Document Title	Specification of Ethernet Interface
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	417
Document Classification	Standard

Document Version	1.2.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Change Description
04.10.2011	1.2.0	AUTOSAR Administration	• Description of payload data in EthIf_Cbk_RxIndication adapted
24.10.2010	1.1.0	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
30.11.2009	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

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

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

Table of Contents

Known Limitations 5

1 Introduction and functional overview 6

2 Acronyms and abbreviations 8

3 Related documentation 9

3.1 Input documents 9
3.2 Related standards and norms 10

4 Constraints and assumptions 11

4.1 Limitations 11
4.2 Applicability to car domains 11

5 Dependencies to other modules 12

5.1 File structure 12
5.1.1 Code file structure 12
5.1.2 Header file structure 13

6 Requirements traceability 14

7 Functional specification 18

7.1 Ethernet BSW stack 18
7.1.1 Indexing scheme 18
7.1.2 Ethernet Interface main function 19
7.1.3 Requirements 19
7.1.4 Configuration description 20
7.1.5 Commercial Off The Shelf stack usage 21
7.2 Error classification 22
7.3 Error detection 23
7.4 Error notification 23
7.5 Debugging 23
7.6 Version checking 24

8 API specification 25

8.1 Imported types 25
8.2 Type definitions 25
8.2.1 EthIf_ConfigType 25
8.2.2 EthIf_StateType 25
8.3 Function definitions 26
8.3.1 EthIf_Init 26
8.3.2 EthIf_ControllerInit 27
8.3.3 EthIf_SetControllerMode 28
8.3.4 EthIf_GetControllerMode 28
8.3.5 EthIf_TransceiverInit 29
8.3.6 EthIf_SetTransceiverMode 30
8.3.7 EthIf_GetTransceiverMode 31
8.3.8 EthIf_GetPhysAddr 32

8.3.9 EthIf_ProvideTxBuffer 33
8.3.10 EthIf_Transmit 34
8.3.11 EthIf_GetVersionInfo 36
8.4 Callback notifications 37
8.4.1 EthIf_Cbk_RxIndication 37
8.4.2 EthIf_Cbk_TxConfirmation 38
8.5 Scheduled functions 39
8.5.1 EthIf_MainFunctionRx 39
8.5.2 EthIf_MainFunctionTx 39
8.6 Expected Interfaces 40
8.6.1 Mandatory Interfaces 40
8.6.2 Optional Interfaces 41
8.6.3 Configurable interfaces 41
9 Sequence diagrams 44
9.1 Initialization 44
9.2 Communication Initialization 45
9.3 Data Transmission 46
9.4 Data Reception 48
9.5 Link State Change 49
10 Configuration specification 50
10.1 How to read this chapter 50
10.1.1 Configuration and configuration parameters 50
10.1.2 Variants 50
10.1.3 Containers 50
10.1.4 Specification template for configuration parameters 51
10.2 Containers and configuration parameters 51
10.2.1 Variants 54
10.2.2 EthIf 54
10.2.3 EthIfGeneral 55
10.2.4 EthIfConfigSet 57
10.2.5 EthIfConnection2EthCtrlMapping 58
10.2.6 EthIfFrameOwnerConfig 58
10.2.7 EthIfRxIndicationConfig 59
10.2.8 EthIfTrcvLinkStateChgConfig 59
10.2.9 EthIfTxConfirmationConfig 60
10.3 Published Information 60
11 Not applicable requirements 61

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 (Internet Protocol, Address Resolution Protocol) 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.

[ETHIF111] ⌈
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).⌋()

[ETHIF123] ⌈
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).⌋()

[ETHIF112] ⌈
Therefore, the Ethernet Interface executable code (however, not the configuration used during runtime) shall be completely independent of the Ethernet Communication Controller(s).⌋()

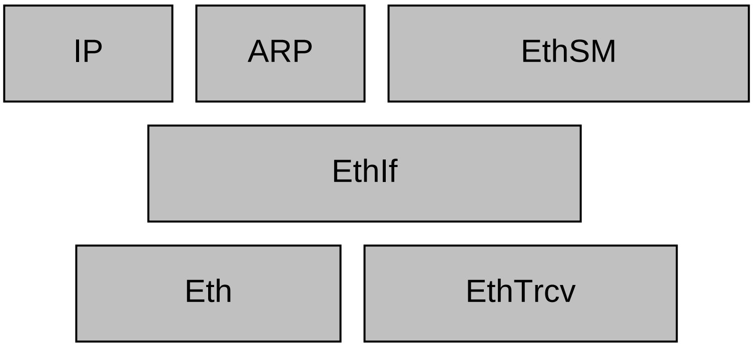

Figure 1.1: Ethernet stack module overview

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:***
ARP	Address Resolution Protocol
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

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] Specification of Communication
AUTOSAR_SWS_COM.pdf

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

[6] Specification of Ethernet Driver
AUTOSAR_SWS_EthernetDriver.pdf

[7] Specification of Ethernet State Manager
AUTOSAR_SWS_EthernetStateManager.pdf

[8] Specification of Ethernet Transceiver Driver
AUTOSAR_SWS_EthernetTransceiver.pdf

[9] Specification of Socket Adapter
AUTOSAR_SWS_SocketAdapter.pdf

[10] Specification of UDP Network Management
AUTOSAR_SWS_UDPNetworkManagement.pdf

[11] Specification of PDU Router
AUTOSAR_SWS_PDURouter.pdf

[12] BSW Scheduler Specification
AUTOSAR_SWS_Scheduler.pdf

[13] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[14] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

[15] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[16] Specification of Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[17] Specification of Diagnostics Event Manager
AUTOSAR_SWS_DiagnosticEventManager

[18] Specification of C Implementation Rules
AUTOSAR_TR_CImplementationRules.pdf

[19] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

3.2 Related standards and norms

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

[21] IEEE 802.3-2006

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. Currently, the Ethernet Interface module is limited to one Ethernet Driver and one Ethernet Transceiver Driver. To support multiple lower layer drivers the configuration would have to be extended.

The implementation is limited to 10MBit and 100MBit Ethernet and transceivers connected via Media Independent Interface (MII).

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:

- Development Error Tracer (DET) for reporting of development errors.
- Diagnostic Event Manager (DEM) for reporting of diagnostic-relevant events and states.
- BSW Scheduler mechanisms for data consistency and main function handling.

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.

5.1 File structure

5.1.1 Code file structure

[ETHIF001] ⌈

This specification shall not completely define the code file structure. The code-file structure shall include the following files named:

- EthIf_Lcfg.c – for link time configurable parameters and
- EthIf_PBcfg.c – for post build time configurable parameters.

These files shall contain all link time and post-build time configurable parameters.⌋()

5.1.2 Header file structure

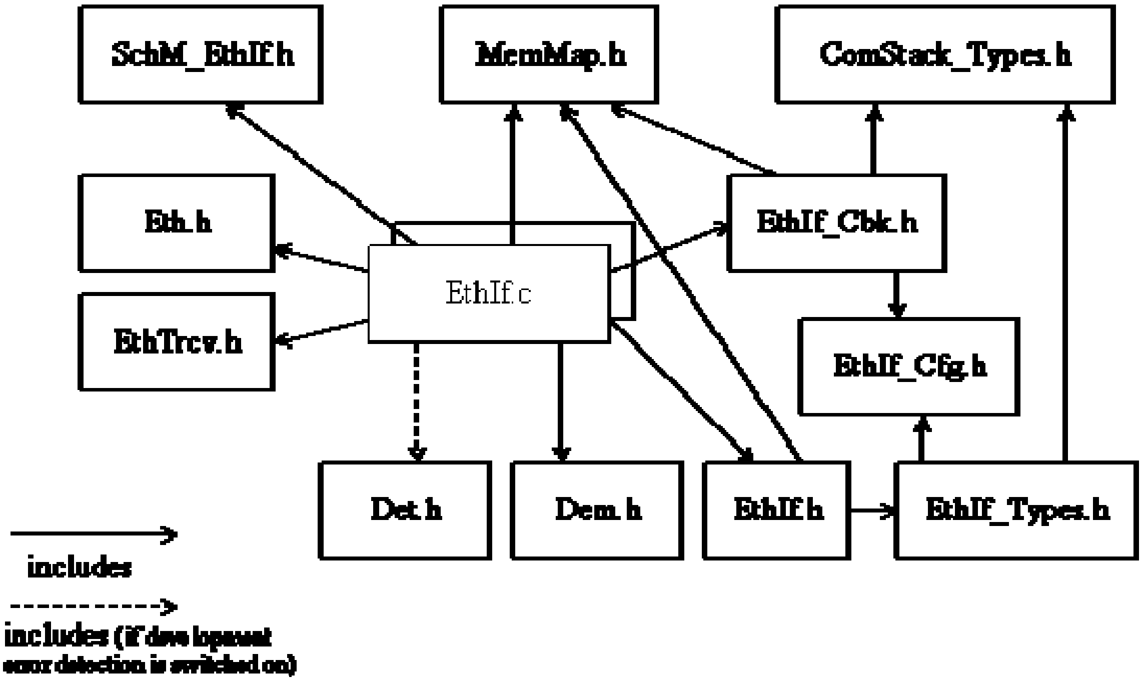

Figure 5.1: Ethernet Interface file structure

[ETHIF002] ⌈

The module shall include the Dem.h file. File Dem.h defines the APIs to report errors as well as the required Event Id symbols. This specification defines the name of the Event Id symbols provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols.⌋()

6 Requirements traceability

Requirement	Satisfied by
-	ETHIF081
-	ETHIF121
-	ETHIF092
-	ETHIF011
-	ETHIF051
-	ETHIF066
-	ETHIF048
-	ETHIF024
-	ETHIF087
-	ETHIF029
-	ETHIF127
-	ETHIF033
-	ETHIF076
-	ETHIF071
-	ETHIF005
-	ETHIF063
-	ETHIF097
-	ETHIF113
-	ETHIF091
-	ETHIF014
-	ETHIF012
-	ETHIF059
-	ETHIF053
-	ETHIF119
-	ETHIF101
-	ETHIF123
-	ETHIF052
-	ETHIF028
-	ETHIF105
-	ETHIF040
-	ETHIF099
-	ETHIF100
-	ETHIF008
-	ETHIF069
-	ETHIF109
-	ETHIF083
-	ETHIF025
-	ETHIF010

