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 Release	4.2.2

Document Change History

Release	Changed by	Change Description
4.2.2	AUTOSAR Release Management	• EthIf_TransceiverInit and EthIf_ControllerInit removed • Development Error Tracer renamed to Default Error Tracer
4.2.1	AUTOSAR Release Management	• Change from Synchronous to Asynchronous API • gPTP Timestamp Support • Ethernet Switch Support • Ethernet Wakeup Support
4.1.3	AUTOSAR Release Management	• Extended UL_RxIndication • Editorial changes
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
4.1.1	AUTOSAR Administration	• Remove „Commercial Off The Shelf“ use case • VLAN support • 1000MBit Ethernet support
4.0.3	AUTOSAR Administration	• Description of payload data in EthIf_Cbk_RxIndication adapted
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

Document Change History		
Release	**Changed by**	**Change Description**
3.1.4	AUTOSAR Administration	• Initial Release

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).⌋ ()

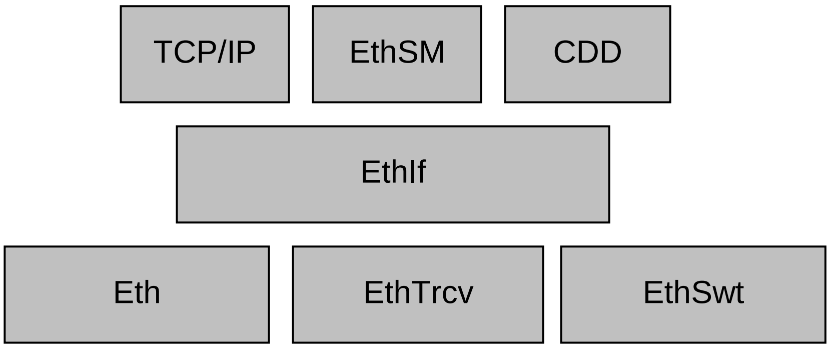


Figure 1: Ethernet stack module overview

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:***
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)
IP	Internet Protocol
MCG	Module Configuration Generator
MII	Media Independent Interface (standardized Interface provided by Ethernet controllers to access Ethernet transceivers)
TCP	Transmission Control Protocol
TCP/IP Stack	Ethernet communication stack
VLAN	Virtual Local Area Network

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 C Implementation Rules
AUTOSAR_TR_CImplementationRules.pdf

[17] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

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

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

3.2 Related standards and norms

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

[21] IEEE 802.3-2006

[22] IEEE 802.1Q-2011

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [20] (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:

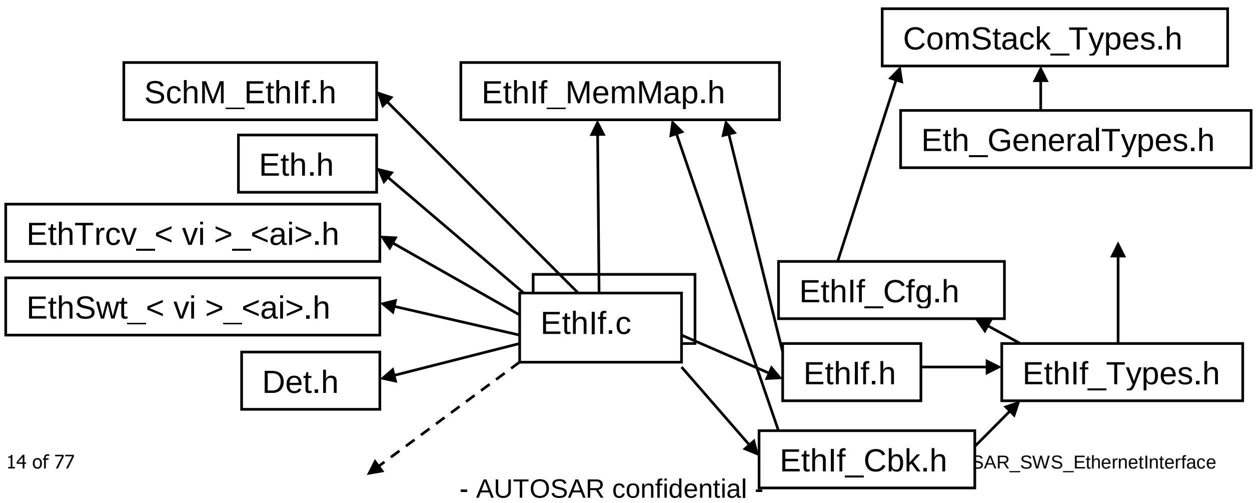
- EthSwt_<vendorID>_<Vendor specific name><driver abbreviation>.h for services and type definitions of the EthSwt (e.g.: EthSwt_99_Ext1.h).⌋ (SRS_BSW_00436)

[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



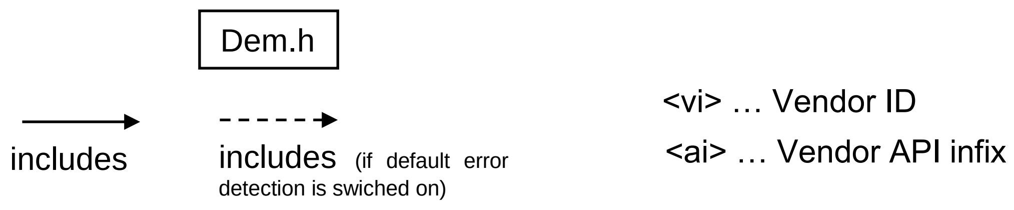


Figure 2: Ethernet Interface file structure

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_EthIf_00003
-	-	SWS_EthIf_00004
-	-	SWS_EthIf_00005
-	-	SWS_EthIf_00006
-	-	SWS_EthIf_00007
-	-	SWS_EthIf_00008
-	-	SWS_EthIf_00009
-	-	SWS_EthIf_00010
-	-	SWS_EthIf_00011
-	-	SWS_EthIf_00012
-	-	SWS_EthIf_00013
-	-	SWS_EthIf_00014
-	-	SWS_EthIf_00017
-	-	SWS_EthIf_00023
-	-	SWS_EthIf_00024
-	-	SWS_EthIf_00025
-	-	SWS_EthIf_00027
-	-	SWS_EthIf_00034
-	-	SWS_EthIf_00035
-	-	SWS_EthIf_00036
-	-	SWS_EthIf_00037
-	-	SWS_EthIf_00038
-	-	SWS_EthIf_00039
-	-	SWS_EthIf_00040
-	-	SWS_EthIf_00041
-	-	SWS_EthIf_00042
-	-	SWS_EthIf_00043
-	-	SWS_EthIf_00044
-	-	SWS_EthIf_00050
-	-	SWS_EthIf_00051
-	-	SWS_EthIf_00052
-	-	SWS_EthIf_00053
-	-	SWS_EthIf_00054
-	-	SWS_EthIf_00055
-	-	SWS_EthIf_00056
-	-	SWS_EthIf_00057

-	-	SWS_EthIf_00058
-	-	SWS_EthIf_00059
-	-	SWS_EthIf_00060
-	-	SWS_EthIf_00061
-	-	SWS_EthIf_00062
-	-	SWS_EthIf_00063
-	-	SWS_EthIf_00064
-	-	SWS_EthIf_00065
-	-	SWS_EthIf_00066
-	-	SWS_EthIf_00067
-	-	SWS_EthIf_00068
-	-	SWS_EthIf_00069
-	-	SWS_EthIf_00070
-	-	SWS_EthIf_00071
-	-	SWS_EthIf_00072
-	-	SWS_EthIf_00073
-	-	SWS_EthIf_00074
-	-	SWS_EthIf_00075
-	-	SWS_EthIf_00076
-	-	SWS_EthIf_00077
-	-	SWS_EthIf_00078
-	-	SWS_EthIf_00079
-	-	SWS_EthIf_00080
-	-	SWS_EthIf_00081
-	-	SWS_EthIf_00082
-	-	SWS_EthIf_00085
-	-	SWS_EthIf_00086
-	-	SWS_EthIf_00087
-	-	SWS_EthIf_00088
-	-	SWS_EthIf_00089
-	-	SWS_EthIf_00090
-	-	SWS_EthIf_00091
-	-	SWS_EthIf_00092
-	-	SWS_EthIf_00093
-	-	SWS_EthIf_00094
-	-	SWS_EthIf_00095
-	-	SWS_EthIf_00096
-	-	SWS_EthIf_00097
-	-	SWS_EthIf_00098

