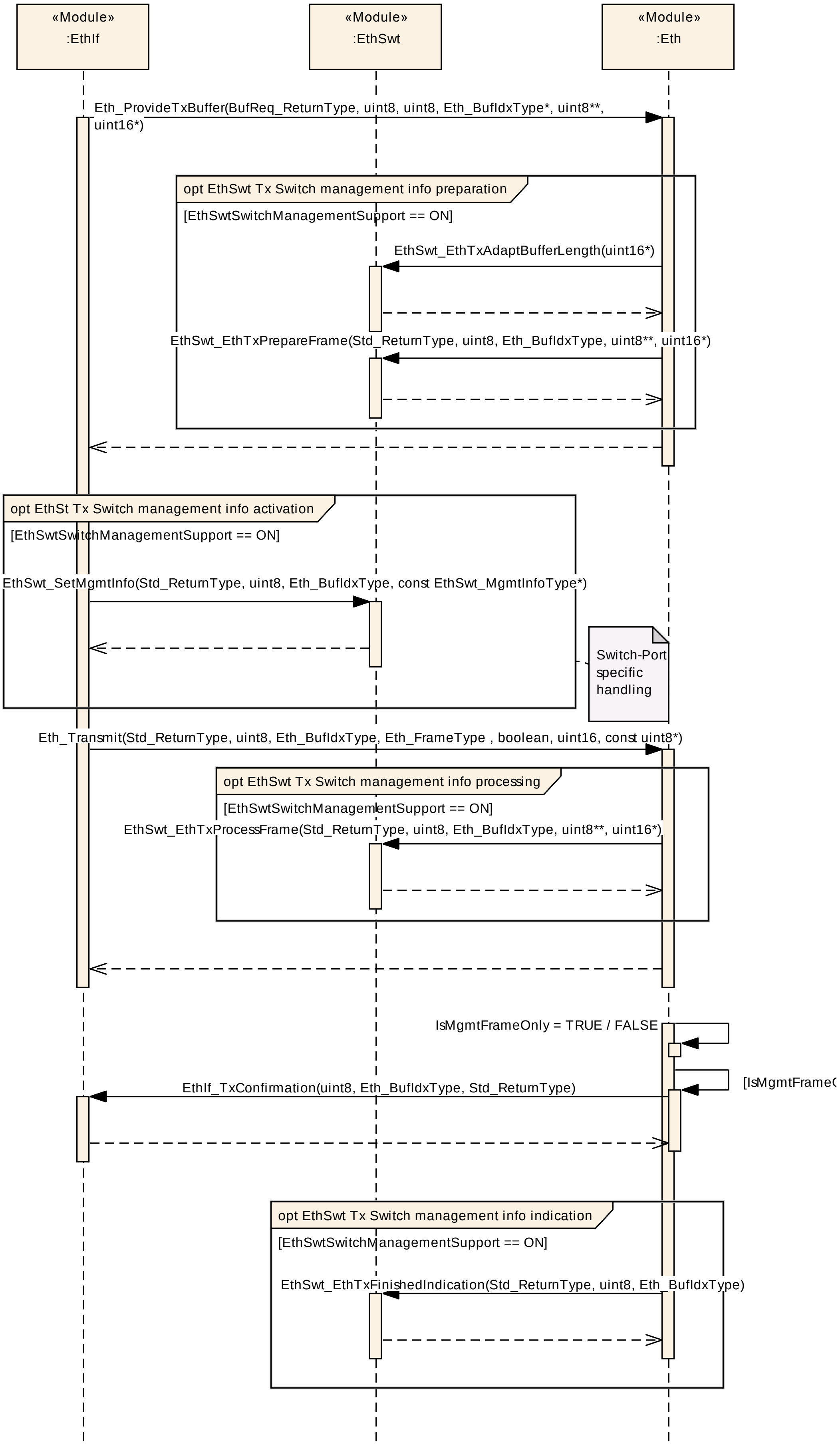
«Module»
:EthIf
«Module»
:EthSwt
«Module»
:Eth
Eth_ProvideTxBuffer(BufReq_ReturnType, uint8, uint8, Eth_BufIdxType*, uint8**, uint16*)
opt EthSwt Tx Switch management info preparation
[EthSwtSwitchManagementSupport == ON]
EthSwt_EthTxAdaptBufferLength(uint16*)
EthSwt_EthTxPrepareFrame(Std_ReturnType, uint8, Eth_BufIdxType, uint8**, uint16*)
opt EthSt Tx Switch management info activation
[EthSwtSwitchManagementSupport == ON]
EthSwt_SetMgmtInfo(Std_ReturnType, uint8, Eth_BufIdxType, const EthSwt_MgmtInfoType*)
Switch-Port specific handling
Eth_Transmit(Std_ReturnType, uint8, Eth_BufIdxType, Eth_FrameType , boolean, uint16, const uint8*)
opt EthSwt Tx Switch management info processing
[EthSwtSwitchManagementSupport == ON]
EthSwt_EthTxProcessFrame(Std_ReturnType, uint8, Eth_BufIdxType, uint8**, uint16*)
IsMgmtFrameOnly = TRUE / FALSE
[IsMgmtFrame(
EthIf_TxConfirmation(uint8, Eth_BufIdxType, Std_ReturnType)
opt EthSwt Tx Switch management info indication
[EthSwtSwitchManagementSupport == ON]
EthSwt_EthTxFinishedIndication(Std_ReturnType, uint8, Eth_BufIdxType)