-	ETHIF022
-	ETHIF110
-	ETHIF035
-	ETHIF107
-	ETHIF078
-	ETHIF044
-	ETHIF021
-	ETHIF046
-	ETHIF103
-	ETHIF013
-	ETHIF093
-	ETHIF074
-	ETHIF049
-	ETHIF042
-	ETHIF002
-	ETHIF057
-	ETHIF067
-	ETHIF124
-	ETHIF070
-	ETHIF041
-	ETHIF045
-	ETHIF036
-	ETHIF060
-	ETHIF020
-	ETHIF017
-	ETHIF094
-	ETHIF084
-	ETHIF004
-	ETHIF085
-	ETHIF116
-	ETHIF102
-	ETHIF106
-	ETHIF082
-	ETHIF006
-	ETHIF126
-	ETHIF090
-	ETHIF055
-	ETHIF050
-	ETHIF062
-	ETHIF016
-	ETHIF015
-	ETHIF034

-	ETHIF089
-	ETHIF065
-	ETHIF019
-	ETHIF009
-	ETHIF114
-	ETHIF003
-	ETHIF061
-	ETHIF058
-	ETHIF001
-	ETHIF038
-	ETHIF043
-	ETHIF039
-	ETHIF098
-	ETHIF073
-	ETHIF037
-	ETHIF064
-	ETHIF068
-	ETHIF122
-	ETHIF079
-	ETHIF117
-	ETHIF032
-	ETHIF108
-	ETHIF095
-	ETHIF023
-	ETHIF056
-	ETHIF018
-	ETHIF080
-	ETHIF088
-	ETHIF030
-	ETHIF047
-	ETHIF026
-	ETHIF075
-	ETHIF054
-	ETHIF027
-	ETHIF118
-	ETHIF104
-	ETHIF096
-	ETHIF077
-	ETHIF112
-	ETHIF007
-	ETHIF120
-	ETHIF072

-	ETHIF086
-	ETHIF111
-	ETHIF031
BSW00170	ETHIF999

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 7.1 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.

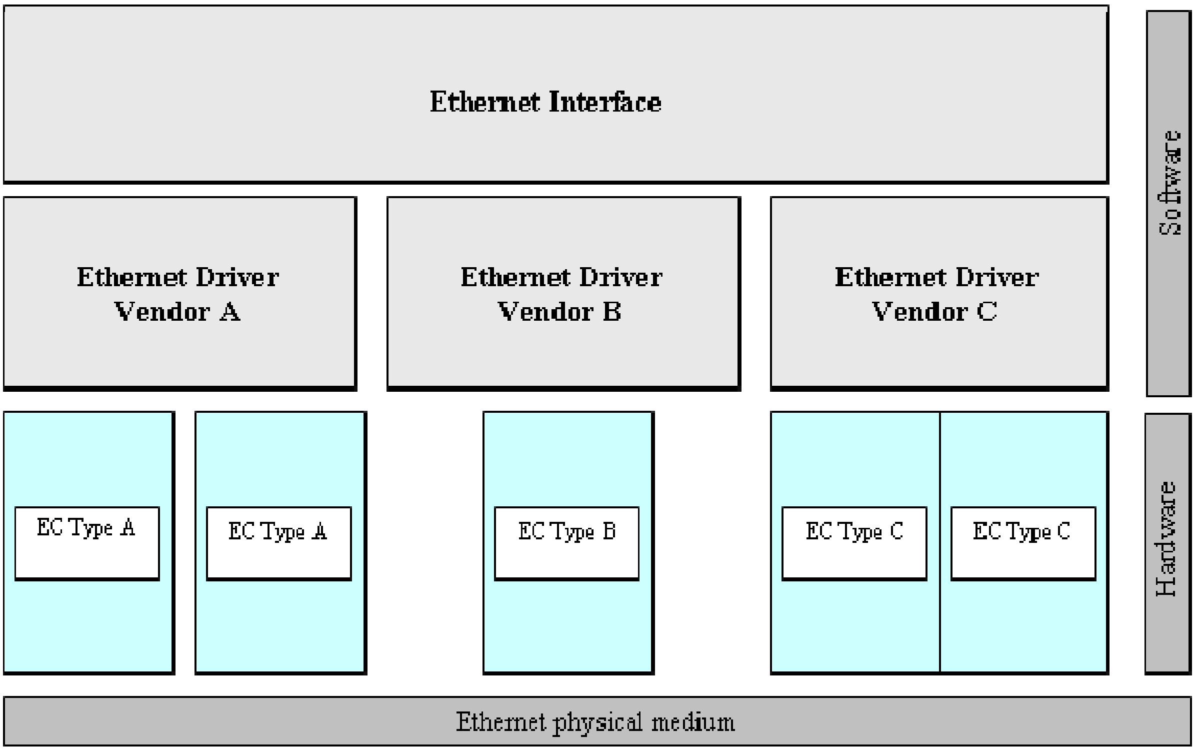

Figure 7.1: Basic Structure of the Ethernet BSW stack

7.1.1 Indexing scheme

Users of the Ethernet Interface identify Ethernet controller resources using an indexing scheme as depicted in Figure 7.2.

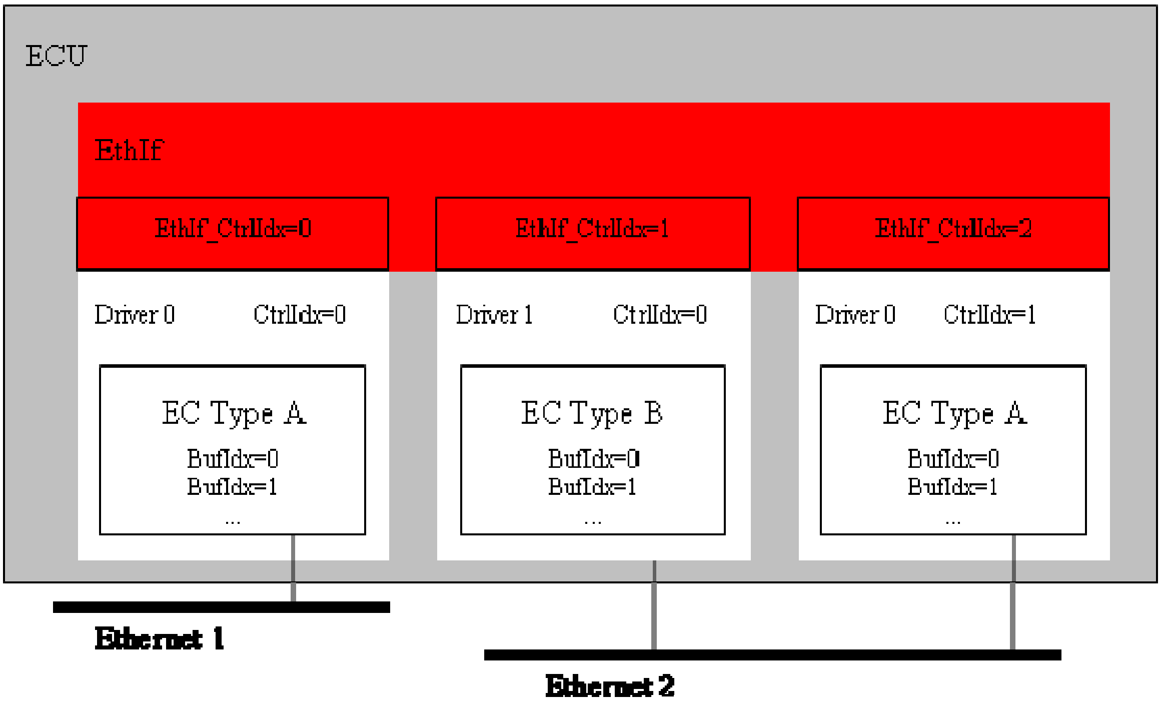

Figure 7.2: Ethernet Interface indexing scheme

[ETHIF003] ⌈
The Ethernet Interface is using a virtual zero-based controller index (EthIf_CtrlIdx) to abstract the access for upper software layers. It counts over all drivers with all local controller instances (Driver + CtrlIdx) which may be connected to several networks (Ethernet) providing buffers for data transmission (BufIdx).⌋()

7.1.2 Ethernet Interface main function

[ETHIF004] ⌈
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.⌋()

7.1.3 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.

[ETHIF005] ⌈

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

[ETHIF006] ⌈

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

[ETHIF007] ⌈
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.⌋()

[ETHIF008] ⌈
In case development error detection is enabled for the Ethernet Interface module: The Ethernet Interface module shall check API parameters for validity and report detected errors to the DET. ⌋()

DET API functions are specified in [16].

[ETHIF009] ⌈
The Ethernet Interface module implementation shall conform to the HIS subset of the MISRA C Standard (see document [18]).⌋()

[ETHIF010] ⌈
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.⌋()

[ETHIF011] ⌈

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

7.1.4 Configuration description

[ETHIF012] ⌈
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.⌋()

[ETHIF117] ⌈

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.⌋()

[ETHIF118] ⌈

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

[ETHIF013] ⌈
The configuration of the Ethernet Interface module shall be configured at ECU configuration time. None of the communication parameters shall be configured at runtime. ⌋()

[ETHIF014] ⌈
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 [6]) shall be evaluated for Ethernet Interface module configuration.

7.1.5 Commercial Off The Shelf stack usage

[ETHIF015] ⌈

A commercial off the shelf stack (COTS) shall be useable. ⌋()

The commercial stack is useable without adaption (Variant 1 in Figure 7.3). However, the Ethernet State Manager is not able to control the Ethernet controller and Ethernet transceiver in this case. The commercial stack may be adapted for usage with the Ethernet Interface. In this case, the Ethernet State Manager is able to control both Ethernet controller and Ethernet transceiver (Variant 2 in Figure 7.3).