-	-	SWS_EthIf_00099
-	-	SWS_EthIf_00100
-	-	SWS_EthIf_00101
-	-	SWS_EthIf_00102
-	-	SWS_EthIf_00103
-	-	SWS_EthIf_00104
-	-	SWS_EthIf_00105
-	-	SWS_EthIf_00106
-	-	SWS_EthIf_00107
-	-	SWS_EthIf_00108
-	-	SWS_EthIf_00109
-	-	SWS_EthIf_00111
-	-	SWS_EthIf_00112
-	-	SWS_EthIf_00113
-	-	SWS_EthIf_00114
-	-	SWS_EthIf_00116
-	-	SWS_EthIf_00117
-	-	SWS_EthIf_00118
-	-	SWS_EthIf_00123
-	-	SWS_EthIf_00124
-	-	SWS_EthIf_00127
-	-	SWS_EthIf_00128
-	-	SWS_EthIf_00129
-	-	SWS_EthIf_00130
-	-	SWS_EthIf_00132
-	-	SWS_EthIf_00134
-	-	SWS_EthIf_00135
-	-	SWS_EthIf_00136
-	-	SWS_EthIf_00137
-	-	SWS_EthIf_00138
-	-	SWS_EthIf_00139
-	-	SWS_EthIf_00140
-	-	SWS_EthIf_00141
-	-	SWS_EthIf_00142
-	-	SWS_EthIf_00143
-	-	SWS_EthIf_00144
-	-	SWS_EthIf_00145
-	-	SWS_EthIf_00146
-	-	SWS_EthIf_00147

-	-	SWS_EthIf_00149
-	-	SWS_EthIf_00150
-	-	SWS_EthIf_00151
-	-	SWS_EthIf_00152
-	-	SWS_EthIf_00153
-	-	SWS_EthIf_00154
-	-	SWS_EthIf_00155
-	-	SWS_EthIf_00156
-	-	SWS_EthIf_00157
-	-	SWS_EthIf_00158
-	-	SWS_EthIf_00159
-	-	SWS_EthIf_00160
-	-	SWS_EthIf_00161
-	-	SWS_EthIf_00162
-	-	SWS_EthIf_00164
-	-	SWS_EthIf_00165
-	-	SWS_EthIf_00166
-	-	SWS_EthIf_00167
-	-	SWS_EthIf_00168
-	-	SWS_EthIf_00169
-	-	SWS_EthIf_00170
-	-	SWS_EthIf_00171
-	-	SWS_EthIf_00172
-	-	SWS_EthIf_00173
-	-	SWS_EthIf_00174
-	-	SWS_EthIf_00175
-	-	SWS_EthIf_00176
-	-	SWS_EthIf_00177
-	-	SWS_EthIf_00178
-	-	SWS_EthIf_00179
-	-	SWS_EthIf_00180
-	-	SWS_EthIf_00181
-	-	SWS_EthIf_00182
-	-	SWS_EthIf_00183
-	-	SWS_EthIf_00184
-	-	SWS_EthIf_00185
-	-	SWS_EthIf_00186
-	-	SWS_EthIf_00187
-	-	SWS_EthIf_00188

-	-	SWS_EthIf_00189
-	-	SWS_EthIf_00190
-	-	SWS_EthIf_00191
-	-	SWS_EthIf_00192
-	-	SWS_EthIf_00193
-	-	SWS_EthIf_00194
-	-	SWS_EthIf_00195
-	-	SWS_EthIf_00196
-	-	SWS_EthIf_00197
-	-	SWS_EthIf_00198
-	-	SWS_EthIf_00199
-	-	SWS_EthIf_00200
-	-	SWS_EthIf_00201
-	-	SWS_EthIf_00202
-	-	SWS_EthIf_00203
-	-	SWS_EthIf_00204
-	-	SWS_EthIf_00205
-	-	SWS_EthIf_00206
-	-	SWS_EthIf_00207
-	-	SWS_EthIf_00208
-	-	SWS_EthIf_00209
-	-	SWS_EthIf_00210
-	-	SWS_EthIf_00211
-	-	SWS_EthIf_00212
-	-	SWS_EthIf_00213
-	-	SWS_EthIf_00214
-	-	SWS_EthIf_00215
-	-	SWS_EthIf_00216
-	-	SWS_EthIf_00217
-	-	SWS_EthIf_00218
-	-	SWS_EthIf_00219
-	-	SWS_EthIf_00220
-	-	SWS_EthIf_00221
-	-	SWS_EthIf_00222
-	-	SWS_EthIf_00223
-	-	SWS_EthIf_00224
-	-	SWS_EthIf_00228
-	-	SWS_EthIf_00229
-	-	SWS_EthIf_00230

-	-	SWS_EthIf_00231
-	-	SWS_EthIf_00232
-	-	SWS_EthIf_00233
-	-	SWS_EthIf_00234
-	-	SWS_EthIf_00235
-	-	SWS_EthIf_00236
-	-	SWS_EthIf_00238
-	-	SWS_EthIf_00239
-	-	SWS_EthIf_00240
-	-	SWS_EthIf_00241
-	-	SWS_EthIf_00242
-	-	SWS_EthIf_00243
-	-	SWS_EthIf_00244
-	-	SWS_EthIf_00246
-	-	SWS_EthIf_00247
-	-	SWS_EthIf_00248
-	-	SWS_EthIf_00250
-	-	SWS_EthIf_00252
-	-	SWS_EthIf_00253
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

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.

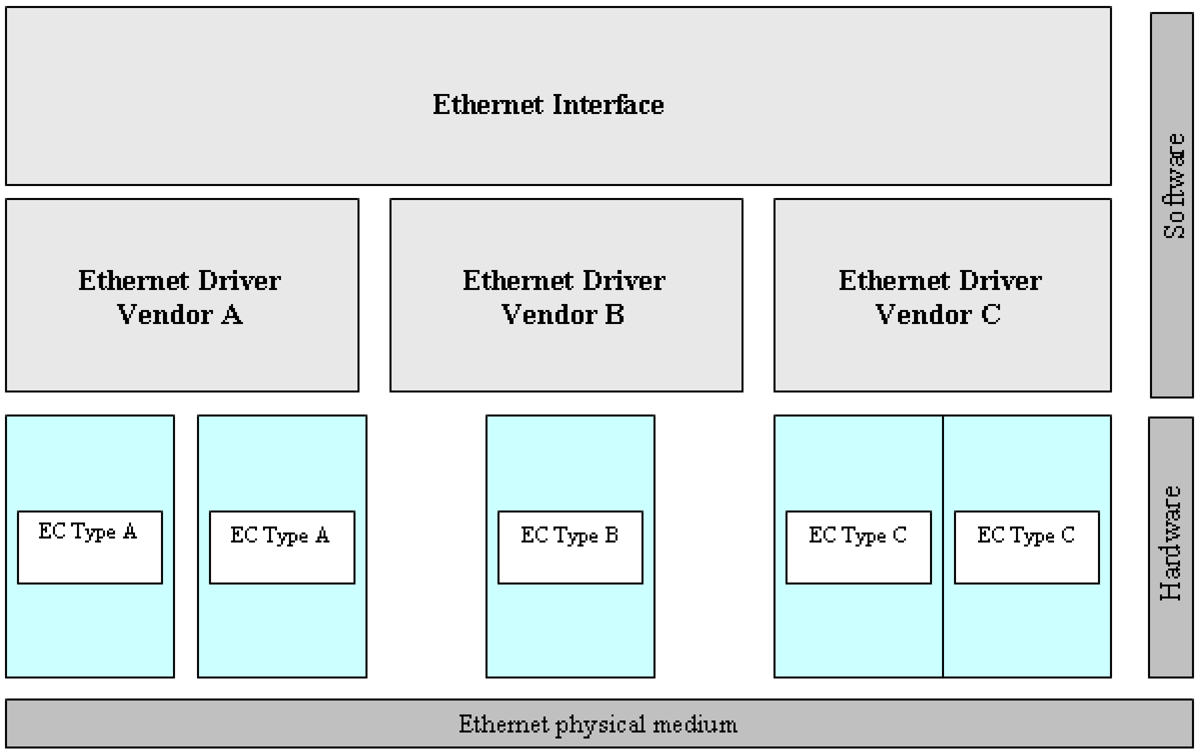


Figure 3: Basic Structure of the Ethernet BSW stack

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.