Figure 12: Switch Management support for transmission

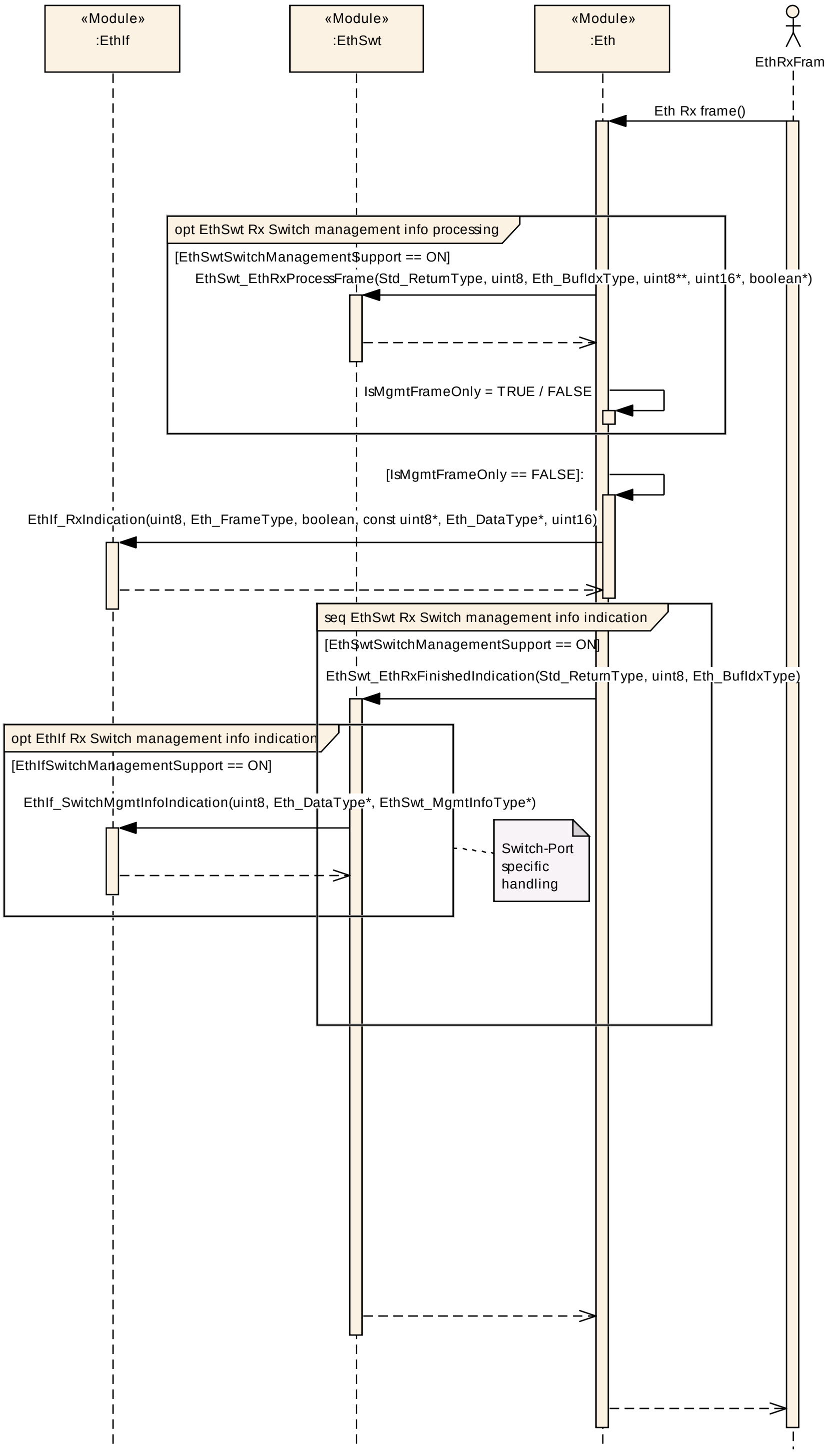
«Module»
:EthIf
«Module»
:EthSwt
«Module»
:Eth
EthRxFrame
Eth Rx frame()
opt EthSwt Rx Switch management info processing
[EthSwtSwitchManagementSupport == ON]
EthSwt_EthRxProcessFrame(Std_ReturnType, uint8, Eth_BufIdxType, uint8**, uint16*, boolean*)
IsMgmtFrameOnly = TRUE / FALSE
[IsMgmtFrameOnly == FALSE]:
EthIf_RxIndication(uint8, Eth_FrameType, boolean, const uint8*, Eth_DataType*, uint16)
seq EthSwt Rx Switch management info indication
[EthSwtSwitchManagementSupport == ON]
EthSwt_EthRxFinishedIndication(Std_ReturnType, uint8, Eth_BufIdxType)
opt EthIf Rx Switch management info indication
[EthIfSwitchManagementSupport == ON]
EthIf_SwitchMgmtInfoIndication(uint8, Eth_DataType*, EthSwt_MgmtInfoType*)
Switch-Port specific handling

Figure 13: Switch Management support for reception

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 Interface.

Chapter 10.3 specifies published information of the module Ethernet Interface.

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.