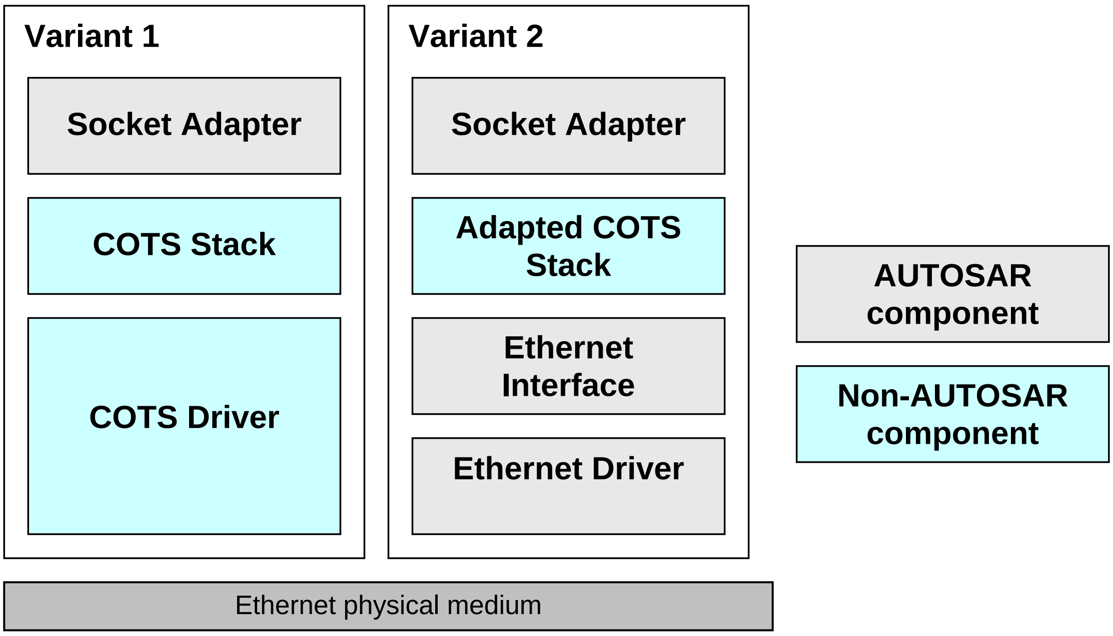

Figure 7.3: BSW stack architecture variants

7.2 Error classification

[ETHIF016] ⌈

The configuration of the Dem assigns values for production code Event Ids. The file Dem.h includes the file Dem_IntErrId.h. The file Dem_IntErrId.h publishes the values. ⌋()

[ETHIF017] ⌈

Development error values are of type uint8.⌋()

Type or error	***Relevance***	***Related error code***	***Value [hex]***
Invalid controller index	Development	ETHIF_E_INV_CTRL_IDX	0x01
Invalid transceiver index	Development	ETHIF_E_INV_TRCV_IDX	0x02
EthIf module was not initialized	Development	ETHIF_E_NOT_INITIALIZED	0x03
Invalid pointer in parameter list	Development	ETHIF_E_INV_POINTER	0x04
Invalid parameter	Development	ETHIF_E_INV_PARAM	0x05
None	Production		Assigned by DEM

7.3 Error detection

[ETHIF018] ⌈
The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch *EthIfDevErrorDetect* (see chapter 10) shall activate or deactivate the detection of all development errors.⌋()

[ETHIF019] ⌈
The *EthIfDevErrorDetect* switch enables API parameter checking. Chapter 7.2 and 8 contain the detailed description of the detected errors.⌋()

[ETHIF020] ⌈

Switching off the detection of production code errors shall not be possible.⌋()

7.4 Error notification

[ETHIF021] ⌈
The module shall report development errors to the *Det_ReportError* service of the Development Error Tracer (DET) if the pre-processor switch *EthIfDevErrorDetect* is set (see chapter 10).⌋()

[ETHIF022] ⌈

The module shall report production errors to the Diagnostic Event Manager.⌋()

7.5 Debugging

[ETHIF119] ⌈
Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable.⌋()

[ETHIF120] ⌈
All type definitions of variables, which shall be debugged, shall be accessible by the header file EthIf.h.⌋()

[ETHIF121] ⌈
The declaration of variables in the header file shall be such, that it is possible to calculate the size of the variables by C-“sizeof”.⌋()

[ETHIF122] ⌈

Variables available for debugging shall be described in the respective Basic Software Module Description.⌋()

7.6 Version checking

[ETHIF126]⌈

The Ethernet Interface module shall perform inter-module checks to avoid integration of incompatible files.

The imported include files shall be checked by preprocessing directives.⌋()

The Ethernet Interface module shall verify the following version numbers:
- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_RELEASE_MINOR_VERSION

Where <MODULENAME> is the module abbreviation of the other (external) modules providing header files included by the Ethernet Interface module.

If the values are not identical to the expected values, the Ethernet Interface module shall report an error.

8 API specification

8.1 Imported types

This chapter lists all types included from the following files:

[ETHIF023] ⌈

Module	***Imported Type***
ComStack_Types	BufReq_ReturnType
Dem	Dem_EventIdType
	Dem_EventStatusType
Eth	Eth_DataType
	Eth_FrameType
	Eth_ModeType
EthCtrl	EthCtrl_FrameType
	EthCtrl_ModeType
	TcpIp_DataType
	EthCtrl_ConfigType
EthTrcv	EthTrcv_BaudRateType
	EthTrcv_DuplexModeType
	EthTrcv_LinkStateType
	EthTrcv_ModeType
	EthTrcv_ConfigType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋()

8.2 Type definitions

8.2.1 EthIf_ConfigType

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

8.2.2 EthIf_StateType

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

[ETHIF024] ⌈

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	

⌋()

[ETHIF025] ⌈
The function shall store the access to the configuration structure for subsequent API calls.⌋()

[ETHIF114] ⌈
The function shall change the state of the component from ETHIF_STATE_UNINIT to ETHIF_STATE_INIT.⌋()

[ETHIF026] ⌈
If development error detection is enabled: the function shall check the parameter CfgPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER.⌋()

[ETHIF116] ⌈
If development error detection is enabled: the function shall check the parameter CfgPtr for containing a valid configuration. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM.⌋()

[ETHIF027] ⌈

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

[ETHIF028]⌈
Configuration: The user shall pass the post-build configuration or a NULL_PTR as parameter depending on the configuration variant.⌋()

8.3.2 EthIf_ControllerInit

[ETHIF029]⌈

Service name:	EthIf_ControllerInit	
Syntax:	`Std_ReturnType EthIf_ControllerInit( uint8 CtrlIdx, uint8 CfgIdx )`	
Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	CfgIdx	Index of the used configuration
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: controller could not be initialized
Description:	Initializes the indexed controller	

⌋()

[ETHIF030]⌈
The function EthIf_ControllerInit shall forward the call to function Eth_ControllerInit of the respective Ethernet Controller Driver.⌋()

[ETHIF031]⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED and return E_NOT_OK.⌋()

[ETHIF032]⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX and return E_NOT_OK.⌋()

[ETHIF033]⌈

Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.3 EthIf_SetControllerMode

[ETHIF034] ⌈

Service name:	EthIf_SetControllerMode	

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

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

⌋()

[ETHIF035] ⌈
The function EthIf_SetControllerMode shall forward the call to function Eth_SetControllerMode of the respective Ethernet Controller Driver.⌋()

[ETHIF036] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED and return E_NOT_OK.⌋()

[ETHIF037] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX and return E_NOT_OK.⌋()

[ETHIF038] ⌈

Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.4 EthIf_GetControllerMode

[ETHIF039] ⌈

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	ETHCTRL_MODE_DOWN: the controller is disabled ETHCTRL_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	

⌋()

[ETHIF040] ⌈

The function EthIf_GetControllerMode shall forward the call to function Eth_GetControllerMode of the respective Ethernet Controller Driver.⌋()

[ETHIF041] ⌈

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

[ETHIF042] ⌈

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

[ETHIF043] ⌈

If development error detection is enabled: the function shall check the parameter CtrlModePtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER and return E_NOT_OK.⌋()

[ETHIF044] ⌈

Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.5 EthIf_TransceiverInit

[ETHIF045] ⌈

Service name:	EthIf_TransceiverInit
Syntax:	`Std_ReturnType EthIf_TransceiverInit(`

	uint8 TrcvIdx, uint8 CfgIdx)	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TrcvIdx	Index of the Ethernet transceiver within the context of the Ethernet Interface
Parameters (inout):	None	
Parameters (out):	CfgIdx	Index of the used configuration
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: transceiver could not be initialized
Description:	Initializes the indexed transceiver	

⌋()

[ETHIF046] ⌈
The function EthIf_TransceiverInit shall forward the call to function EthTrcv_TransceiverInit of the respective Ethernet Transceiver Driver.⌋()

[ETHIF047] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED and return E_NOT_OK.⌋()

[ETHIF048] ⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX and return E_NOT_OK.⌋()

[ETHIF049] ⌈

Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.6 EthIf_SetTransceiverMode

[ETHIF050] ⌈

Service name:	EthIf_SetTransceiverMode	
Syntax:	Std_ReturnType EthIf_SetTransceiverMode(uint8 TrcIdx, EthTrcv_ModeType TrcvMode)	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TrcIdx	Index of the Ethernet transceiver within the context of the Ethernet Interface
	TrcvMode	ETHTRCV_MODE_DOWN: disable the transceiver

		ETHTRCV_MODE_ACTIVE: enable the transceiver
Parameters (inout):	None	
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	

⌋()

[ETHIF051] ⌈
The function EthIf_SetTransceiverMode shall forward the call to function EthTrcv_SetTransceiverMode of the respective Ethernet Transceiver Driver.⌋()

[ETHIF052] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED and return E_NOT_OK.⌋()

[ETHIF053] ⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX and return E_NOT_OK.⌋()