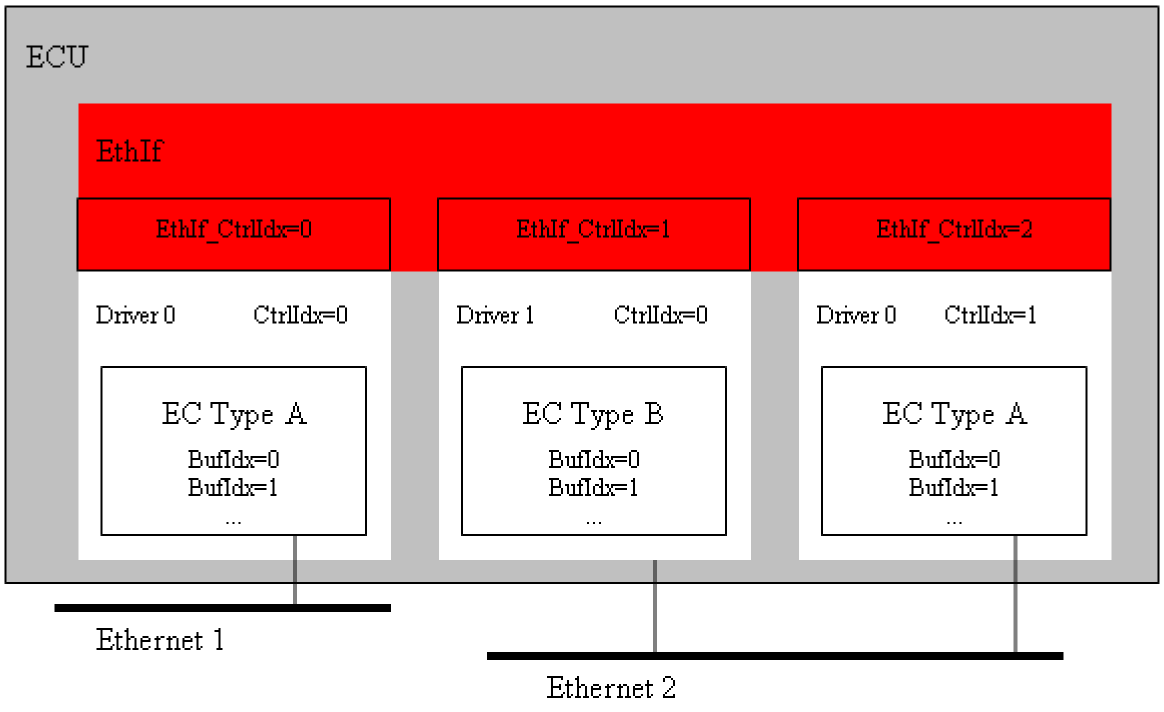


Figure 4: Ethernet Interface controller indexing scheme

[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

The EthIf introduces a indexing scheme for EthSwtes. All managed EthSwtes are collected in the EthIfSwitch configuration container. Each EthSwt is given a zero-based consecutive configuration index in the configuration of the EthIf by the parameter EthIfSwitchIdx.
Each EthSwt driver configuration keeps an own zero-based configuration index locally and the EthIf translates the EthIfSwitchIdx to the respective EthSwtIdx value.

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

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.

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 default 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_00009] ⌈
The Ethernet Interface module implementation shall conform to the HIS subset of the MISRA C Standard (see document [16]). ⌋()

[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: if no VLAN ID is set the virtual controller represents the untagged VLAN. ⌋()

[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.

7.2 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 index	Default Error	ETHIF_E_INV_TRCV_IDX	0x02
EthIf module was not initialized	Default Error	ETHIF_E_NOT_INITIALIZED	0x03
Invalid pointer in parameter list	Default Error	ETHIF_E_PARAM_POINTER	0x04
Invalid parameter	Default Error	ETHIF_E_INV_PARAM	0x05
Initialization failure	Default Error	ETHIF_E_INIT_FAILED	0x06

⌋()

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
Dem	Dem_EventIdType
	Dem_EventStatusType
EcuM	EcuM_WakeupSourceType
EthSwt	EthSwt_MacVlanType
Eth_GeneralTypes	EthTrcv_BaudRateType
	EthTrcv_DuplexModeType
	EthTrcv_LinkStateType
	EthTrcv_ModeType
	EthTrcv_WakeupModeType
	Eth_BufIdxType
	Eth_DataType
	Eth_FilterActionType
	Eth_FrameType
	Eth_ModeType
	Eth_RateRatioType
	Eth_RxStatusType
	Eth_TimeIntDiffType
	Eth_TimeStampQualType
	Eth_TimeStampType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋()

8.2 Type definitions

[SWS_EthIf_00152] ⌈
EthIf.h shall include Eth_GeneralTypes.h for the include of 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.	

⌋()

8.3 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 default 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 default 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 respective Ethernet Controller Driver. ⌋()

[SWS_EthIf_00036] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_EthIf_00037] ⌈
If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error 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 respective Ethernet Controller Driver. ⌋()

[SWS_EthIf_00041] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ⌋()

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

[SWS_EthIf_00043] ⌈
If default error detection is enabled: the function shall check the parameter CtrlModePtr for being valid. If the check fails, the function shall raise the default error 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_SetTransceiverMode

[SWS_EthIf_00050] ⌈

Service name:	EthIf_SetTransceiverMode	
Syntax:	`Std_ReturnType EthIf_SetTransceiverMode( uint8 CtrlIdx, EthTrcv_ModeType TrcvMode )`	
Service ID[hex]:	0x06	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	TrcvMode	ETHTRCV_MODE_DOWN: disable the transceiver ETHTRCV_MODE_ACTIVE: enable the transceiver
Parameters	None	

(inout):		
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: transceiver mode could not be changed
Description:	Enable / disable the indexed transceiver	

⌋()
[SWS_EthIf_00051] ⌈
The function EthIf_SetTransceiverMode shall forward the call to function EthTrcv_SetTransceiverMode of the respective Ethernet Transceiver Driver. ⌋()

[SWS_EthIf_00052] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_EthIf_00053] ⌈
If default error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the default error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_EthIf_00054] ⌈
Caveat: The function requires previous initialization (EthIf_Init). ⌋()

8.3.5 EthIf_GetTransceiverMode

[SWS_EthIf_00055] ⌈

Service name:	EthIf_GetTransceiverMode	
Syntax:	`Std_ReturnType EthIf_GetTransceiverMode( uint8 CtrlIdx, EthTrcv_ModeType* TrcvModePtr )`	
Service ID[hex]:	0x07	
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):	TrcvModePtr	ETHTRCV_MODE_DOWN: the transceiver is disabled ETHTRCV_MODE_ACTIVE: the transceiver is enabled
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: transceiver mode could not be obtained
Description:	Obtain state of the indexed transceiver	

⌋()
[SWS_EthIf_00056] ⌈
The function EthIf_GetTransceiverMode shall forward the call to function EthTrcv_GetTransceiverMode of the respective Ethernet Transceiver Driver. ⌋()

[SWS_EthIf_00057] ⌈

If default 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 default error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_EthIf_00058] ⌈
If default error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the default error ETHIF_E_INV_TRCV_IDX otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_EthIf_00059] ⌈
If default error detection is enabled: the function shall check the parameter TrcvModePtr for being valid. If the check fails, the function shall raise the default error ETHIF_E_PARAM_POINTER otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_EthIf_00060] ⌈
Caveat: The function requires previous initialization (EthIf_Init). ⌋()

8.3.6 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 respective Ethernet Transceiver Driver. ⌋()

[SWS_EthIf_00235] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_EthIf_00236] ⌈
If default error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the default 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)

8.3.7 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 respective Ethernet Transceiver Driver. ⌋()

[SWS_EthIf_00240] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ⌋()

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

[SWS_EthIf_00242] ⌈
If default error detection is enabled: the function shall check the parameter TrcvWakeupModePtr for being valid. If the check fails, the function shall raise the default 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. ⌋()

8.3.8 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 default 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 default error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ⌋()

[SWS_EthIf_00247] ⌈
If default error detection is enabled: the function shall check the parameter WakeupSource for being valid. If the check fails, the function shall raise the default 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)

8.3.9 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 default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

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

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

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

8.3.10 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 default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

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

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

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

8.3.11 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 default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

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

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

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

8.3.12 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 EthIf_GetPortMacAddr shall be pre compile time configurable On/Off by the configuration parameter: EthIfGetPortMacAddrApi. ⌋()

[SWS_EthIf_00193] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

[SWS_EthIf_00194] ⌈
If default 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 default error ETHIF_E_PARAM_POINTER. ⌋()

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

8.3.13 EthIf_GetArlTable

[SWS_EthIf_00196] ⌈

Service name:	EthIf_GetArlTable	
Syntax:	`Std_ReturnType EthIf_GetArlTable( uint8 SwitchIdx, EthSwt_MacVlanType[]* ArlTable )`	
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):	None	
Parameters (out):	ArlTable	Returns the ARL table of the switch consisting of a list of structs with MAC-address, VLAN-ID and port.
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: switch port could not be initialized
Description:	Obtains the address resolution table of a switch	

⌋()

[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_00198] ⌈
The function EthIf_GetArlTable shall be pre compile time configurable On/Off by the configuration parameter: EthIfGetArlTable. ⌋()

[SWS_EthIf_00199] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

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

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

8.3.14 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 EthIf_GetBufferLevel shall be pre compile time configurable On/Off by the configuration parameter: EthIfGetBufferLevelApi. ⌋()

[SWS_EthIf_00205] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

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

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

8.3.15 EthIf_GetDropCount

[SWS_EthIf_00208] ⌈

Service name:	EthIf_GetDropCount	
Syntax:	`Std_ReturnType EthIf_GetDropCount( uint8 SwitchIdx, uint32[]* DropCount )`	
Service ID[hex]:	0x2b	
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):	DropCount	The interpretation of this list of values is switch dependent
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: switch port could not be initialized
Description:	Reads a list with drop counter values of the corresponding switch. The meaning of these values is switch dependent and can include values like 1.) dropped packets due to buffer overrun, 2.) dropped packets due to CRC errors, etc.	

