Document Title	Specification of Ethernet Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	430
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
30.09.2011	1.2.0	AUTOSAR Administration	• Description of buffer behaviour in Eth_SetControllerMode extended.
26.10.2010	1.1.0	AUTOSAR Administration	• Enhanced development error detection for active controller before controller access • Further post-build configurable parameters • Improved description of 'XxxCtrlIdx' semantics • 'Instance ID' removed from Version Info (concerns Eth_GetVersionInfo API) • Additional development error in Eth_GetVersionInfo API
30.11.2009	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

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

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

Table of Contents

Known Limitations

Currently, chapter 5 Dependencies to other modules does not describe the versions of dependent modules. Thus, a version check will extend the chapter.

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module Ethernet Driver.

In the AUTOSAR Layered Software Architecture, the Ethernet Driver belongs to the *Microcontroller Abstraction Layer*, or more precisely, to the *Communication Drivers*.

This indicates the main task of the Ethernet Driver:
Provide to the upper layer (Ethernet Interface) a hardware independent interface comprising multiple equal controllers. This interface shall be uniform for all controllers. Thus, the upper layer (Ethernet Interface) may access the underlying bus system in a uniform manner. The interface provides functionality for initialization, configuration and data transmission. The configuration of the Ethernet Driver however is bus specific, since it takes into account the specific features of the communication controller.

A single Ethernet Driver module supports only one type of controller hardware, but several controllers of the same type. The Ethernet Driver's prefix requires a unique namespace. The Ethernet Interface can access different controller types using different Ethernet Drivers using this prefix. The decision which driver to use to access a particular controller is a configuration parameter of the Ethernet Interface.

Figure 1.1 depicts the lower part of the Ethernet stack. One Ethernet Interface accesses several controllers using one or several Ethernet Drivers.

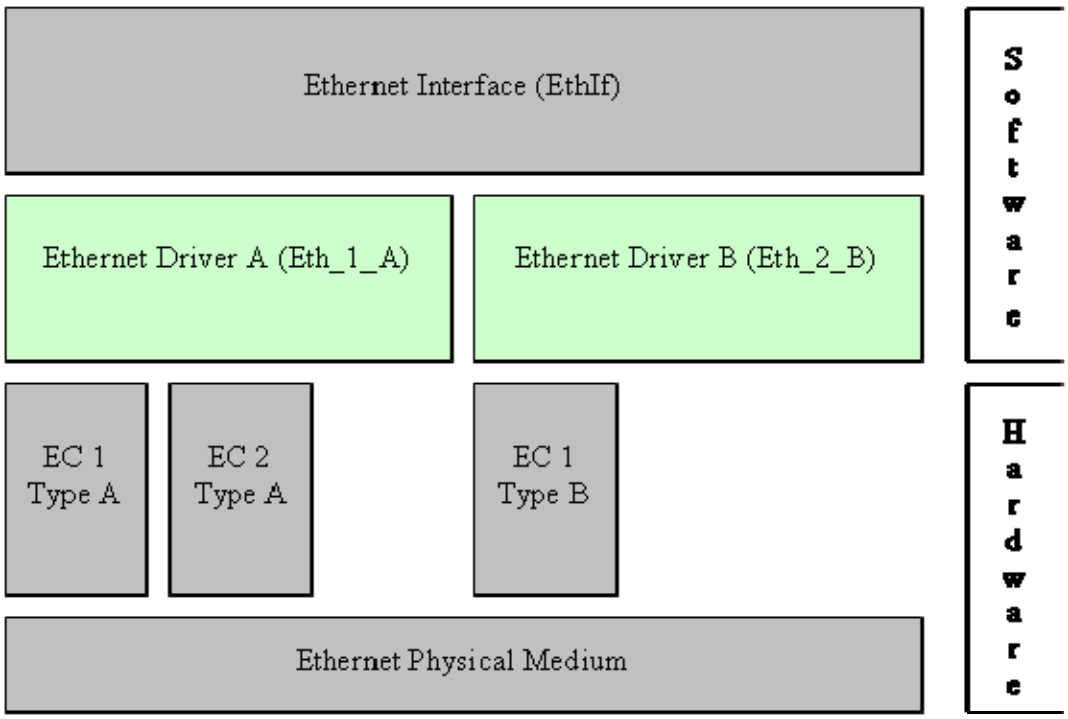

Figure 1.1: Ethernet stack module overview

Note: The Ethernet Driver is specified in a way that allows for object code delivery of the code module, following the "one-fits-all" principle, i.e. the entire configuration of the Ethernet Interface can be carried out without modifying any source code. Thus, the configuration of the Ethernet Driver can be carried out largely without detailed knowledge of the Ethernet Driver software.