[ETHIF054] ⌈

Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.7 EthIf_GetTransceiverMode

[ETHIF055] ⌈

Service name:	EthIf_GetTransceiverMode	

Syntax:	`Std_ReturnType EthIf_GetTransceiverMode(`
	`uint8 TrcIdx,`
	`EthTrcv_ModeType* TrcvModePtr`
	`)`	

Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TrcIdx	Index of the Ethernet transceiver within the context of the Ethernet Interface
	TrcvModePtr	ETHTRCV_MODE_DOWN: the transceiver is disabled ETHTRCV_MODE_ACTIVE: the transceiver is enabled
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: transceiver mode could not be obtained
Description:	Obtain state of the indexed transceiver	

⌋()

[ETHIF056] ⌈
The function EthIf_GetTransceiverMode shall forward the call to function EthTrcv_GetTransceiverMode of the respective Ethernet Transceiver Driver.⌋()

[ETHIF057] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED and return E_NOT_OK.⌋()

[ETHIF058] ⌈
If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX and return E_NOT_OK.⌋()

[ETHIF059] ⌈
If development error detection is enabled: the function shall check the parameter TrcvModePtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER and return E_NOT_OK.⌋()

[ETHIF060] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.8 EthIf_GetPhysAddr

[ETHIF061] ⌈

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. Please refer to [16] for the physical source address specification.
Return value:	None	
Description:	Obtains the physical source address used by the indexed controller	

⌋()

[ETHIF062] ⌈
The function EthIf_GetPhysAddr shall forward the call to the respective Ethernet Controller Driver.⌋()

[ETHIF063] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[ETHIF064] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[ETHIF065] ⌈
If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER.⌋()

[ETHIF066] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.9 EthIf_ProvideTxBuffer

[ETHIF067] ⌈

Service name:	EthIf_ProvideTxBuffer	

Syntax:	`BufReq_ReturnType EthIf_ProvideTxBuffer(`
	`uint8 CtrlIdx,`
	`uint8* BufIdxPtr,`
	`Eth_DataType** 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
	BufPtr	Pointer to the granted buffer
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
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

⌋()

[ETHIF068] ⌈
The function EthIf_ProvideTxBuffer shall forward the call to the respective Ethernet Controller Driver.⌋()

[ETHIF069] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED and return BUFREQ_E_NOT_OK.⌋
()

[ETHIF070] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX and return BUFREQ_E_NOT_OK.⌋()

[ETHIF071] ⌈
If development error detection is enabled: the function shall check the parameter BufIdxPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER and return BUFREQ_E_NOT_OK.⌋()

[ETHIF072] ⌈
If development error detection is enabled: the function shall check the parameter BufPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER and return BUFREQ_E_NOT_OK.⌋()

[ETHIF073] ⌈
If development error detection is enabled: the function shall check the parameter LenBytePtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER and return BUFREQ_E_NOT_OK.⌋()

[ETHIF074] ⌈
Caveat: The function requires previous initialization (EthIf_Init).⌋()

8.3.10 EthIf_Transmit

[ETHIF075] ⌈

Service name:	EthIf_Transmit
Syntax:	`BufReq_ReturnType EthIf_Transmit(`

	`uint8 CtrlIdx,`
	`uint8 BufIdx,`
	`Eth_FrameType FrameType,`
	`boolean TxConfirmation,`
	`uint16 LenByte,`
	`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:	BufReq_ReturnType	E_OK: success E_NOT_OK: transmission failed
Description:	Triggers transmission of a previously filled transmit buffer	

⌋()

[ETHIF076] ⌈
The function EthIf_Transmit shall forward the call to the respective Ethernet Controller Driver.⌋()

[ETHIF077] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED and return E_NOT_OK.⌋()

[ETHIF078] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX and return E_NOT_OK.⌋()

[ETHIF079] ⌈
If development error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM and return E_NOT_OK.⌋()

[ETHIF080] ⌈
If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER and return E_NOT_OK.⌋()

[ETHIF081] ⌈

Caveat: The function requires previous buffer request (EthIf_ProvideTxBuffer).⌋()

8.3.11 EthIf_GetVersionInfo

[ETHIF082] ⌈

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	

⌋()

[ETHIF083] ⌈
The function EthIf_ GetVersionInfo shall return the version information
of this module. The version information includes:
- Two bytes for the vendor ID
- Two bytes for the module ID
- Three bytes version number. The numbering shall be vendor specific; it
consists of:
- The major, the minor and the patch version number of the module.
- The AUTOSAR specification version number shall not be included. The AUTOSAR
specification version number is checked during compile time and therefore not
required in this API.⌋()

[ETHIF084] ⌈
The function EthIf_ GetVersionInfo shall be pre compile time configurable On/Off by the configuration parameter: EthIfVersionInfoApi using the keyword ETHIF_GET_VERSION_INFO.⌋()

[ETHIF127] ⌈
If development error detection is enabled: the function shall check the parameter VersionInfoPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_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_Cbk_RxIndication

[ETHIF085] ⌈

Service name:	EthIf_Cbk_RxIndication	

Syntax:	`void EthIf_Cbk_RxIndication(`
	`uint8 CtrlIdx,`
	`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
	DataPtr	Ethernet frame comprising the following elements in the listed order: Target MAC, Source MAC, VLAN tag (optional), Type, Payload.
	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	

⌋()

[ETHIF086] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[ETHIF087] ⌈
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[ETHIF088] ⌈
If development error detection is enabled: the function shall check the parameter DataPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_POINTER.⌋()

[ETHIF089] ⌈

Caveat: The function requires previous initialization (EthIf_Init).⌋()

[ETHIF090] ⌈

Caveat: The function shall be callable on interrupt level.⌋()

8.4.2 EthIf_Cbk_TxConfirmation

[ETHIF091] ⌈

Service name:	EthIf_Cbk_TxConfirmation	

Syntax:	`void EthIf_Cbk_TxConfirmation(`
	`uint8 CtrlIdx,`
	`uint8 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	

⌋()

[ETHIF092] ⌈

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

[ETHIF093] ⌈

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

[ETHIF094] ⌈

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

[ETHIF095] ⌈

Caveat: The function requires previous initialization (EthIf_Init).⌋()

[ETHIF096] ⌈

Caveat: The function shall be callable on interrupt level.⌋()

8.5 Scheduled functions

The Basic Software Scheduler shall directly call these functions. The following functions shall have no return value and no parameter. The functions shall not be reentrant.
Terms and definitions:
Fixed cyclic: Fixed cyclic means that one cycle time is defined at configuration and shall not be changed because functionality is requiring that fixed timing (e.g. filters).
Variable cyclic: Variable cyclic means that the cycle times are defined at configuration, but might be mode dependent and therefore vary during runtime.
On pre condition: On pre condition means that no cycle time can be defined. The function will be called when conditions are fulfilled. Alternatively, the function may be called cyclically however the cycle time will be assigned dynamically during runtime by other modules.

8.5.1 EthIf_MainFunctionRx

[ETHIF097] ⌈

Service name:	EthIf_MainFunctionRx
Syntax:	`void EthIf_MainFunctionRx( void )`
Service ID[hex]:	0x20
Timing:	FIXED_CYCLIC
Description:	The function checks for new received frames and issues transmission confirmations in polling mode. It checks also for transceiver state changes.

⌋()

[ETHIF098] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[ETHIF099] ⌈
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

[ETHIF113] ⌈

Service name:	EthIf_MainFunctionTx
Syntax:	`void EthIf_MainFunctionTx( void )`
Service ID[hex]:	0x21
Timing:	FIXED_CYCLIC

Description:	The function issues transmission confirmations in polling mode. It checks also for transceiver state changes.

⌋()

[ETHIF124] ⌈
If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOT_INITIALIZED.⌋()

[ETHIF100] ⌈
The transmission confirmation check shall be pre compile time configurable On/Off by the configuration parameter: ETHIF_ENABLE_TX_INTERRUPT⌋()

[ETHIF101] ⌈
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.

[ETHIF102] ⌈

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.
EthCtrl_ControllerInit	Initializes the indexed controller
EthCtrl_GetControllerMode	Obtains the state of the indexed controller
EthCtrl_GetCounterState	Reads the value of a counter specified with its memory offset
EthCtrl_GetPhysAddr	Obtains the physical source address used by the indexed controller
EthCtrl_GetVersionInfo	Returns the version information of this module
EthCtrl_Init	Initializes the Ethernet Driver
EthCtrl_ProvideTxBuffer	Provides access to a transmit buffer of the specified controller
EthCtrl_ReadMii	Reads a transceiver register
EthCtrl_Receive	Triggers frame reception
EthCtrl_SetControllerMode	Enables / disables the indexed controller
EthCtrl_Transmit	Triggers transmission of a previously filled transmit buffer
EthCtrl_TxConfirmation	Triggers frame transmission confirmation
EthCtrl_WriteMii	Configures a transceiver register or triggers a function offered by the receiver
EthTrcv_GetBaudRate	Obtains the baud rate of the indexed transceiver
EthTrcv_GetDuplexMode	Obtains the duplex mode of the indexed transceiver

EthTrcv_GetLinkState	Obtains the link state of the indexed transceiver
EthTrcv_GetTransceiverMode	Obtains the state of the indexed transceiver
EthTrcv_GetVersionInfo	Returns the version information of this module
EthTrcv_Init	Initializes the Ethernet Transceiver Driver
EthTrcv_SetTransceiverMode	Enables / disables the indexed transceiver
EthTrcv_StartAutoNegotiation	Restarts the negotiation of the transmission parameters used by the indexed transceiver
EthTrcv_TransceiverInit	Initializes the indexed transceiver

⌋()

8.6.2 Optional Interfaces

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

[ETHIF103] ⌈

API function	***Description***
Det_ReportError	Service to report development errors.
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.