⌋()

[SWS_EthIf_00209] ⌈
The function EthIf_GetDropCount shall read a list of values of the switch. ⌋()

[SWS_EthIf_00210] ⌈
The function EthIf_GetDropCount shall be pre compile time configurable On/Off by the configuration parameter: EthIfGetDropCount. ⌋()

[SWS_EthIf_00211] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

[SWS_EthIf_00212] ⌈
If default error detection is enabled: the function shall check the parameter DropCount for being valid. If the check fails, the function shall raise the default error ETHIF_E_PARAM_POINTER. ⌋()

[SWS_EthIf_00213] ⌈
Caveat: The function requires previous initialization (EthIf_Init). ⌋()

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

[SWS_EthIf_00217] ⌈
If default 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 default 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 EthIf_ResetConfiguration shall be pre compile time configurable On/Off by the configuration parameter: EthIfResetConfigurationApi. ⌋()

[SWS_EthIf_00222] ⌈
If default 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 default 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, than an the remaining bits will be filled with 0.	

⌋()
[SWS_EthIf_00155] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

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

[SWS_EthIf_00157] ⌈
If default error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the default error 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, uint8 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 â€œdisableâ€• functionality, due to the fact, that the message type is always "time stamped" by network design.	

⌋()
[SWS_EthIf_00161] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

[SWS_EthIf_00162] ⌈
If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error 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:	`void EthIf_GetEgressTimeStamp(`

	uint8 CtrlIdx, uint8 BufIdx, Eth_TimeStampQualType* timeQualPtr, Eth_TimeStampType* timeStampPtr)
Service ID[hex]:	0x24
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):	timeQualPtr — quality of HW time stamp, e.g. based on current drift timeStampPtr — current time stamp
Return value:	None
Description:	Reads back the egress time stamp on a dedicated message object. It must be called within the TxConfirmation() function.

⌋()

[SWS_EthIf_00167] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

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

[SWS_EthIf_00169] ⌈
If default error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the default error 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:	void 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:	None	
Description:	Reads back the ingress time stamp on a dedicated message object. It must be called within the RxIndication() function.	

⌋()

[SWS_EthIf_00173] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

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

[SWS_EthIf_00175] ⌈
If default error detection is enabled: the function shall check the parameter DataPtr, timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the default error 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_SetCorrectionTime

[SWS_EthIf_00178] ⌈

Service name:	EthIf_SetCorrectionTime	
Syntax:	`void EthIf_SetCorrectionTime( uint8 CtrlIdx, const Eth_TimeIntDiffType* timeOffsetPtr, const Eth_RateRatioType* rateRatioPtr )`	
Service ID[hex]:	0x26	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the addresses ETH controller.
	timeOffsetPtr	offset between time stamp grandmaster and time stamp by local clock: (OriginTimeStampSync[FUP] – IngressTimeStampSync) + Pdelay
	rateRatioPtr	time elements to calculate and to modify the ratio of the frequency of

		the grandmaster in relation to the frequency of the Local Clock with: ratio = OriginTimeStampDelta / IngressTimeStampDelta
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Allows the Time Slave to adjust the local ETH Reference clock in HW.	

⌋()

[SWS_EthIf_00179] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

[SWS_EthIf_00180] ⌈
If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETHIF_E_INV_CTRL_IDX. ⌋()

[SWS_EthIf_00181] ⌈
If default error detection is enabled: the function shall check the parameter timeOffsetPtr and timeRatioPtr for being valid. If the check fails, the function shall raise the default error ETHIF_E_PARAM_POINTER. ⌋()

[SWS_EthIf_00182] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGlobalTimeSupport. ⌋()

[SWS_EthIf_00183] ⌈
Caveat: The function requires previous initialization (EthIf_Init). ⌋()

8.3.23 EthIf_SetGlobalTime

[SWS_EthIf_00184] ⌈

Service name:	EthIf_SetGlobalTime	
Syntax:	`Std_ReturnType EthIf_SetGlobalTime( uint8 CtrlIdx, const Eth_TimeStampType* timeStampPtr )`	
Service ID[hex]:	0x27	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the addresses ETH controller.
	timeStampPtr	new time stamp
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Allows the Time Master to adjust the global ETH Reference clock in HW. We can use this method to set a global time base on ETH in general or to synchronize the global ETH time base with another time base, e.g. FlexRay.	

⌋()

[SWS_EthIf_00185] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

[SWS_EthIf_00186] ⌈
If default error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the default error ETHIF_E_INV_CTRL_IDX. ⌋()

[SWS_EthIf_00187] ⌈
If default error detection is enabled: the function shall check the parameter timeStampPtr for being valid. If the check fails, the function shall raise the default error ETHIF_E_PARAM_POINTER. ⌋()

[SWS_EthIf_00188] ⌈
The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGlobalTimeSupport. ⌋()

[SWS_EthIf_00189] ⌈
Caveat: The function requires previous initialization (EthIf_Init). ⌋()

8.3.24 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:	Non 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 default 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 default error ETHIF_E_NOT_INITIALIZED and return BUFREQ_E_NOT_OK. ⌋()

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

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

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

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

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

8.3.25 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:	Non Reentrant	
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 default 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 default error ETHIF_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ⌋()

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

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

[SWS_EthIf_00080] ⌈
If default error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the default error 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). ⌋()

8.3.26 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 default error detection is enabled: the function shall check the parameter VersionInfoPtr for being valid. If the check fails, the function shall raise the default error 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 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 default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

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

[SWS_EthIf_00088] ⌈
If default error detection is enabled: the function shall check the parameter DataPtr for being valid. If the check fails, the function shall raise the default 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 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 )`
Service ID[hex]:	0x11
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 transmitted buffer
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Confirms frame transmission by the indexed controller	

⌋()

[SWS_EthIf_00092] ⌈
If default 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 default error ETHIF_E_NOT_INITIALIZED. ⌋()

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

[SWS_EthIf_00094] ⌈
If default error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the default error 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 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 CtrlIdx, EthTrcv_ModeType TrcvMode )`	
Service ID[hex]:	0x0f	
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 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.	

⌋()
[SWS_EthIf_00253] ⌈
The function shall call EthSM_TrcvModeIndication. ⌋()

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 default 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 default 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_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 default 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 default 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. ⌋()

8.6 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***
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. OBD Events Suppression shall be ignored for this computation.
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 specified controller
Eth_ReadMii	Reads a transceiver register
Eth_Receive	Triggers frame reception
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.
EthSM_TrcvModeIndication	Called when mode has been read out. Either triggered by previous EthIf_GetTransceiverMode or by EthIf_SetTransceiverMode call. Can directly be called within the trigger functions.
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

⌋()

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***
Det_ReportError	Service to report development errors.
Eth_GetCounterState	Reads the value of a counter specified with its memory offset
EthTrcv_GetBaudRate	Obtains the baud rate of the indexed transceiver
EthTrcv_StartAutoNegotiation	Restarts the negotiation of the transmission parameters used by the indexed transceiver
SchM_Enter_EthIf	Invokes the SchM_Enter function to enter a module local exclusive area.
SchM_Exit_EthIf	Invokes the SchM_Exit function to exit an exclusive area.