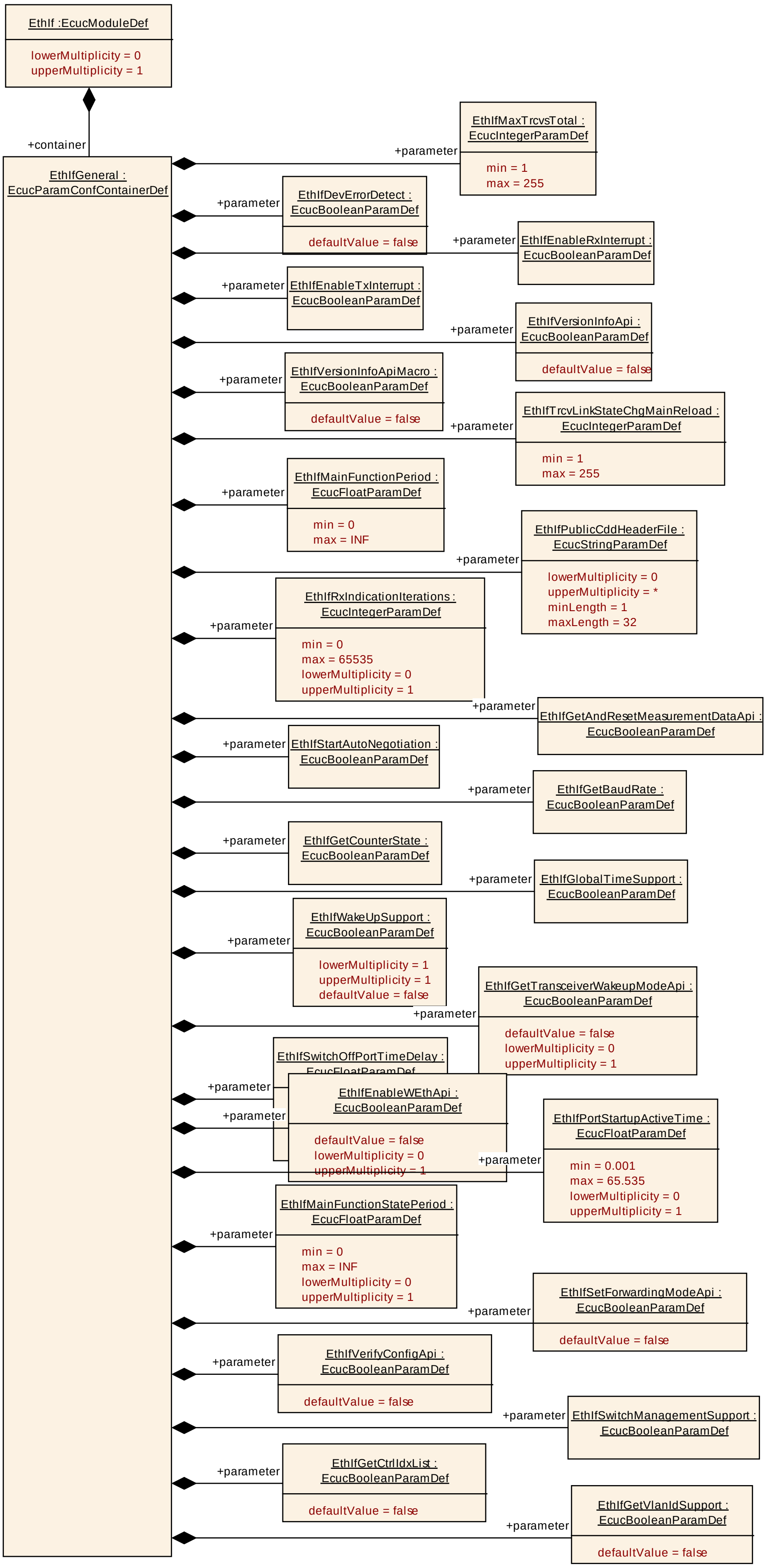
EthIf :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
EthIfGeneral :
EcucParamConfContainerDef
+parameter
EthIfMaxTrcvsTotal :
EcucIntegerParamDef
min = 1
max = 255
+parameter
EthIfDevErrorDetect :
EcucBooleanParamDef
defaultValue = false
+parameter
EthIfEnableRxInterrupt :
EcucBooleanParamDef
+parameter
EthIfEnableTxInterrupt :
EcucBooleanParamDef
+parameter
EthIfVersionInfoApi :
EcucBooleanParamDef
defaultValue = false
+parameter
EthIfVersionInfoApiMacro :
EcucBooleanParamDef
defaultValue = false
+parameter
EthIfTrcvLinkStateChgMainReload :
EcucIntegerParamDef
min = 1
max = 255
+parameter
EthIfMainFunctionPeriod :
EcucFloatParamDef
min = 0
max = INF
+parameter
EthIfPublicCddHeaderFile :
EcucStringParamDef
lowerMultiplicity = 0
upperMultiplicity = *
minLength = 1
maxLength = 32
+parameter
EthIfRxIndicationIterations :
EcucIntegerParamDef
min = 0
max = 65535
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthIfGetAndResetMeasurementDataApi :
EcucBooleanParamDef
+parameter
EthIfStartAutoNegotiation :
EcucBooleanParamDef
+parameter
EthIfGetBaudRate :
EcucBooleanParamDef
+parameter
EthIfGetCounterState :
EcucBooleanParamDef
+parameter
EthIfGlobalTimeSupport :
EcucBooleanParamDef
+parameter
EthIfWakeUpSupport :
EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
defaultValue = false
+parameter
EthIfGetTransceiverWakeupModeApi :
EcucBooleanParamDef
defaultValue = false
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthIfSwitchOffPortTimeDelay :
EcucFloatParamDef
+parameter
EthIfEnableWEthApi :
EcucBooleanParamDef
defaultValue = false
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthIfPortStartupActiveTime :
EcucFloatParamDef
min = 0.001
max = 65.535
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthIfMainFunctionStatePeriod :
EcucFloatParamDef
min = 0
max = INF
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthIfSetForwardingModeApi :
EcucBooleanParamDef
defaultValue = false
+parameter
EthIfVerifyConfigApi :
EcucBooleanParamDef
defaultValue = false
+parameter
EthIfSwitchManagementSupport :
EcucBooleanParamDef
+parameter
EthIfGetCtrlIdxList :
EcucBooleanParamDef
defaultValue = false
+parameter
EthIfGetVlanIdSupport :
EcucBooleanParamDef
defaultValue = false