[ETHIF104] ⌈

Service name:	<User>_RxIndication	

Syntax:	`void <User>_RxIndication(`
	`uint8 CtrlIdx,`
	`Eth_DataType* BufPtr,`
	`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
	BufPtr	Pointer to buffer with received payload
	LenByte	Received payload length in bytes
Parameters (inout):	None	
Parameters (out):	None	
Return value:	void	--
Description:	Indicates the reception of an Ethernet frame	

⌋()

[ETHIF105] ⌈

The callback function shall be configurable by the configuration parameter: EthIfRxIndicationFunction⌋()

[ETHIF106]⌈

Service name:	<User>_TxConfirmation	

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

⌋()

[ETHIF107]⌈
The callback function shall be configurable by the configuration parameter: EthIfTxConfirmationFunction⌋()

[ETHIF108]⌈

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 if the transceiver is disabled ETHTRCV_LINK_STATE_ACTIVE if the transceiver is enabled
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indicates the change of a transceiver state	

⌋()

[ETHIF109]⌈
The callback function shall be configurable by the configuration parameter: EthIfTrcvLinkStateChgFunction⌋()

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

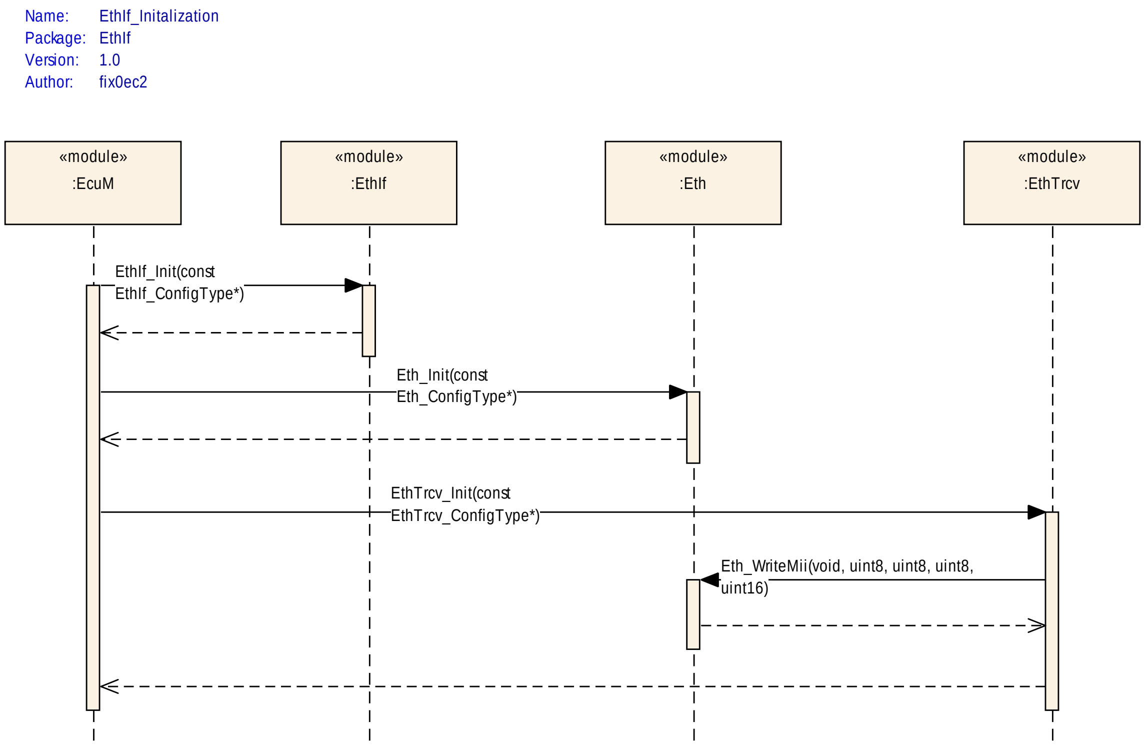
Name: EthIf_Initalization
Package: EthIf
Version: 1.0
Author: fix0ec2
«module» :EcuM
«module» :EthIf
«module» :Eth
«module» :EthTrcv
EthIf_Init(const EthIf_ConfigType*)
Eth_Init(const Eth_ConfigType*)
EthTrcv_Init(const EthTrcv_ConfigType*)
Eth_WriteMii(void, uint8, uint8, uint8, uint16)

Figure 9.1: Initialization

9.2 Communication Initialization

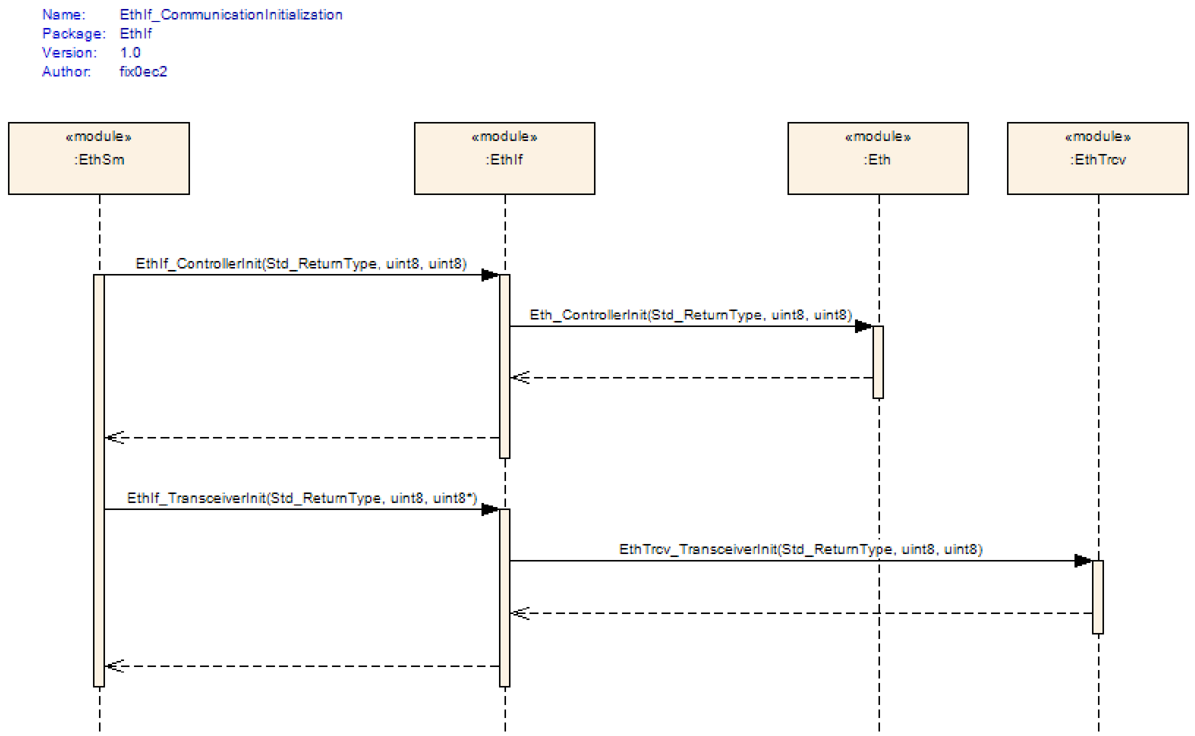
Name: EthIf_CommunicationInitialization
Package: EthIf
Version: 1.0
Author: fix0ec2
«module» :EthSm
«module» :EthIf
«module» :Eth
«module» :EthTrcv
EthIf_ControllerInit(Std_ReturnType, uint8, uint8)
Eth_ControllerInit(Std_ReturnType, uint8, uint8)
EthIf_TransceiverInit(Std_ReturnType, uint8, uint8*)
EthTrcv_TransceiverInit(Std_ReturnType, uint8, uint8)

Figure 9.2: Communication Initialization

9.3 Data Transmission

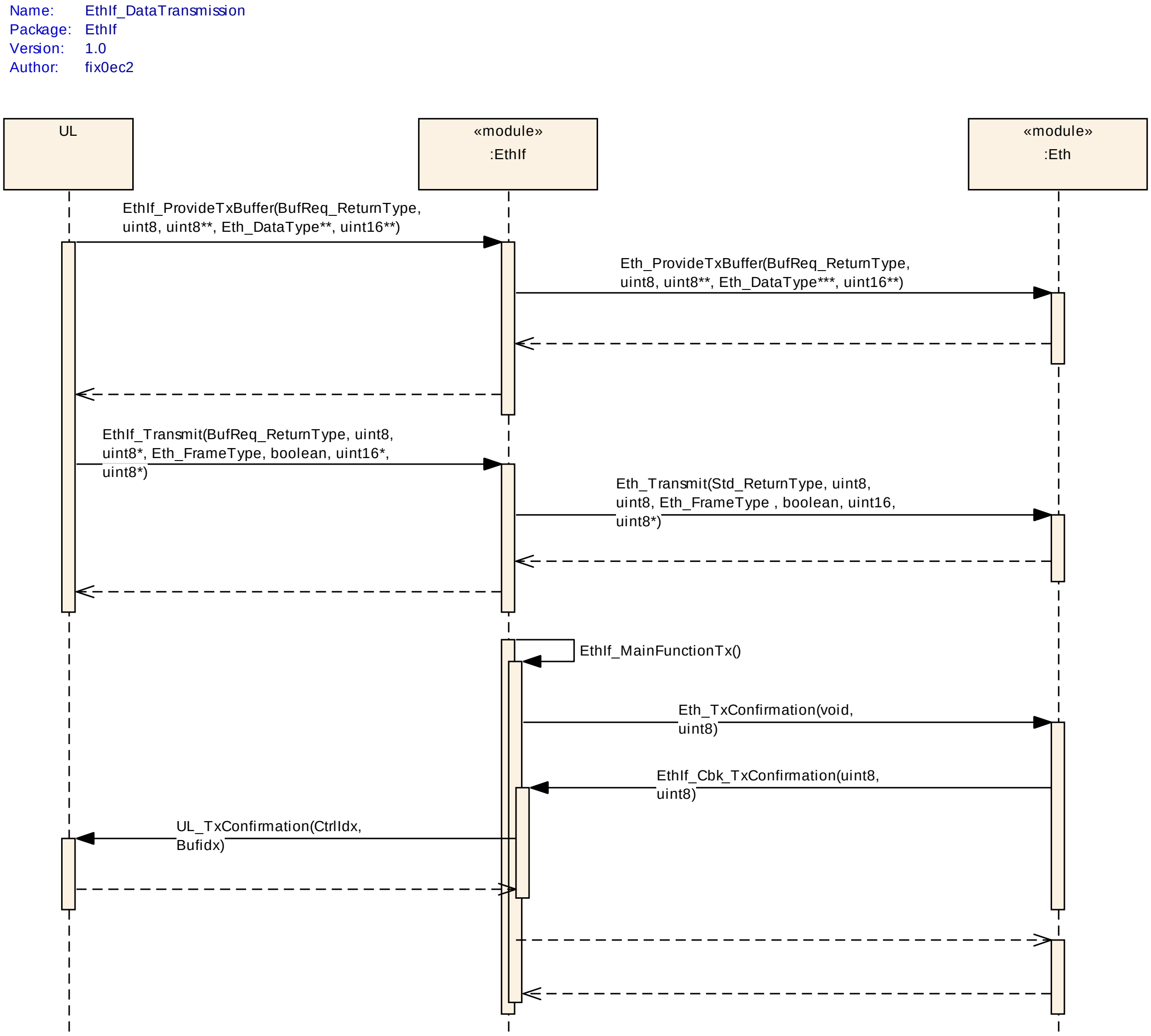
Name: EthIf_DataTransmission
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module» :EthIf
«module» :Eth
EthIf_ProvideTxBuffer(BufReq_ReturnType, uint8, uint8**, Eth_DataType**, uint16**)
Eth_ProvideTxBuffer(BufReq_ReturnType, uint8, uint8**, Eth_DataType***, uint16**)
EthIf_Transmit(BufReq_ReturnType, uint8, uint8*, Eth_FrameType, boolean, uint16*, uint8*)
Eth_Transmit(Std_ReturnType, uint8, uint8, Eth_FrameType , boolean, uint16, uint8*)
EthIf_MainFunctionTx()
Eth_TxConfirmation(void, uint8)
EthIf_Cbk_TxConfirmation(uint8, uint8)
UL_TxConfirmation(CtrlIdx, Bufidx)