⌋()

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:	<User>_TxConfirmation	

Syntax:	`void <User>_TxConfirmation(`
	`uint8 CtrlIdx,`
	`Eth_BufIdxType BufIdx`
	`)`	

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
Parameters (inout):	None	
Parameters (out):	None	
Return value:	void	--
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. ⌋()

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

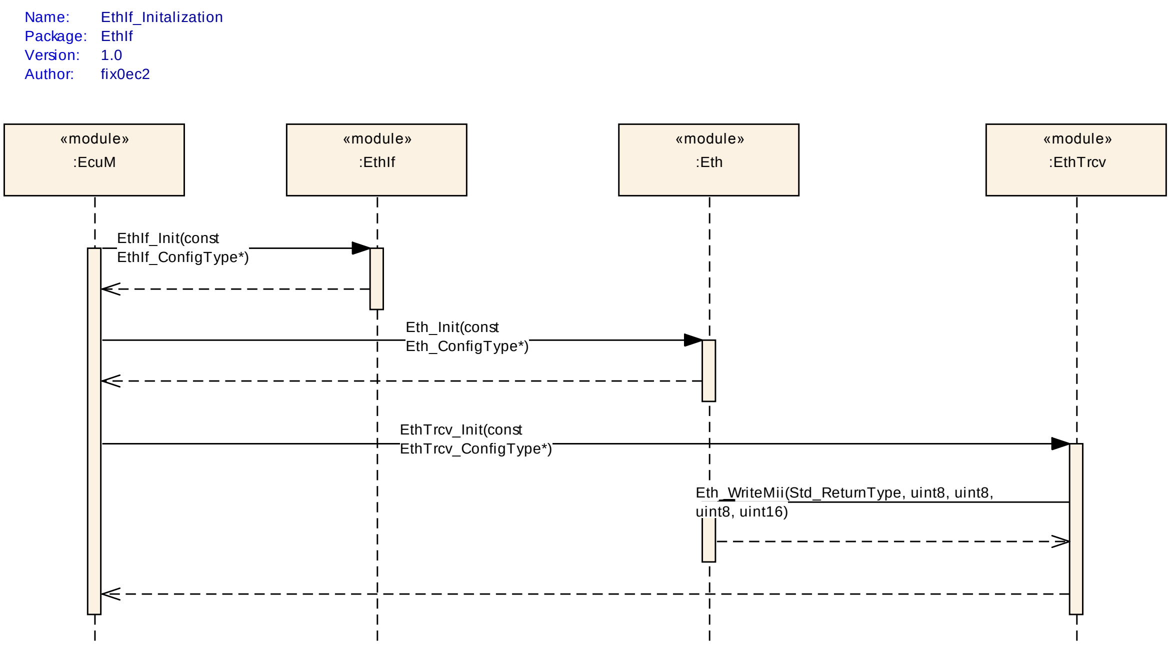

Name: EthIf_Initalization
Package: EthIf
Version: 1.0
Author: fix0ec2
«module» :EcuM
«module» :EthIf
«module» :Eth
«module» :EthTrcv
EthIf_Init(const EthIf_ConfigType*)
Eth_Init(const Eth_ConfigType*)
EthTrcv_Init(const EthTrcv_ConfigType*)
Eth_WriteMii(Std_ReturnType, uint8, uint8, uint8, uint16)

Figure 5: Initialization

9.2 Communication Initialization

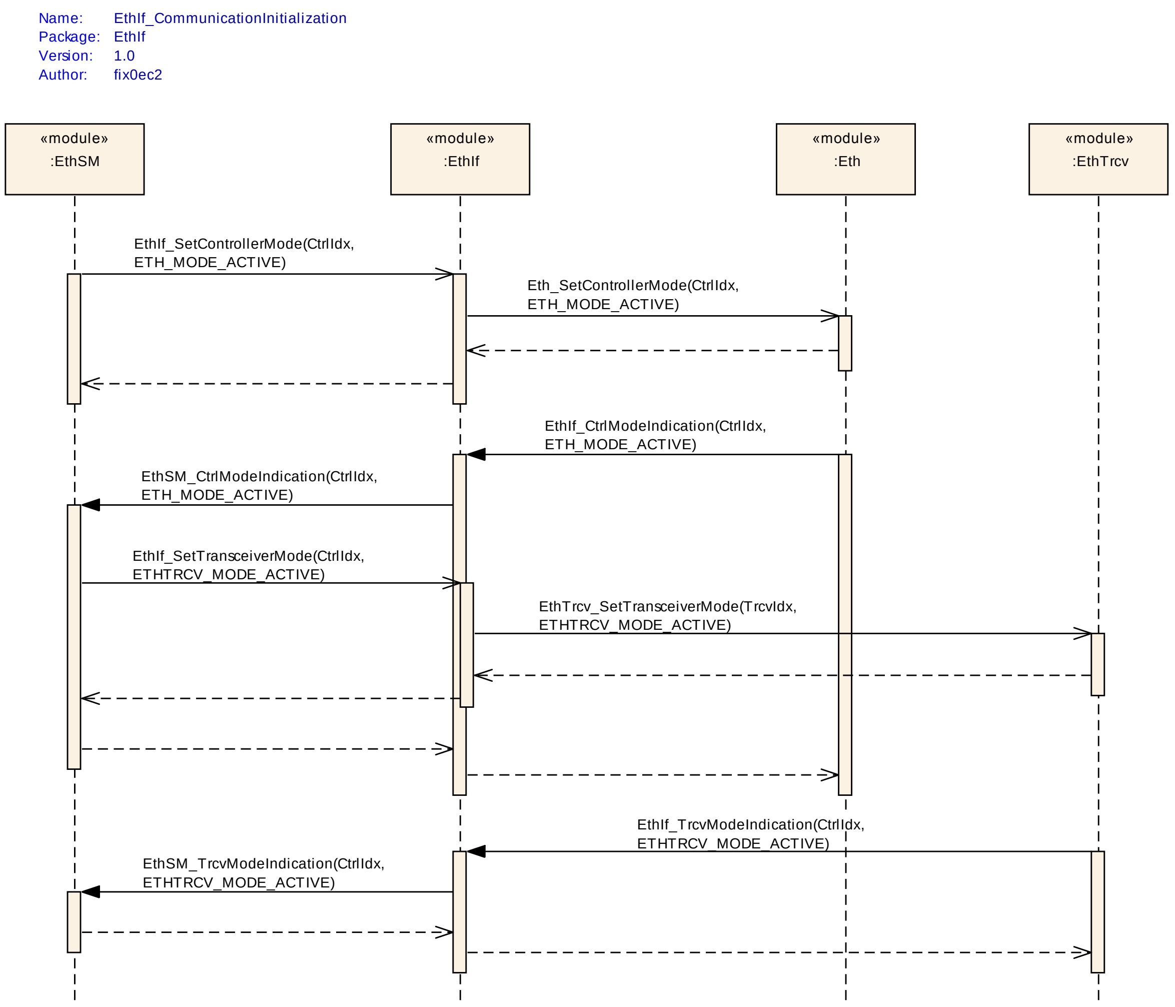

Name: EthIf_CommunicationInitialization
Package: EthIf
Version: 1.0
Author: fix0ec2
«module» :EthSM
«module» :EthIf
«module» :Eth
«module» :EthTrcv
EthIf_SetControllerMode(CtrlIdx, ETH_MODE_ACTIVE)
Eth_SetControllerMode(CtrlIdx, ETH_MODE_ACTIVE)
EthIf_CtrlModeIndication(CtrlIdx, ETH_MODE_ACTIVE)
EthSM_CtrlModeIndication(CtrlIdx,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