2 Acronyms and abbreviations

Abbreviation / Acronym:	*Description:*
EC	Ethernet controller
Eth	Ethernet Driver (AUTOSAR BSW module)
EthIf	Ethernet Interface (AUTOSAR BSW module)
EthTrcv	Ethernet Transceiver Driver (AUTOSAR BSW module)
ISR	Interrupt Service Routine
MCG	Module Configuration Generator
MII	Media Independent Interface (standardized Interface provided by Ethernet controllers to access Ethernet transceivers)
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] AUTOSAR General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[4] Specification of Communication
AUTOSAR_SWS_COM.pdf

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

[6] Specification of Ethernet Interface
AUTOSAR_SWS_EthernetInterface.pdf

[7] Specification of Ethernet State Manager
AUTOSAR_SWS_EthernetStateManager.pdf

[8] Specification of Ethernet Transceiver Driver
AUTOSAR_SWS_EthernetTransceiver.pdf

[9] Specification of Socket Adapter
AUTOSAR_SWS_SocketAdapter.pdf

[10] Specification of UDP Network Management
AUTOSAR_SWS_UDPNetworkManagement.pdf

[11] Specification of PDU Router
AUTOSAR_SWS_PDURouter.pdf

[12] BSW Scheduler Specification
AUTOSAR_SWS_Scheduler.pdf

[13] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[14] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

[15] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

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

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 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 Driver module.

Modules that use Ethernet Driver module:

- Ethernet Interface (EthIf)
- Ethernet Transceiver Driver (EthTrcv)

Modules used by the Ethernet Driver 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:

- On certain systems the controller might share resources with other components (e.g. the MCU, Port), and may depend on their configuration. If those resources are within scope of the other modules (e.g. PLL configuration, memory mapping, etc.) the Ethernet Driver module does not take care of configuring those components but requires their preceding initialization.

5.1 File structure

5.1.1 Code file structure

ETH001:
This specification shall not completely define the code file structure. The code-file structure shall include the following files named:

- Eth_Lcfg.c – link time configurable parameters and
- Eth_PBcfg.c – post build time configurable parameters.