Figure 10.1: Ethernet Interface general configuration structure

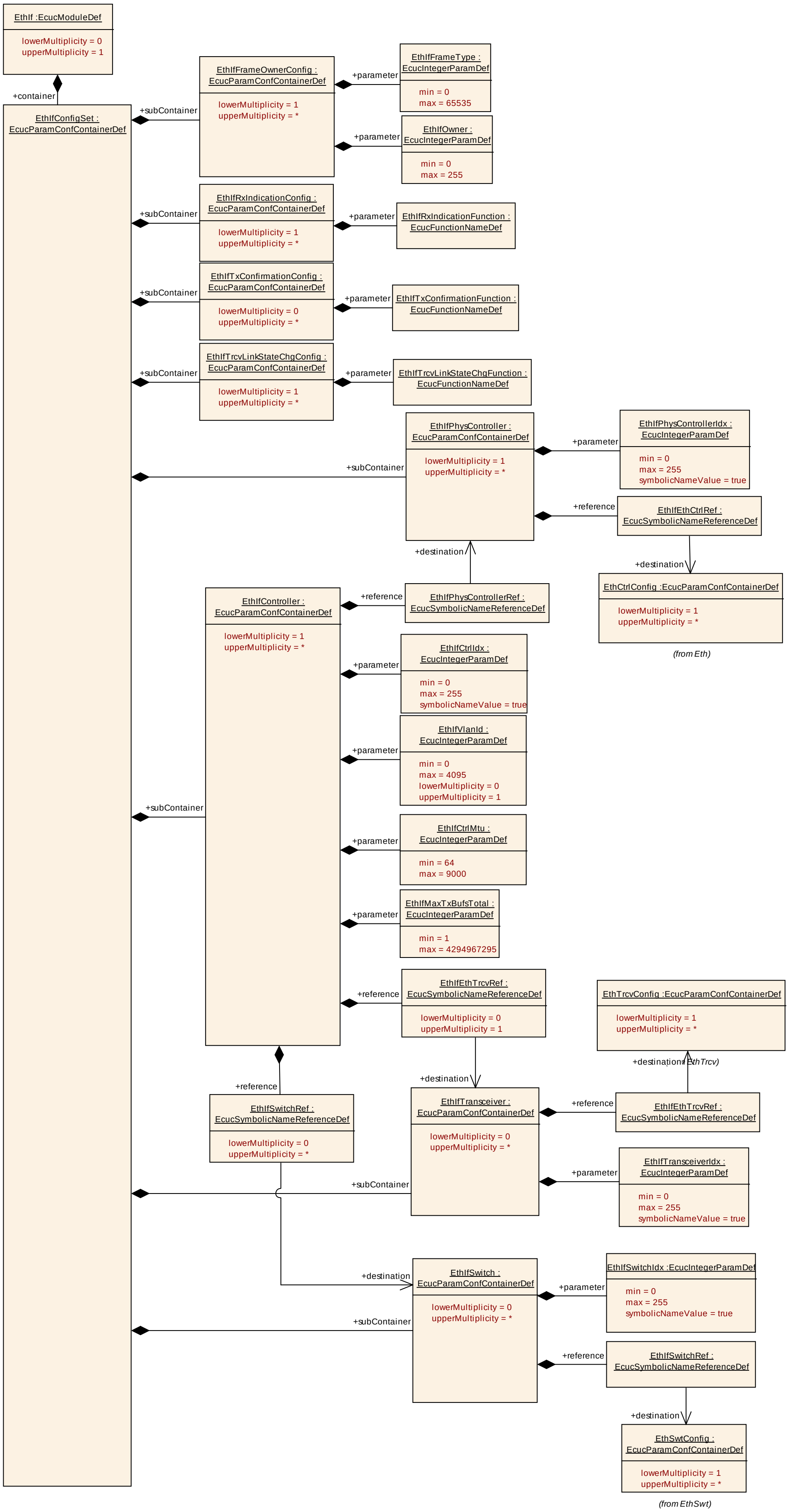
EthIf :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
EthIfConfigSet : EcucParamConfContainerDef
+subContainer
EthIfFrameOwnerConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
EthIfFrameType : EcucIntegerParamDef
min = 0
max = 65535
+parameter
EthIfOwner : EcucIntegerParamDef
min = 0
max = 255
+subContainer
EthIfRxIndicationConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
EthIfRxIndicationFunction : EcucFunctionNameDef
+subContainer
EthIfTxConfirmationConfig : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+parameter
EthIfTxConfirmationFunction : EcucFunctionNameDef
+subContainer
EthIfTrcvLinkStateChgConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
EthIfTrcvLinkStateChgFunction : EcucFunctionNameDef
+subContainer
EthIfPhysController : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
EthIfPhysControllerIdx : EcucIntegerParamDef
min = 0
max = 255
symbolicNameValue = true
+reference
EthIfEthCtrlRef : EcucSymbolicNameReferenceDef
+destination
EthCtrlConfig :EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from Eth)
+destination
EthIfController : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+reference
EthIfPhysControllerRef : EcucSymbolicNameReferenceDef
+parameter
EthIfCtrlIdx : EcucIntegerParamDef
min = 0
max = 255
symbolicNameValue = true
+parameter
EthIfVlanId : EcucIntegerParamDef
min = 0
max = 4095
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
+parameter
EthIfCtrlMtu : EcucIntegerParamDef
min = 64
max = 9000
+parameter
EthIfMaxTxBufsTotal : EcucIntegerParamDef
min = 1
max = 4294967295
+reference
EthIfEthTrcvRef : EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
EthTrcvConfig :EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+destination
(from EthTrcv)
+reference
EthIfSwitchRef : EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = *
+destination
EthIfTransceiver : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+reference
EthIfEthTrcvRef : EcucSymbolicNameReferenceDef
+parameter
EthIfTransceiverIdx : EcucIntegerParamDef
min = 0
max = 255
symbolicNameValue = true
+subContainer
+destination
EthIfSwitch : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+subContainer
+parameter
EthIfSwitchIdx :EcucIntegerParamDef
min = 0
max = 255
symbolicNameValue = true
+reference
EthIfSwitchRef : EcucSymbolicNameReferenceDef
+destination
EthSwtConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from EthSwt)