Figure 6: Communication Initialization

9.3 Data Transmission

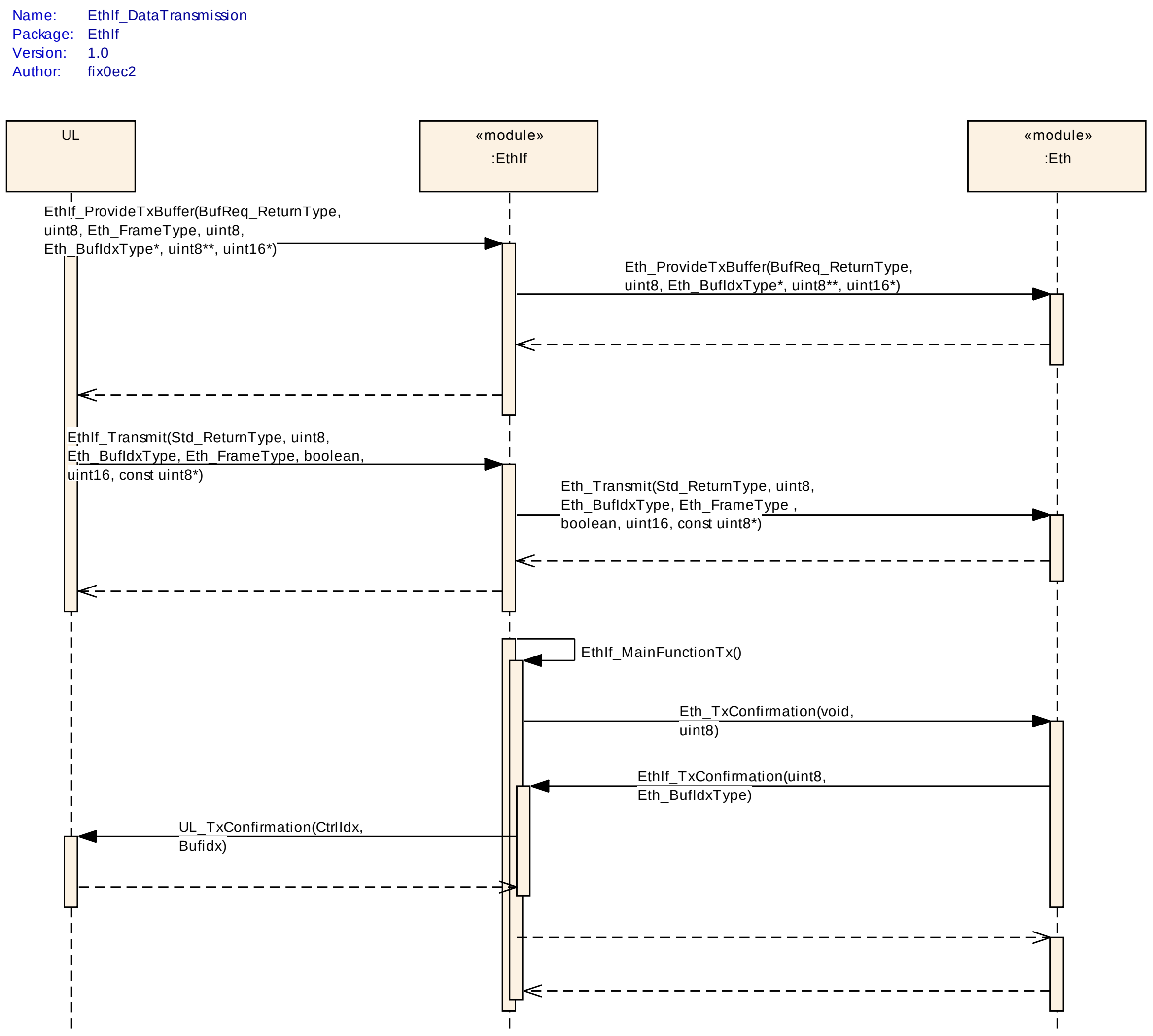

Name: EthIf_DataTransmission
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module» :EthIf
«module» :Eth
EthIf_ProvideTxBuffer(BufReq_ReturnType, uint8, Eth_FrameType, uint8, Eth_BufIdxType*, uint8**, uint16*)
Eth_ProvideTxBuffer(BufReq_ReturnType, uint8, Eth_BufIdxType*, uint8**, uint16*)
EthIf_Transmit(Std_ReturnType, uint8, Eth_BufIdxType, Eth_FrameType, boolean, uint16, const uint8*)
Eth_Transmit(Std_ReturnType, uint8, Eth_BufIdxType, Eth_FrameType , boolean, uint16, const uint8*)
EthIf_MainFunctionTx()
Eth_TxConfirmation(void, uint8)
EthIf_TxConfirmation(uint8, Eth_BufIdxType)
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_Cbk_TxConfirmation.

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

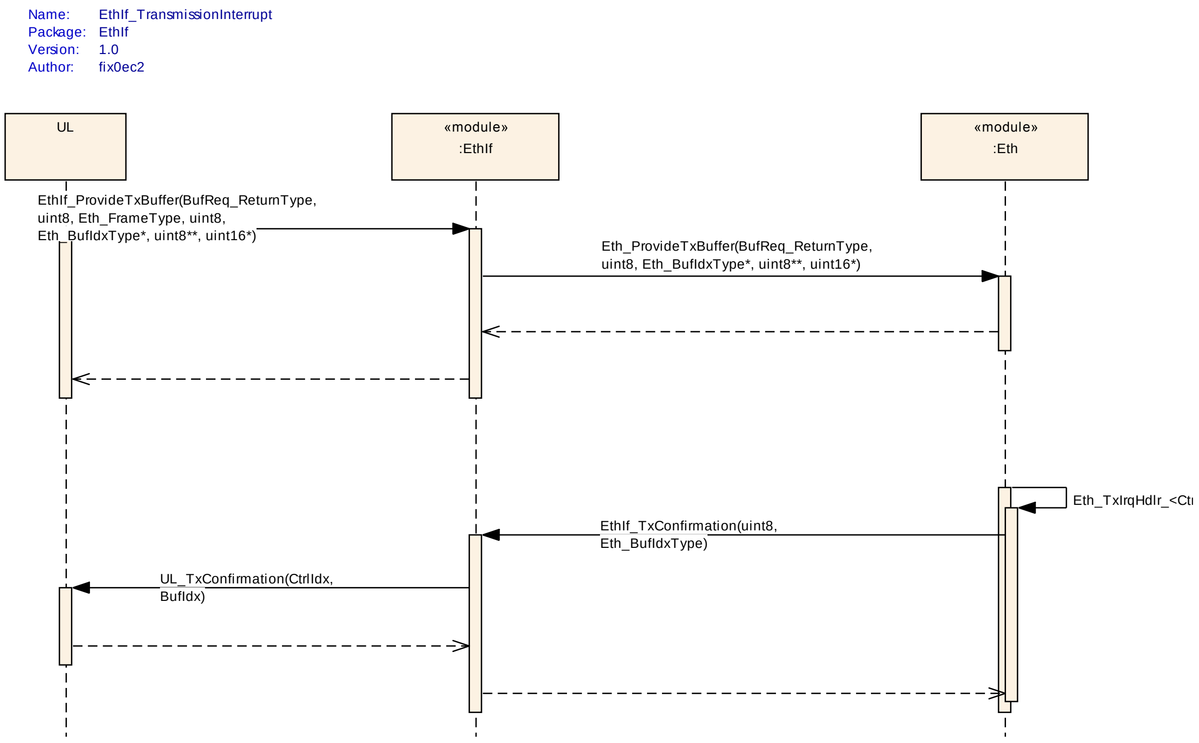

Name: EthIf_TransmissionInterrupt
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module» :EthIf
«module» :Eth
EthIf_ProvideTxBuffer(BufReq_ReturnType, uint8, Eth_FrameType, uint8, Eth_BufIdxType*, uint8**, uint16*)
Eth_ProvideTxBuffer(BufReq_ReturnType, uint8, Eth_BufIdxType*, uint8**, uint16*)
Eth_TxIrqHdlr_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