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

5.1.2 Header file structure

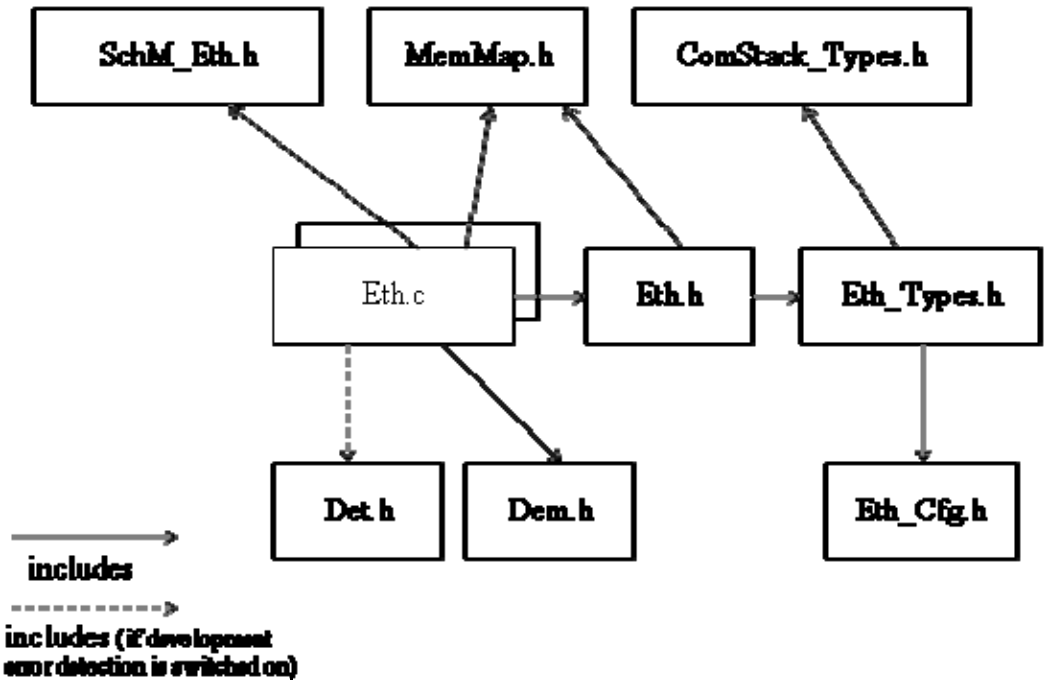

Figure 5.1 Ethernet Driver file structure

ETH002:
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

Document: AUTOSAR requirements on Basic Software, general

Requirement	***Satisfied by***
[BSW00344] Reference to link time configuration	Chapter 10.2.1
[BSW00404] Reference to post build time configuration	Chapter 10.2.1
[BSW00405] Reference to multiple configuration sets	Chapter 10
[BSW00345] Pre--compile--time configuration	Chapter 10.2.1
[BSW00159] Tool based configuration.	Chapter 10.2
[BSW00167] Static configuration checking	Chapter 7.1.3
[BSW00171] Configurability of optional functionality	Chapter 10.2
[BSW00170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable (no interface to SW-Components)
[BSW00380] Separate C-Files for configuration parameters	Chapter 5.1.1
[BSW00419] Separate C-Files for pre-compile time configuration parameters	Chapter 5.1.1
[BSW00381] Separate configuration header file for pre--compile time parameters	Chapter 5.1.2
[BSW00412] Separate H--File for configuration parameters	Chapter 5.1.2
[BSW00383] List dependencies of configuration files	Chapter 5.1.2
[BSW00384] List dependencies to other modules	Chapter 5
[BSW00387] Specify the configuration class of callback function	Chapter 8.4
[BSW00388] Introduce containers	Chapter 10.1.3
[BSW00389] Containers shall have names	Chapter 10.2
[BSW00390] Parameter content shall be unique within the module	Chapter 10.2
[BSW00391] Parameter shall have unique names	Chapter 10.2
[BSW00392] Parameters shall have a type	Chapter 10.2
[BSW00393] Parameters shall have a range	Chapter 10.2
[BSW00394] Specify the scope of the parameters	Chapter 10.2

[BSW00395] List the required parameters (per parameter)	Chapter 10.2
[BSW00396] Configuration classes	Chapter 10.2
[BSW00397] Pre-compile-time parameters	Chapter 10
[BSW00398] Link-time parameters	Chapter 10
[BSW00399] Loadable Post-build time parameters	Chapter 10
[BSW00400] Selectable Post--build time parameters	Chapter 10
[BSW00438] Post Build Configuration Data Structure	Chapter 10
[BSW00402] Published information	Chapter 10.3
[BSW00375] Notification of wake--up reason	Not relevant
[BSW00101] Initialization interface	Chapter 8.3.1
[BSW00416] Sequence of Initialization	Not relevant
[BSW00406] Check module initialization	Chapter 8
[BSW00437] NoInit-Area in RAM	Not relevant
[BSW00168] Diagnostic Interface of SW components	Not relevant
[BSW00407] Function to read out published parameters	Chapter 8.3.13
[BSW00423] Usage of SW--C template to describe BSW modules with AUTOSAR Interfaces	Not relevant
[BSW00424] BSW main processing function task allocation	Not relevant
[BSW00425] Trigger conditions for schedulable objects	Not relevant
[BSW00426] Exclusive areas in BSW modules	Not relevant
[BSW00427] ISR description for BSW modules	Not relevant
[BSW00428] Execution order dependencies of main processing functions	No dependencies
[BSW00429] Restricted BSW OS functionality access	Not relevant
[BSW00431] The BSW Scheduler module implements task bodies	Chapter 8
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not relevant
[BSW00433] Calling of main processing functions	Not relevant
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Chapter 8.7.1
[BSW00336] Shutdown interface	Not relevant