Figure 10.2: Ethernet Interface interface configuration structure

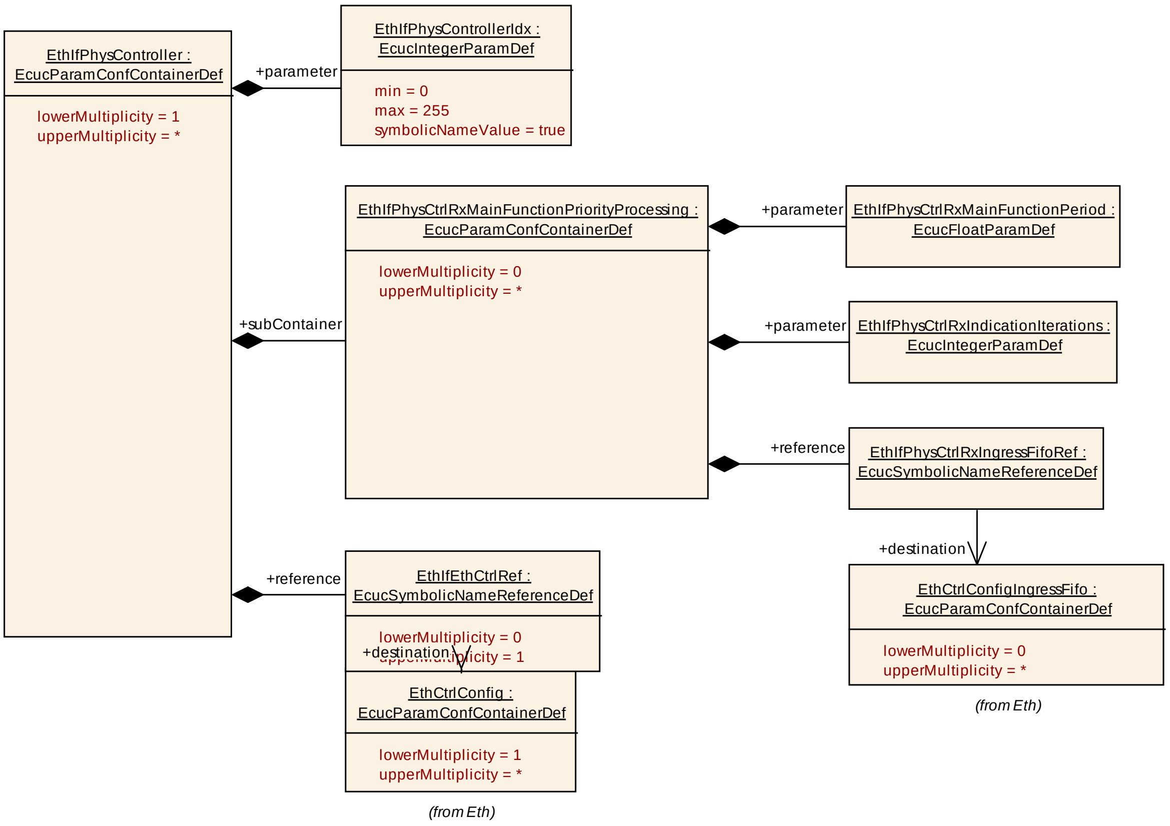
EthIfPhysController : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
EthIfPhysControllerIdx : EcucIntegerParamDef
min = 0
max = 255
symbolicNameValue = true
+subContainer
EthIfPhysCtrlRxMainFunctionPriorityProcessing : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+parameter
EthIfPhysCtrlRxMainFunctionPeriod : EcucFloatParamDef
+parameter
EthIfPhysCtrlRxIndicationIterations : EcucIntegerParamDef
+reference
EthIfPhysCtrlRxIngressFifoRef : EcucSymbolicNameReferenceDef
+destination
EthCtrlConfigIngressFifo : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
(from Eth)
+reference
EthIfEthCtrlRef : EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
EthCtrlConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from Eth)

Figure 10.3: Ethernet Interface physical controller configuration structure

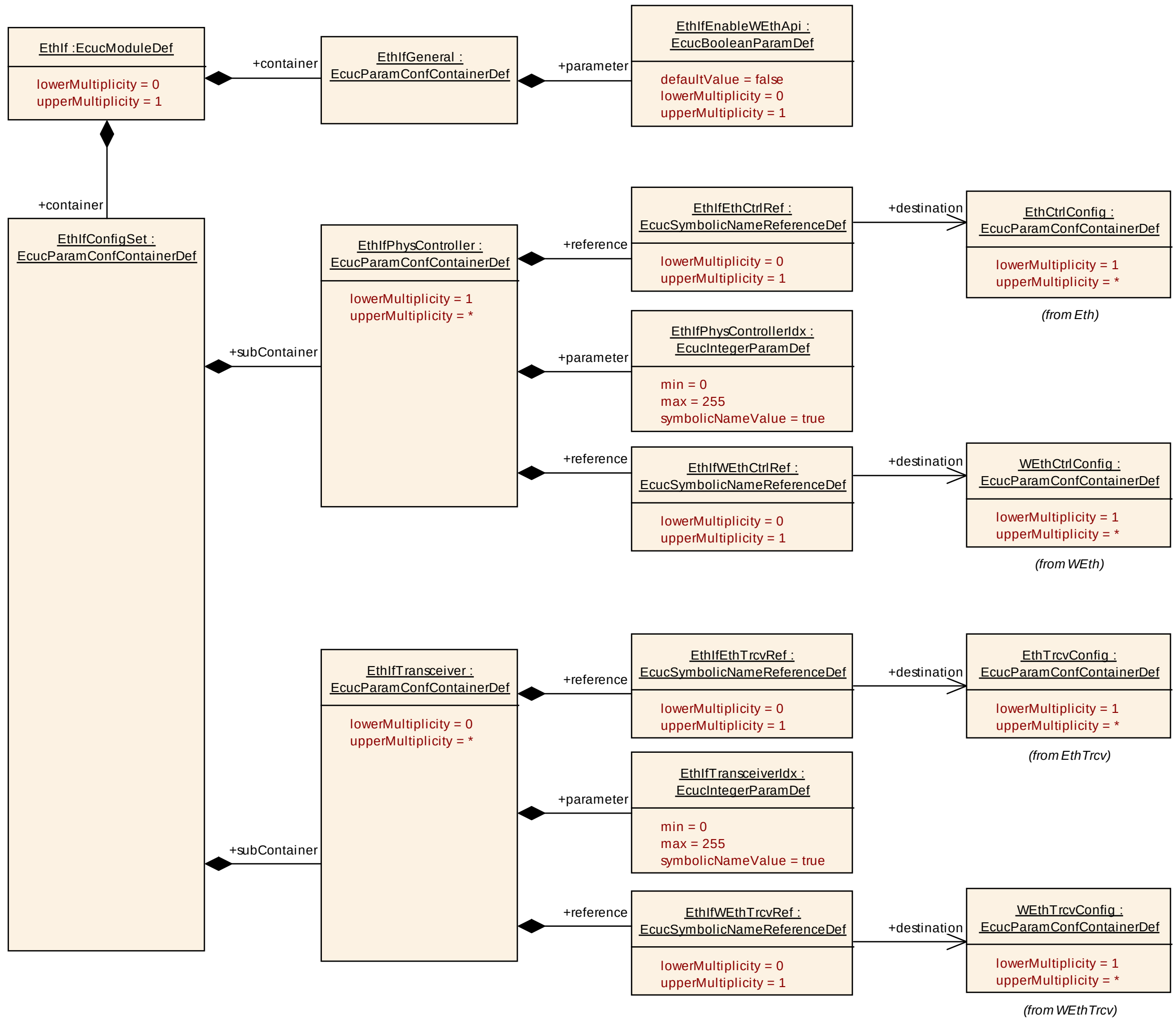

Figure 10.4: Ethernet Interface controller and transceiver configuration structure

10.1.1 EthIf

SWS Item	**ECUC_EthIf_00049 :**
Module Name	*EthIf*
Module Description	Configuration of the EthIf (Ethernet Interface) module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthIfConfigSet	1	Collecting container for all parameters with post-build configuration classes.
EthIfGeneral	1	This container contains the general configuration parameters of the Ethernet Interface.