Figure 8: Frame Transmission in Interrupt Mode

9.4 Data Reception

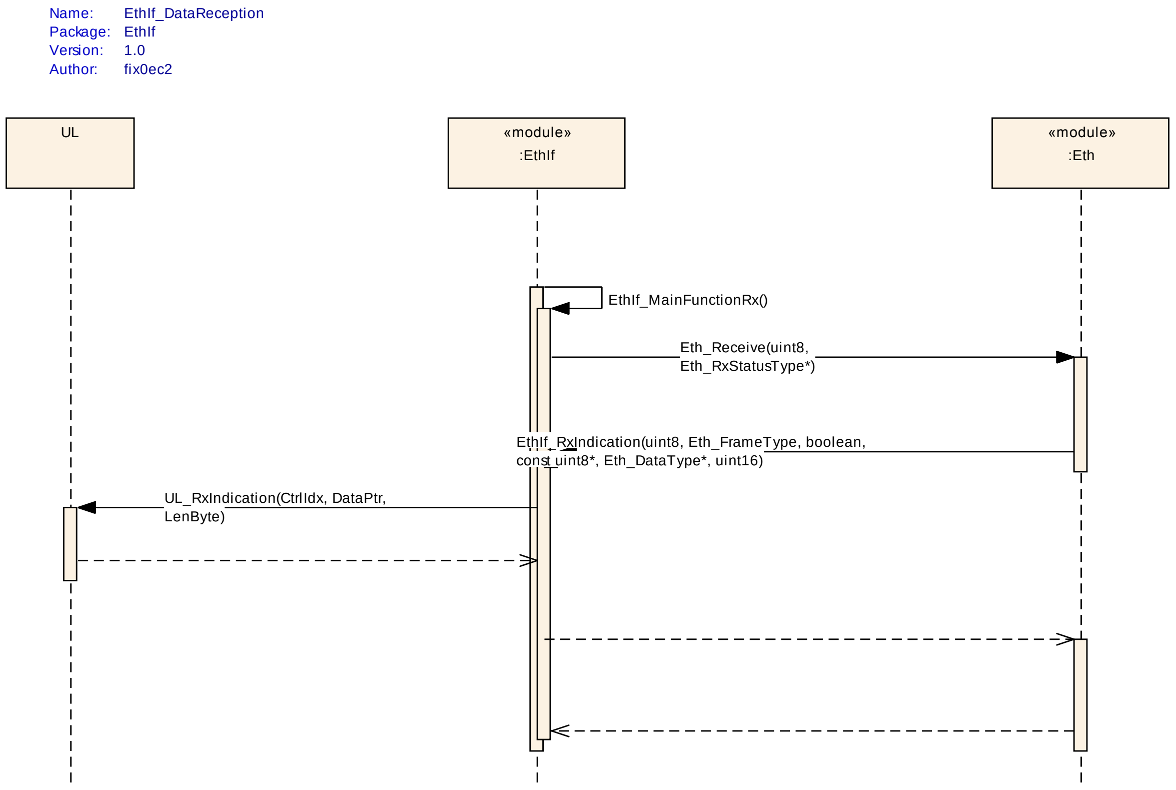

Name: EthIf_DataReception
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module» :EthIf
«module» :Eth
EthIf_MainFunctionRx()
Eth_Receive(uint8, Eth_RxStatusType*)
EthIf_RxIndication(uint8, Eth_FrameType, boolean, const uint8*, Eth_DataType*, uint16)
UL_RxIndication(CtrlIdx, DataPtr, LenByte)

Figure 9: Frame Reception in Polling Mode

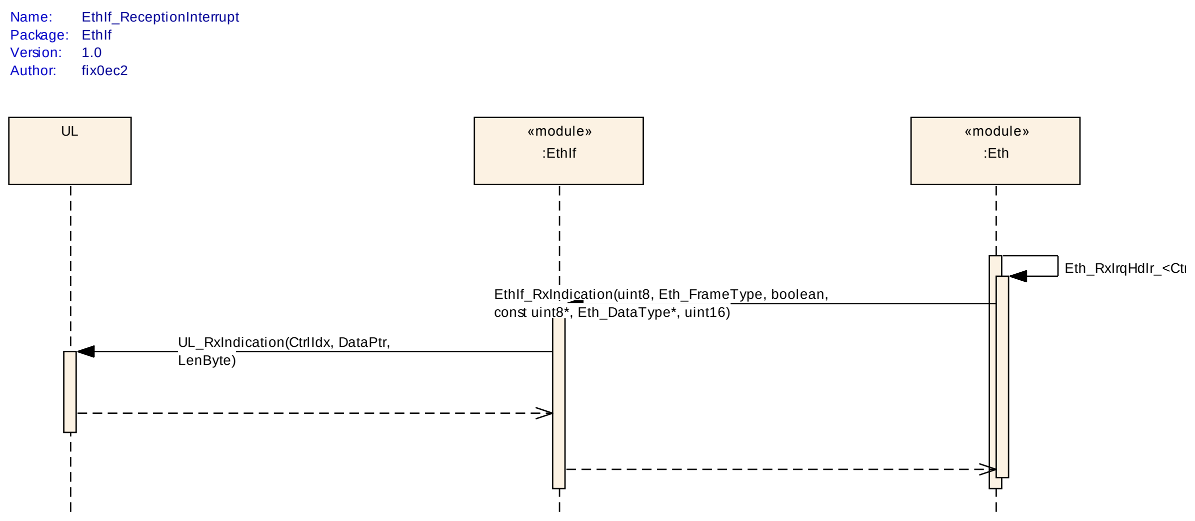

Name: EthIf_ReceptionInterrupt
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module» :EthIf
«module» :Eth
Eth_RxIrqHdlr_<Ct
EthIf_RxIndication(uint8, Eth_FrameType, boolean, const uint8*, Eth_DataType*, uint16)
UL_RxIndication(CtrlIdx, DataPtr, LenByte)

Figure 10: Frame Reception in Interrupt Mode

9.5 Link State Change

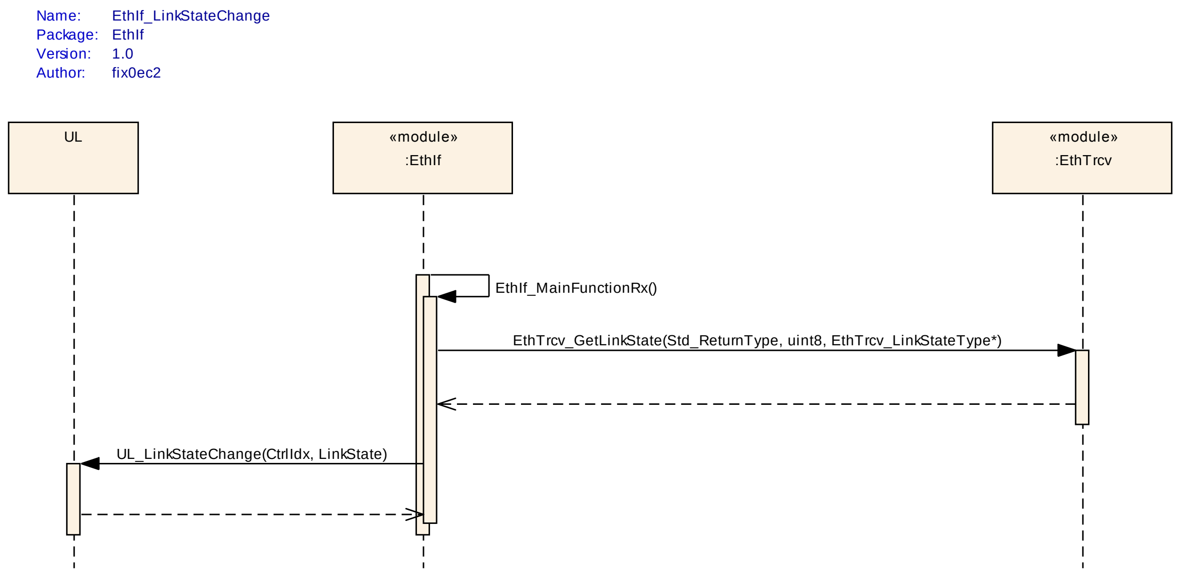

Name: EthIf_LinkStateChange
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module» :EthIf
«module» :EthTrcv
EthIf_MainFunctionRx()
EthTrcv_GetLinkState(Std_ReturnType, uint8, EthTrcv_LinkStateType*)
UL_LinkStateChange(CtrlIdx, LinkState)

Figure 11: Link State Change

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.