Figure 9.3: Frame Transmission in Polling Mode

[ETHIF115] ⌈

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.⌋()

[ETHIF125] ⌈

EthIf_Cbk_TxConfirmation shall forward the confirmation to the registered call-back functions <User>_TxConfirmation.⌋()

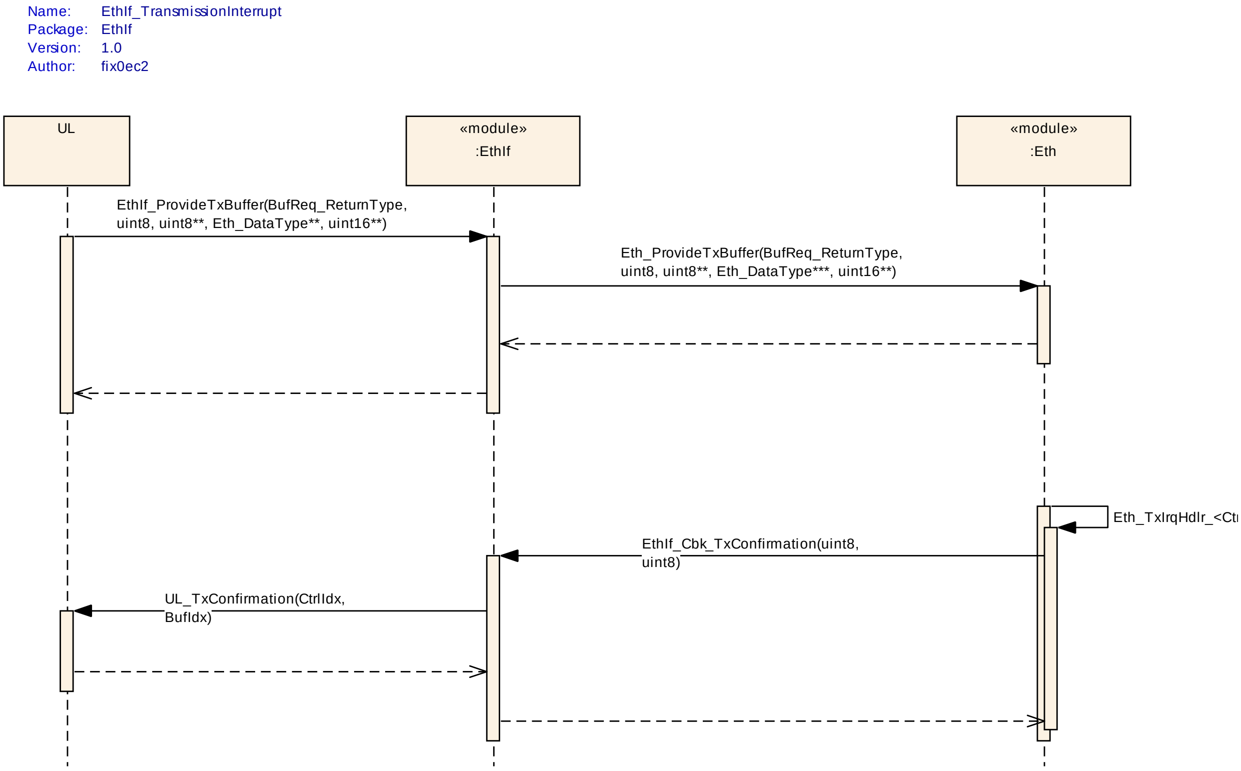
Name: EthIf_TransmissionInterrupt
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module»
:EthIf
«module»
:Eth
EthIf_ProvideTxBuffer(BufReq_ReturnType, uint8, uint8**, Eth_DataType**, uint16**)
Eth_ProvideTxBuffer(BufReq_ReturnType, uint8, uint8**, Eth_DataType***, uint16**)
Eth_TxIrqHdlr_<Ct
EthIf_Cbk_TxConfirmation(uint8,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