10.1.2 EthIfGeneral

SWS Item	**ECUC_EthIf_00001 :**		
Container Name	EthIfGeneral		
Description	This container contains the general configuration parameters of the Ethernet Interface.		
Configuration Parameters			

SWS Item	**ECUC_EthIf_00004 :**		
Name	EthIfDevErrorDetect		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00005 :**		
Name	EthIfEnableRxInterrupt		
Description	Enables / Disables receive interrupt.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00006 :**		
Name	EthIfEnableTxInterrupt		
Description	Enables / Disables the transmit interrupt.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00075 :**
Name	EthIfEnableWEthApi
Description	Enables / Disables API's for WEth / WEthTrcv
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Post-Build Variant Multiplicity	false

Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00072 :**		
Name	EthIfGetAndResetMeasurementDataApi		
Description	Enables / Disables the Get and Reset Measurement Data API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00034 :**		
Name	EthIfGetBaudRate		
Description	Enables / Disables GetBaudRate API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00035 :**		
Name	EthIfGetCounterState		
Description	Enables / Disables GetCounterState API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00070 :**		
Name	EthIfGetCtrlIdxList		
Description	Enables / Disables GetCtrlIdxList API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

SWS Item	**ECUC_EthIf_00041 :**		
Name	EthIfGetTransceiverWakeupModeApi		
Description	Enables / Disables EthIf_GetTransceiverWakeupMode API		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: Only valid if EthIfWakeUpSupport is TRUE		

SWS Item	**ECUC_EthIf_00071 :**		
Name	EthIfGetVlanIdSupport		
Description	Enables / Disables GetVlanId API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00039 :**		
Name	EthIfGlobalTimeSupport		
Description	Enables/Disables the Global Time APIs used amongst others by Global Time Synchronization over Ethernet.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00023 :**		
Name	EthIfMainFunctionPeriod		
Description	Specifies the period of main function EthIf_MainFunctionRx and EthIf_MainFunctionTx in seconds. Ethernet Interface does not require this information but the BSW scheduler.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants

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

SWS Item	**ECUC_EthIf_00056 :**		
Name	EthIfMainFunctionStatePeriod		
Description	Specifies the period of main function EthIf_MainFunctionState in seconds. Ethernet Interface does not require this information but the BSW scheduler.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: If parameter is defined, then EthIf_MainFunctionState shall be generated.		

SWS Item	**ECUC_EthIf_00003 :**		
Name	EthIfMaxTrcvsTotal		
Description	Limits the total number of transceivers.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00055 :**		
Name	EthIfPortStartupActiveTime		
Description	Denote the time delay after the mode "ETHTRCV_MODE_ACTIVE" of all EthIfSwitchPorts are requested via EthIf_StartAllPorts. This is only used for ports in EthIfSwtPortGroups which are not referenced by a EthIfController or the reference is of type "link-information".		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0.001 .. 65.535]		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE

	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00024 :**		
Name	EthIfPublicCddHeaderFile		
Description	Defines header files for callback functions which shall be included in case of CDDs. Range of characters is 1.. 32.		
Multiplicity	0..*		
Type	EcucStringParamDef		
Default value	--		
maxLength	32		
minLength	1		
regularExpression	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_EthIf_00030 :**		
Name	EthIfRxIndicationIterations		
Description	Maximum number of Ethernet frames per Ethernet controller polled from the Ethernet driver within EthIf_MainFunctionRx.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00062 :**		
Name	EthIfSetForwardingModeApi		
Description	Enables /disables EthIf_SetForwardingMode API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00033 :**
Name	EthIfStartAutoNegotiation
Description	Enables / Disables StartAutoNegotiation API.
Multiplicity	1

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

SWS Item	**ECUC_EthIf_00064 :**		
Name	EthIfSwitchManagementSupport		
Description	Enables/Disables the Switch management APIs to support a Switch-port specific communication attribute access.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00054 :**		
Name	EthIfSwitchOffPortTimeDelay		
Description	Denote the time delay after the mode "ETHTRCV_MODE_DOWN" of a EthIfSwitchPortGroup will be executed. This is only used for EthIfSwtPortGroups which not referenced by a EthIfController or the reference is of type "link-information". The time delay shall be greater than the UdpNm timings, because UdpNm shall finish its shutdown handling. (Repeat Message State, Prepare Bus-Sleep state, Bus-Sleep state).		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0.001 .. 65.535]		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local dependency: EthIfSwitchOffPortTimeDelay > (UdpNmTimeoutTime + UdpNmWaitBusSleepTime)		

SWS Item	**ECUC_EthIf_00009 :**		
Name	EthIfTrcvLinkStateChgMainReload		
Description	Specifies the frequency of transceiver link state change checks in each period of main function EthIf_MainFunctionTx.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		

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

SWS Item	**ECUC_EthIf_00063 :**		
Name	EthIfVerifyConfigApi		
Description	Enables /disables EthIf_VerifyConfig API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00007 :**		
Name	EthIfVersionInfoApi		
Description	Enables / Disables version info API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00008 :**		
Name	EthIfVersionInfoApiMacro		
Description	Enables / Disables version info API macro implementation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00040 :**		
Name	EthIfWakeUpSupport		
Description	Configures if wakeup is supported or not.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.1.3 EthIfConfigSet