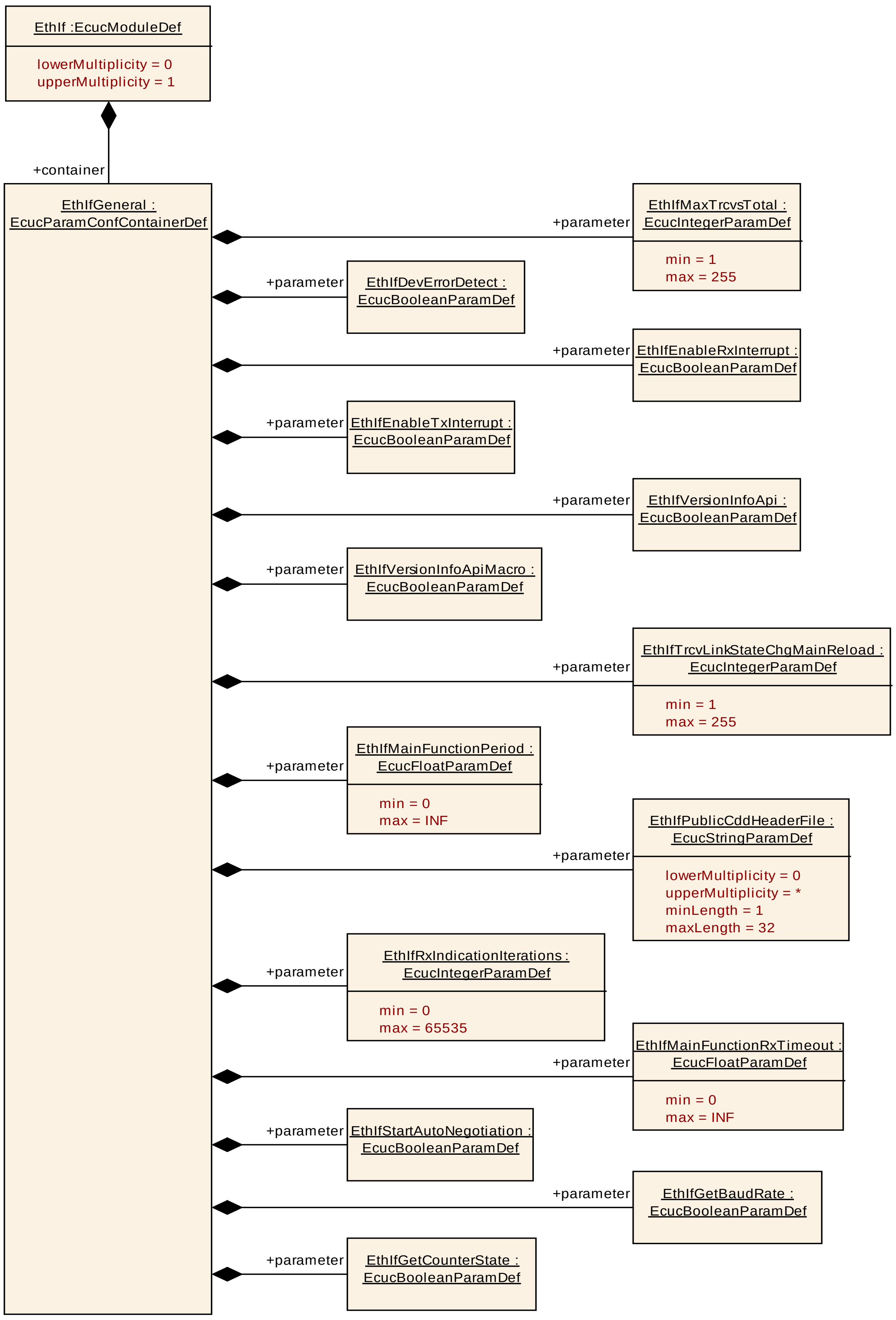

EthIf :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
EthIfGeneral : EcucParamConfContainerDef
+parameter
EthIfMaxTrcvsTotal : EcucIntegerParamDef
min = 1
max = 255
+parameter
EthIfDevErrorDetect : EcucBooleanParamDef
+parameter
EthIfEnableRxInterrupt : EcucBooleanParamDef
+parameter
EthIfEnableTxInterrupt : EcucBooleanParamDef
+parameter
EthIfVersionInfoApi : EcucBooleanParamDef
+parameter
EthIfVersionInfoApiMacro : EcucBooleanParamDef
+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
+parameter
EthIfMainFunctionRxTimeout : EcucFloatParamDef
min = 0
max = INF
+parameter
EthIfStartAutoNegotiation : EcucBooleanParamDef
+parameter
EthIfGetBaudRate : EcucBooleanParamDef
+parameter
EthIfGetCounterState : EcucBooleanParamDef

Figure 10.1: Ethernet Interface general configuration structure

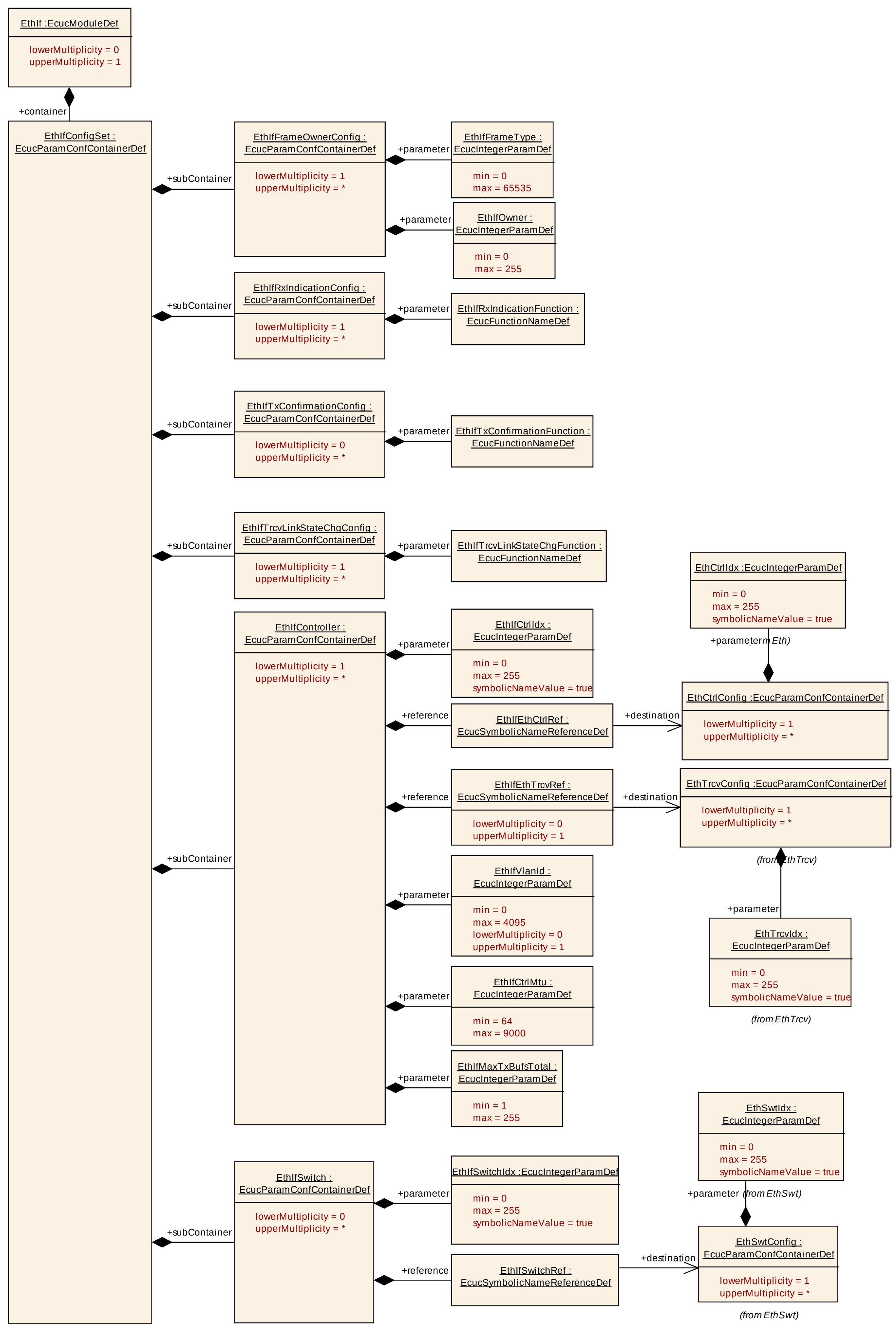


Figure 10.2: Ethernet Interface Interface configuration structure

10.1.1 Variants

VARIANT-POST-BUILD: All configuration parameters in container 'EthGeneral' shall be configurable at pre-compile time.
Use case: Object code delivery, selectable configuration

VARIANT-LINK-TIME: All configuration parameters in container 'EthGeneral' shall be configurable at pre-compile time.
Use case: Object code delivery, single configuration

VARIANT-PRE-COMPILE: All configuration parameters shall be configurable at pre-compile time.
Use case: Execution time optimizations, fix configuration

10.1.2 EthIf

Module Name	*EthIf*
Module Description	Configuration of the EthIf (Ethernet Interface) module.
Post-Build Variant Support	true

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.

10.1.3 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 Default Error Tracer (Det) detection and notification ON or OFF. • true: enabled (ON). • false: disabled (OFF).
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_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_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_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_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_00031 : (Obsolete)**	
Name	EthIfMainFunctionRxTimeout	
Description	This parameter is deprecated and will be removed in future. Old description: Timeout in seconds after which the EthIf stops to receive frames in an EthIfMainFunctionRx period. **Tags:** atp.Status=obsolete atp.StatusRevisionBegin=4.2.2	
Multiplicity	0..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_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_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_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_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_00007 :**		
Name	EthIfVersionInfoApi		
Description	Enables / Disables version info 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_00008 :**		
Name	EthIfVersionInfoApiMacro		
Description	Enables / Disables version info API macro implementation.		
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_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

10.1.4 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
EthIfRxIndicationConfig	1..*	Configuration of receive callback functions.
EthIfSwitch	0..*	This container contains the configuration of EthIfSwitches.
EthIfTrcvLinkStateChgConfig	1..*	Specifies link state change callback function
EthIfTxConfirmationConfig	0..*	Configuration of transmit indication callback functions.

10.1.5 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 .. 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_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 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	EthIfEthCtrlRef		
Description	Reference to a Controller, which is handled by a specific Driver. This reference is unique for the ECU.		
Multiplicity	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_00028 :		
Name	EthIfEthTrcvRef		
Description	Reference to a Ethernet Transceiver.		
Multiplicity	0..1		
Type	Symbolic name reference to [EthTrcvConfig]		
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		

No Included Containers

10.1.6 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

10.1.7 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

10.1.8 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

10.1.9 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

10.1.10 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).