Figure 9.4: Frame Transmission in Interrupt Mode

9.4 Data Reception

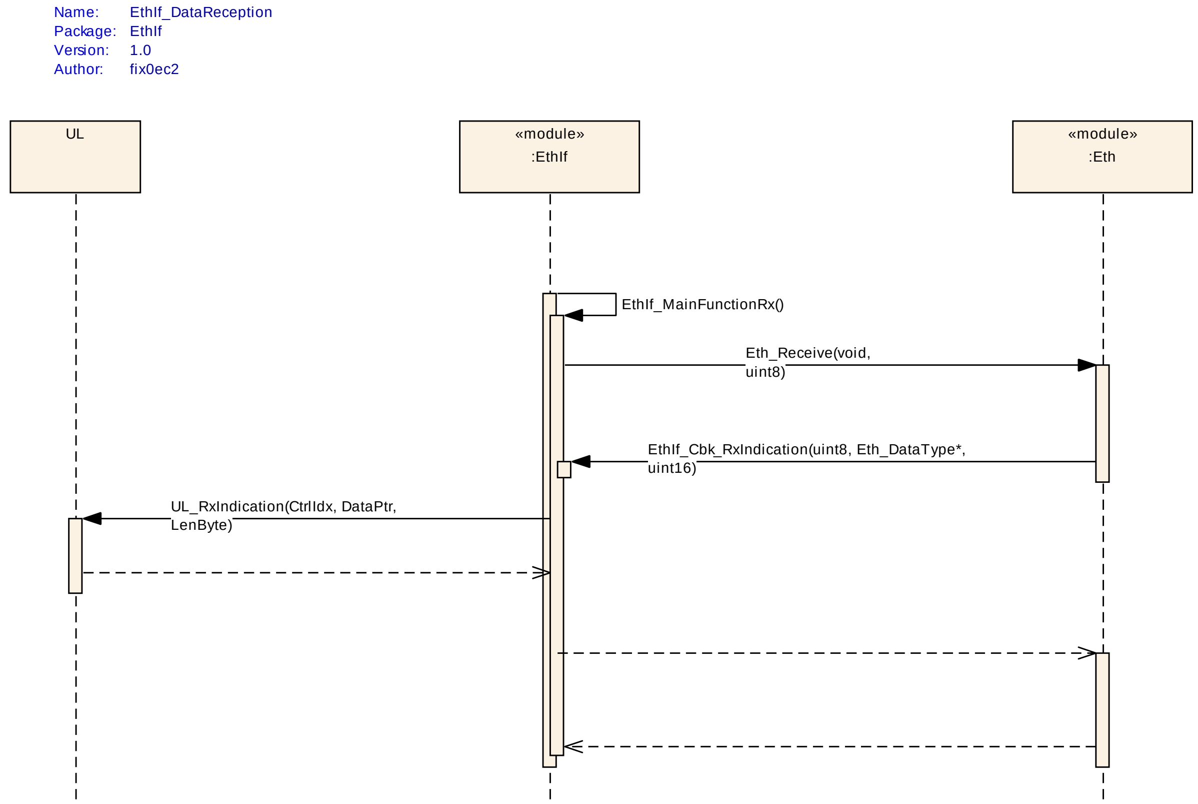

Figure 9.5: Frame Reception in Polling Mode

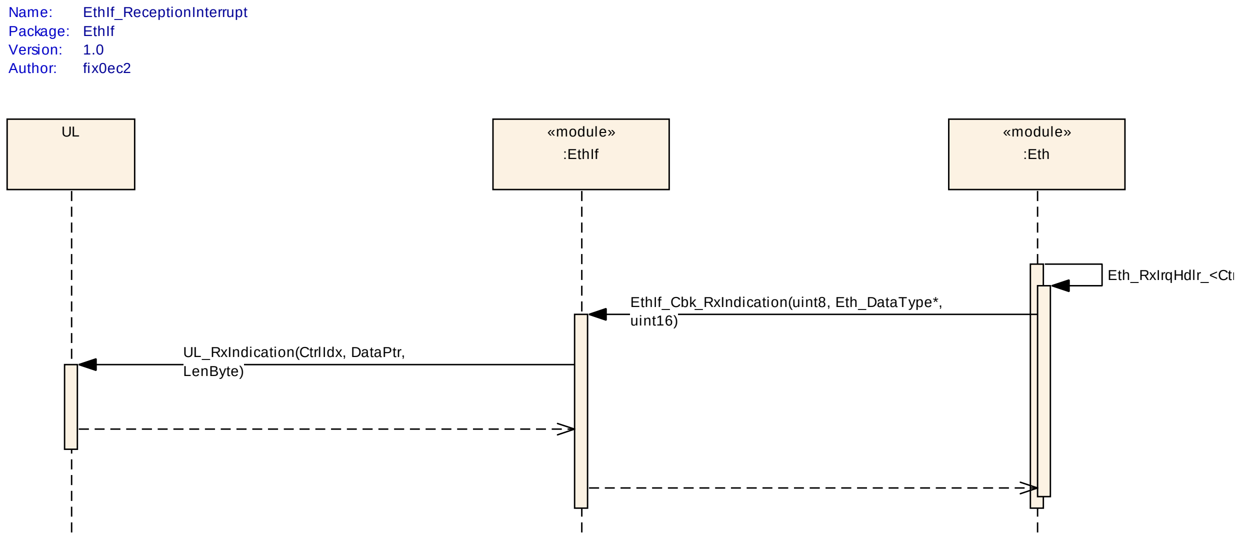
Name: EthIf_ReceptionInterrupt
Package: EthIf
Version: 1.0
Author: fix0ec2
UL
«module»
:EthIf
«module»
:Eth
Eth_RxIrqHdlr_<Ct
EthIf_Cbk_RxIndication(uint8, Eth_DataType*,
uint16)
UL_RxIndication(CtrlIdx, DataPtr,
LenByte)

Figure 9.6: Frame Reception in Interrupt Mode

9.5 Link State Change

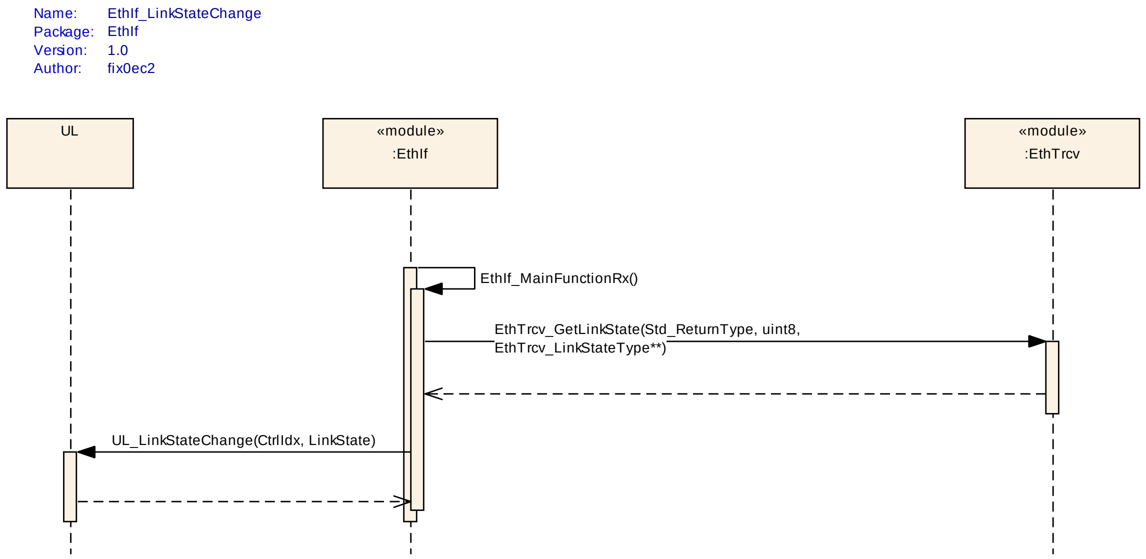

Figure 9.7: Link State Change

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 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [2].
- AUTOSAR ECU Configuration Specification [13]. This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Variants

Variants describe sets of configuration parameters. E.g., variant 1: only pre-compile time configuration parameters; variant 2: mix of pre-compile and post-build time configuration parameters. In one variant, a parameter can only be of one configuration class.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.

- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.1.4 Specification template for configuration parameters

The following tables consist of three sections:

- the general section
- the configuration parameter section
- the section of included/referenced containers

Pre-compile time - specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* - the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* - the configuration parameter shall be of configuration class *Post Build* and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 7.5.