SWS Item	**ECUC_EthIf_00010 :**
Container Name	EthIfConfigSet
Description	Collecting container for all parameters with post-build configuration classes.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthIfController	1..*	This container contains the configuration of EthIfController.
EthIfFrameOwnerConfig	1..*	Configuration of Ethernet frame owner
EthIfPhysController	1..*	This container contains the configuration of EthIfPhysController. The usage of EthIfEthCtrlRef and EthIfWEthCtrlRef is exclusive OR.
EthIfRxIndicationConfig	1..*	Configuration of receive callback functions.
EthIfSwitch	0..*	This container contains the configuration of EthIfSwitches.
EthIfSwitchMgmtInfoIndicationConfig	0..*	Configuration of Switch Management callback function.
EthIfSwitchPortGroup	0..*	This container contains the configuration of EthIfSwitchPortGroups. If EthIfSwitchPortGroups are controlled by PNC one EthIfSwitchPortGroup per PNC shall exist. The host port shall be part of all EthIfSwitchPortGroups. The up link port of a master switch and the up link port of the slave switch shall be part of all EthIfSwitchPortGroups that contain EthSwtPorts belonging to the slave switch.
EthIfSwitchTimeStampIndicationConfig	0..*	Configuration of Switch timestamp indications.
EthIfTransceiver	0..*	This container contains the configuration of EthIfTransceiver. The usage of EthIfEthTrcvRef and EthIfWEthTrcvRefis exclusive OR.
EthIfTrcvLinkStateChgConfig	1..*	Specifies link state change callback function
EthIfTxConfirmationConfig	0..*	Configuration of transmit indication callback functions.

10.1.4 EthIfController

SWS Item	**ECUC_EthIf_00025 :**
Container Name	EthIfController
Description	This container contains the configuration of EthIfController.

Configuration Parameters

SWS Item	**ECUC_EthIf_00026 :**		
Name	EthIfCtrlIdx		
Description	This parameter provides a zero-based consecutive index of the Ethernet Communication Controllers. Upper layer BSW modules and the EthIf itself use this index to identify a Ethernet CC.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_EthIf_00032 :**		
Name	EthIfCtrlMtu		
Description	Specifies the maximum transmission unit (MTU) of the EthIfCtrl in [bytes]. Note: in case a VLAN tag is used for the EthIfCtrl, the MTU is 4 bytes smaller than the maximum payload size of an Ethernet frame which can be transmitted on the network.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	64 .. 9000		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: EthIfVlanId		

SWS Item	**ECUC_EthIf_00002 :**		
Name	EthIfMaxTxBufsTotal		
Description	Limits the total number of transmit buffers.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00029 :**		
Name	EthIfVlanId		
Description	A virtual-LAN is identified by this attribute according to IEEE 802.1Q.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4095		
Default value	--		
Post-Build Variant	true		

Multiplicity			
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_EthIf_00028 :**		
Name	EthIfEthTrcvRef		
Description	Reference to an Ethernet transceiver, which is handled by the Ethernet Interface.		
Multiplicity	0..1		
Type	Symbolic name reference to [EthIfTransceiver]		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_EthIf_00027 :**		
Name	EthIfPhysControllerRef		
Description	Reference to a physical Ethernet controller, which is handled by the Ethernet Interface.		
Multiplicity	1		
Type	Symbolic name reference to [EthIfPhysController]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_EthIf_00048 :**		
Name	EthIfSwitchRefOrPortGroupRef		
Description	The choice reference allows to configure either the EthIfController references an EthIfSwitch or an EthIfSwitchPortGroup. Reference to a EthIfSwitchPortGroup. In case port groups are controlled by PNC EthIfSwitchPortGroupRefSemantics shall have the value ETHIF_SWITCH_PORT_GROUP_LINK_INFO. In case port groups are controlled by the EhtIfController EthIfSwitchPortGroupRefSemantics shall have the value ETHIF_SWITCH_PORT_GROUP_CONTROL.		
Multiplicity	0..1		
Type	Choice reference to [EthIfSwitch , EthIfSwitchPortGroup]		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: no reference to EthIfSwitchPortGroup is only allowed if there are no EthIfSwitchPortGroups defined.		

No Included Containers

10.1.5 EthIfFrameOwnerConfig

SWS Item	**ECUC_EthIf_00011 :**
Container Name	EthIfFrameOwnerConfig
Description	Configuration of Ethernet frame owner
Configuration Parameters	

SWS Item	**ECUC_EthIf_00012 :**		
Name	EthIfFrameType		
Description	Selects the Ethernet frame type.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00013 :**		
Name	EthIfOwner		
Description	Selects the owner of an Ethernet frame type. The owner is a zero based index into the callback function configuration 'EthIfRxIndicationConfig'. I.e. an Ethernet frame of type IPv4 (0x800) at index 0 will call the first callback function configured in 'EthIfRxIndicationConfig'.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.1.6 EthIfPhysController

SWS Item	**ECUC_EthIf_00045 :**
Container Name	EthIfPhysController

Description	This container contains the configuration of EthIfPhysController. The usage of EthIfEthCtrlRef and EthIfWEthCtrlRef is exclusive OR.
Post-Build Variant Multiplicity	false
Configuration Parameters	

SWS Item	**ECUC_EthIf_00046 :**		
Name	EthIfPhysControllerIdx		
Description	This parameter provides a zero-based consecutive index of the physical Ethernet controllers. Upper layer BSW modules and the Ethernet Interface itself use this index to identify a physical Ethernet controller.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_EthIf_00047 :**		
Name	EthIfEthCtrlRef		
Description	Reference to a physical Ethernet controller, which is handled by a specific Ethernet controller driver.		
Multiplicity	0..1		
Type	Symbolic name reference to [EthCtrlConfig]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_EthIf_00073 :**		
Name	EthIfWEthCtrlRef		
Description	Reference to a physical Wireless Ethernet controller, which is handled by a specific Wireless Ethernet controller driver.		
Multiplicity	0..1		
Type	Symbolic name reference to [WEthCtrlConfig]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthIfPhysCtrlRxMainFunctionPriorityProcessing	0..*	Configuration of ingress FIFO based main function processing.