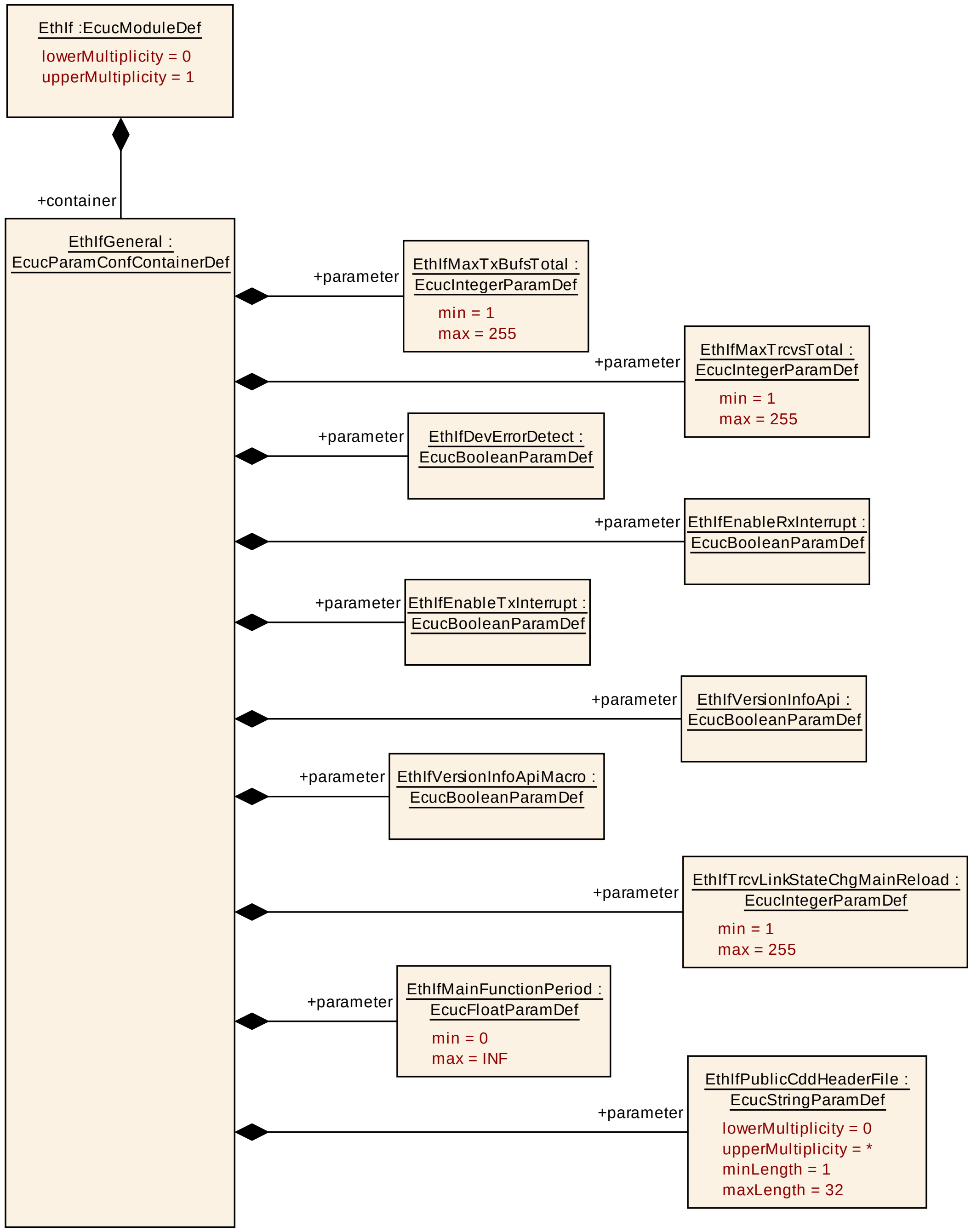

Figure 10.1: Ethernet Interface configuration structure

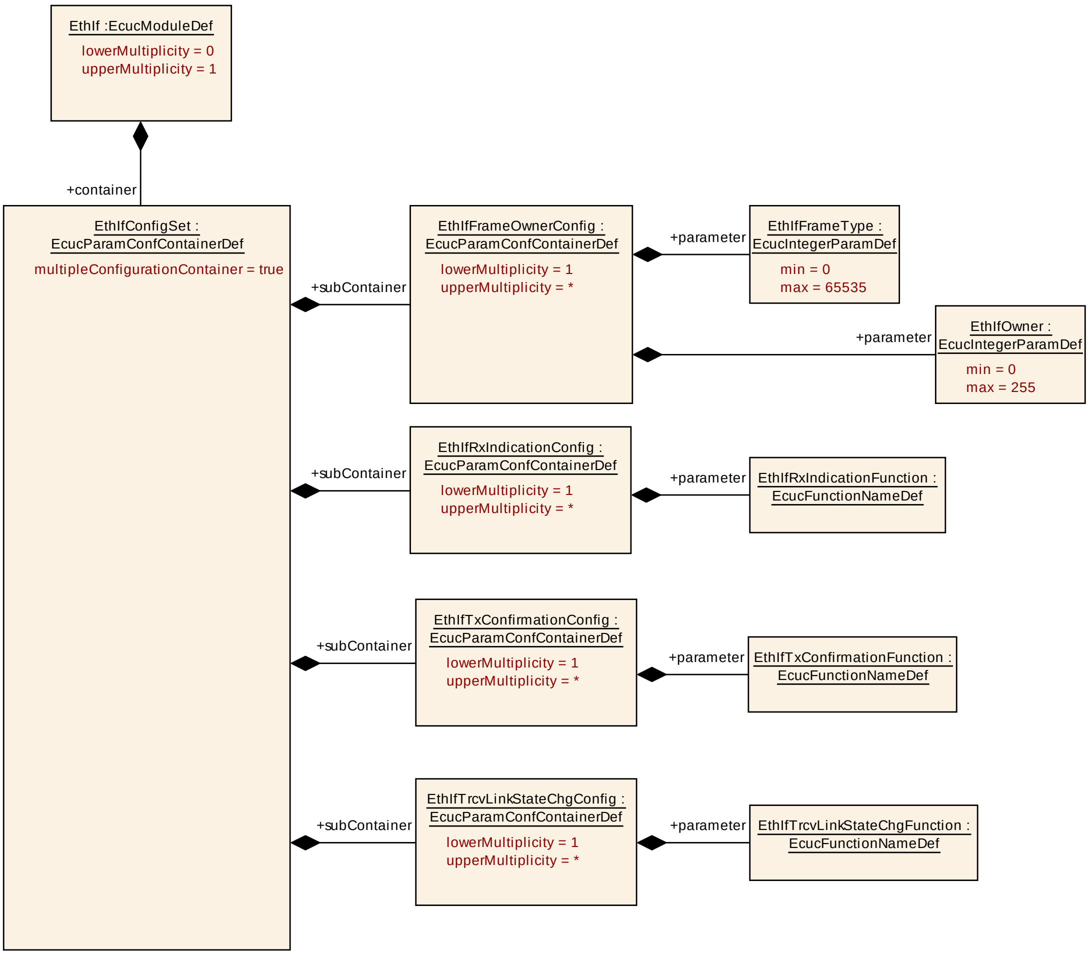
EthIf :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
EthIfConfigSet : EcucParamConfContainerDef
multipleConfigurationContainer = true
+subContainer
EthIfFrameOwnerConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
EthIfFrameType : EcucIntegerParamDef
min = 0
max = 65535
+parameter
EthIfOwner : EcucIntegerParamDef
min = 0
max = 255
+subContainer
EthIfRxIndicationConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
EthIfRxIndicationFunction : EcucFunctionNameDef
+subContainer
EthIfTxConfirmationConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
EthIfTxConfirmationFunction : EcucFunctionNameDef
+subContainer
EthIfTrcvLinkStateChgConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
EthIfTrcvLinkStateChgFunction : EcucFunctionNameDef

Figure 10.2: Ethernet Interface Configuration Set configuration structure

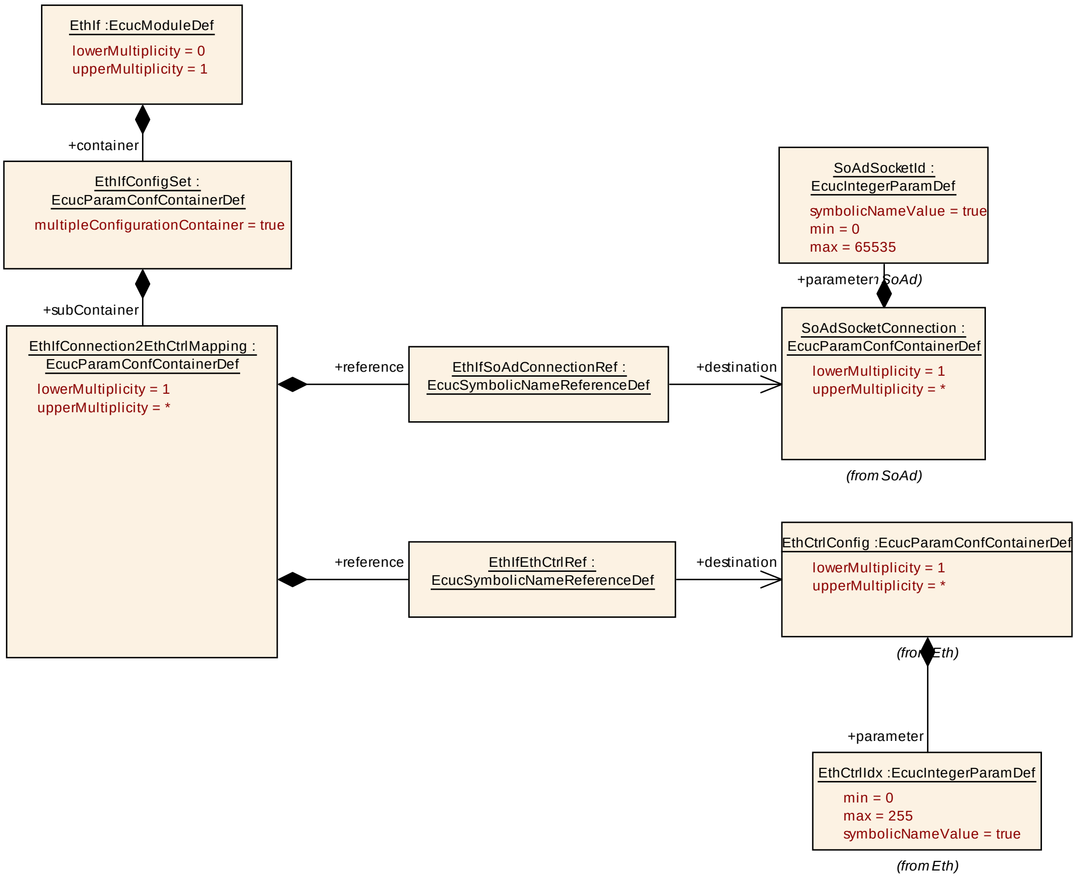

Figure 10.3: Ethernet Interface Connection Controller Mapping configuration structure

10.2.1 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

10.2.2 EthIf

Module Name	*EthIf*

Module Description	Configuration of the EthIf (Ethernet Interface) module.

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.

10.2.3 EthIfGeneral

SWS Item	**ETHIF001_Conf :**
Container Name	EthIfGeneral
Description	This container contains the general configuration parameters of the Ethernet Interface.
Configuration Parameters	

SWS Item	**ETHIF004_Conf :**		
Name	EthIfDevErrorDetect		
Description	Enables / Disables development error detection.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ETHIF005_Conf :**		
Name	EthIfEnableRxInterrupt		
Description	Enables / Disables receive interrupt.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ETHIF006_Conf :**		
Name	EthIfEnableTxInterrupt		
Description	Enables / Disables the transmit interrupt.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ETHIF023_Conf :**
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ETHIF003_Conf :**		
Name	EthIfMaxTrcvsTotal		
Description	Limits the total number of transceivers.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ETHIF002_Conf :**		
Name	EthIfMaxTxBufsTotal		
Description	Limits the total number of transmit buffers.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ETHIF024_Conf :**		
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ETHIF009_Conf :**
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ETHIF007_Conf :**		
Name	EthIfVersionInfoApi		
Description	Enables / Disables version info API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ETHIF008_Conf :**		
Name	EthIfVersionInfoApiMacro		
Description	Enables / Disables version info API macro implementation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.2.4 EthIfConfigSet

SWS Item	**ETHIF010_Conf :**
Container Name	EthIfConfigSet [Multi Config Container]
Description	Collecting container for all parameters with post-build configuration classes.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthIfConnection2EthCtrlMapping	1..*	Maps a particular connection in the Ethernet Interface to the physical Ethernet Controller.
EthIfFrameOwnerConfig	1..*	Configuration of Ethernet frame owner
EthIfRxIndicationConfig	1..*	Configuration of receive callback functions.
EthIfTrcvLinkStateChgConfig	1..*	Specifies link state change callback function
EthIfTxConfirmationConfig	1..*	Configuration of transmit indication callback functions.

10.2.5 EthIfConnection2EthCtrlMapping

SWS Item	**ETHIF020_Conf :**
Container Name	EthIfConnection2EthCtrlMapping
Description	Maps a particular connection in the Ethernet Interface to the physical Ethernet Controller.
Configuration Parameters	

SWS Item	**ETHIF022_Conf :**		
Name	EthIfEthCtrlRef		
Description	Reference to the controller in Ethernet Driver on which this connection will be transmitted / received. Connections are specified in the Socket Adapter [9].		
Multiplicity	1		
Type	Reference to [EthCtrlConfig]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**ETHIF021_Conf :**		
Name	EthIfSoAdConnectionRef		
Description	--		
Multiplicity	1		
Type	Reference to [SoAdSocketConnection]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.6 EthIfFrameOwnerConfig

SWS Item	**ETHIF011_Conf :**
Container Name	EthIfFrameOwnerConfig
Description	Configuration of Ethernet frame owner
Configuration Parameters	

SWS Item	**ETHIF012_Conf :**		
Name	EthIfFrameType		
Description	Selects the Ethernet frame type.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**ETHIF013_Conf :**

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	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.7 EthIfRxIndicationConfig

SWS Item	**ETHIF014_Conf :**
Container Name	EthIfRxIndicationConfig
Description	Configuration of receive callback functions.
Configuration Parameters	

SWS Item	**ETHIF015_Conf :**		
Name	EthIfRxIndicationFunction		
Description	Specifies receive indication callback function.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.8 EthIfTrcvLinkStateChgConfig

SWS Item	**ETHIF018_Conf :**
Container Name	EthIfTrcvLinkStateChgConfig
Description	Specifies link state change callback function
Configuration Parameters	

SWS Item	**ETHIF019_Conf :**
Name	EthIfTrcvLinkStateChgFunction
Description	Specifies link state change callback function
Multiplicity	1
Type	EcucFunctionNameDef

Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.9 EthIfTxConfirmationConfig

SWS Item	**ETHIF016_Conf :**
Container Name	EthIfTxConfirmationConfig
Description	Configuration of transmit indication callback functions.
Configuration Parameters	

SWS Item	**ETHIF017_Conf :**		
Name	EthIfTxConfirmationFunction		
Description	Specifies transmit indication callback function		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.3 Published Information

[ETHIF110] ⌈The standardized common published parameters as required by BSW00402 in the SRS General on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [6].⌋()

Additional module-specific published parameters are listed below if applicable.

11 Not applicable requirements

[ETHIF999]「 These requirements are not applicable to this specification. 」(BSW00170)