10.1.7 EthIfPhysCtrlRxMainFunctionPriorityProcessing

SWS Item	**ECUC_EthIf_00050 :**		
Container Name	EthIfPhysCtrlRxMainFunctionPriorityProcessing		
Description	Configuration of ingress FIFO based main function processing.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

SWS Item	**ECUC_EthIf_00052 :**		
Name	EthIfPhysCtrlRxIndicationIterations		
Description	Max number of Ethernet frames polled per main function invocation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00051 :**		
Name	EthIfPhysCtrlRxMainFunctionPeriod		
Description	Specifies the period of main function in seconds.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[-INF .. INF]		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00053 :**		
Name	EthIfPhysCtrlRxIngressFifoRef		
Description	Reference to the reception FIFO.		
Multiplicity	1		
Type	Symbolic name reference to [EthCtrlConfigIngressFifo]		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.1.8 EthIfRxIndicationConfig

SWS Item	**ECUC_EthIf_00014 :**
Container Name	EthIfRxIndicationConfig
Description	Configuration of receive callback functions.
Configuration Parameters	

SWS Item	**ECUC_EthIf_00015 :**		
Name	EthIfRxIndicationFunction		
Description	Specifies receive indication callback function.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.1.9 EthIfSwitch

SWS Item	**ECUC_EthIf_00036 :**
Container Name	EthIfSwitch
Description	This container contains the configuration of EthIfSwitches.
Configuration Parameters	

SWS Item	**ECUC_EthIf_00037 :**		
Name	EthIfSwitchIdx		
Description	This parameter provides a zero-based consecutive index of the Ethernet Interface Switches. Upper layer BSW modules and the EthIf itself use this index to identify a Ethernet Switch.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_EthIf_00038 :**
Name	EthIfSwitchRef
Description	Reference to a Ethernet Switch, which is handled by a specific Ethernet Switch driver.
Multiplicity	1
Type	Symbolic name reference to [EthSwtConfig]

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

No Included Containers

10.1.10 EthIfSwitchMgmtInfoIndicationConfig

SWS Item	**ECUC_EthIf_00065 :**		
Container Name	EthIfSwitchMgmtInfoIndicationConfig		
Description	Configuration of Switch Management callback function.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

SWS Item	**ECUC_EthIf_00067 :**		
Name	EthIfSwitchMgmtInfoIndicationFunction		
Description	Enables/Disables the ingress Switch management info indication redirected call to upper layers who registered for the call.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.1.11 EthIfSwitchTimeStampIndicationConfig

SWS Item	**ECUC_EthIf_00066 :**		
Container Name	EthIfSwitchTimeStampIndicationConfig		
Description	Configuration of Switch timestamp indications.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

SWS Item	**ECUC_EthIf_00068 :**
Name	EthIfSwitchEgressTimeStampIndicationFunction

Description	Enables/Disables to upper layers an egress timestamp indication function.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthIf_00069 :**		
Name	EthIfSwitchIngressTimeStampIndicationFunction		
Description	Enables/Disables to upper layers an ingress timestamp indication function.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.1.12 EthIfSwitchPortGroup

SWS Item	**ECUC_EthIf_00057 :**
Container Name	EthIfSwitchPortGroup
Description	This container contains the configuration of EthIfSwitchPortGroups. If EthIfSwitchPortGroups are controlled by PNC one EthIfSwitchPortGroup per PNC shall exist. The host port shall be part of all EthIfSwitchPortGroups. The up link port of a master switch and the up link port of the slave switch shall be part of all EthIfSwitchPortGroups that contain EthSwtPorts belonging to the slave switch.
Configuration Parameters	

SWS Item	**ECUC_EthIf_00058 :**	
Name	EthIfSwitchPortGroupIdx	
Description	This parameter provides a zero-based consecutive index of the Ethernet Switch Port Groups. Upper layer BSW modules and the EthIf itself use this index to identify an Ethernet Switch Port Group.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 255	

Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_EthIf_00059 :**		
Name	EthIfSwitchPortGroupRefSemantics		
Description	Defines how the EthIfSwitchRefOrPortGroupRef refering to a EthIfSwitchPortGroup shall be interpreted.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	ETHIF_SWITCH_PORT_GROUP_CONTROL	Used in case all ports in this group are controlled by the EthIf Controller.	
	ETHIF_SWITCH_PORT_GROUP_LINK_INFO	Used in case all ports in this group are controlled by EthIf_SwitchPortGroupRequestMode.	
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: only valid if a EthIfSwitchRefOrPortGroupRef refers to the EthIfSwitchPortGroup.		

SWS Item	**ECUC_EthIf_00060 :**		
Name	EthIfPortRef		
Description	Reference to an Ethernet Switch Port.		
Multiplicity	1..*		
Type	Symbolic name reference to [EthSwtPort]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.1.13 EthIfTransceiver

SWS Item	**ECUC_EthIf_00042 :**
Container Name	EthIfTransceiver
Description	This container contains the configuration of EthIfTransceiver. The usage of EthIfEthTrcvRef and EthIfWEthTrcvRefis exclusive OR.
Post-Build Variant Multiplicity	false
Configuration Parameters	

SWS Item	**ECUC_EthIf_00043 :**
Name	EthIfTransceiverIdx
Description	This parameter provides a zero-based consecutive index of the Ethernet

	transceivers. Upper layer BSW modules and the Ethernet Interface itself use this index to identify an Ethernet tranceiver.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_EthIf_00044 :**		
Name	EthIfEthTrcvRef		
Description	Reference to an Ethernet transceiver, which is handled by a specific Ethernet transceiver driver.		
Multiplicity	0..1		
Type	Symbolic name reference to [EthTrcvConfig]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_EthIf_00074 :**		
Name	EthIfWEthTrcvRef		
Description	Reference to an Wireless Ethernet transceiver, which is handled by a specific Wireless Ethernet transceiver driver.		
Multiplicity	0..1		
Type	Symbolic name reference to [WEthTrcvConfig]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.1.14 EthIfTrcvLinkStateChgConfig

SWS Item	**ECUC_EthIf_00018 :**
Container Name	EthIfTrcvLinkStateChgConfig
Description	Specifies link state change callback function
Configuration Parameters	

SWS Item	**ECUC_EthIf_00019 :**
Name	EthIfTrcvLinkStateChgFunction
Description	Specifies link state change callback function
Multiplicity	1
Type	EcucFunctionNameDef
Default value	--
maxLength	--
minLength	--

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

No Included Containers

10.1.15 EthIfTxConfirmationConfig

SWS Item	**ECUC_EthIf_00016 :**
Container Name	EthIfTxConfirmationConfig
Description	Configuration of transmit indication callback functions.
Configuration Parameters	

SWS Item	**ECUC_EthIf_00017 :**		
Name	EthIfTxConfirmationFunction		
Description	Specifies transmit indication callback function		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

11 Not applicable requirements

[SWS_EthIf_00999]
These requirements are not applicable to this specification (BSW00170).