[BSW00337] Classification of errors	Chapter 7.2
[BSW00338] Detection and Reporting of development errors	Chapter 7.3,7.4
[BSW00369] Do not return development error codes via API	Chapter 8
[BSW00339] Reporting of production relevant error status	Not relevant
[BSW00422] Pre-de-bouncing of production relevant error status	Not relevant
[BSW00417] Reporting of Error Events by Non-Basic Software	Not relevant
[BSW00323] API parameter checking	Chapter 8
[BSW00004] Version check	Chapter 8.3.13
[BSW00409] Header files for production code error IDs	Chapter 5.1.2
[BSW00385] List possible error notifications	Chapter 7.4
[BSW00386] Configuration for detecting an error	Not relevant
[BSW00161] Microcontroller abstraction	Not relevant
[BSW00162] ECU layout abstraction	Not relevant
[BSW00005] No hard coded horizontal interfaces within MCAL	Not relevant
[BSW00415] User dependent include files	Chapter 5.1.2
[BSW00164] Implementation of interrupt service routines	Not relevant
[BSW00325] Runtime of interrupt service routines	Not relevant
[BSW00326] Transition from ISRs to OS tasks	Not relevant
[BSW00342] Usage of source code and object code	Chapter 7.1.2, 10.2.1
[BSW00343] Specification and configuration of time	Not relevant
[BSW00160] Human-readable configuration data	Chapter 10.2
[BSW00007] HIS MISRA C	Chapter 8
[BSW00300] Module naming convention	Chapter 5.1
[BSW00413] Accessing instances of BSW modules	Not relevant
[BSW00347] Naming separation of different instances of BSW drivers	Not relevant
[BSW00441] Enumeration literals and #define naming convention	Chapter 8.2
[BSW00305] Data types naming convention	Chapter 8.2
[BSW00307] Global variables naming convention	Chapter 8.2
[BSW00310] API naming convention	Chapter 8

[BSW00373] Main processing function naming convention	Not relevant
[BSW00327]Error values naming convention	Chapter 7.2
[BSW00335] Status values naming convention	Chapter 8.2.3
[BSW00350] Development error detection keyword	Chapter 7.3
[BSW00408] Configuration parameter naming convention	Chapter 10.2
[BSW00410] Compiler switches shall have defined values	Chapter 10.2
[BSW00411] Get version info keyword	Chapter 8.3.13
[BSW00346] Basic set of module files	Chapter 5.1
[BSW00158] Separation of configuration from implementation	Chapter 5.1
[BSW00314] Separation of interrupt frames and service routines	Not relevant
[BSW00370] Separation of callback interface from API	Chapter 8.4
[BSW00435] Module Header File Structure for the Basic Software Scheduler	Chapter 5.1.2
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	Chapter 5.1.2
[BSW00348] Standard type header	Chapter 5.1.2 (ComStack_Types.h includes Std_Types.h)
[BSW00353] Platform specific type header	Not relevant
[BSW00361] Compiler specific language extension header	Chapter 8
[BSW00301] Limit imported information	Chapter 5.1.2
[BSW00302] Limit exported information	Chapter 8
[BSW00328] Avoid duplication of code	Not relevant
[BSW00312] Shared code shall be reentrant	Not relevant
[BSW00006] Platform independency	Chapter 8
[BSW00439] Declaration of interrupt handlers and ISRs	Not relevant
[BSW00357] Standard API return type	Chapter 8.3
[BSW00377] Module specific API return types	Not relevant
[BSW00304] AUTOSAR integer data types	Chapter 8
[BSW00355] Do not redefine AUTOSAR integer data types	Chapter 8.2
[BSW00378] AUTOSAR boolean type	Chapter 8.3

[BSW00306] Avoid direct use of compiler and platform specific keywords	Chapter 8
[BSW00308] Definition of global data	Not relevant
[BSW00309] Global data with read--only constraint	Not relevant
[BSW00371] Do not pass function pointers via API	Chapter 8.3
[BSW00358] Return type of `init()` functions	Not relevant
[BSW00414] Parameter of init function	Chapter 8.3.1
[BSW00376] Return type and parameters of main processing functions	Not relevant
[BSW00359] Return type of callback functions	Chapter 8.4
[BSW00360] Parameters of callback functions	Chapter 8.4
[BSW00440] Function prototype for callback functions of AUTOSAR Services	Not relevant
[BSW00329] Avoidance of generic interfaces	Chapter 8
[BSW00330] Usage of macros / inline functions instead of functions	Not relevant
[BSW00331] Separation of error and status values	Chapter 7.2 and 8.2.3
[BSW00009] Module User Documentation	Entire document
[BSW00401] Documentation of multiple instances of configuration parameters	Chapter 10.2
[BSW00172] Compatibility and documentation of scheduling strategy	Chapter 8
[BSW00010] Memory resource documentation	Implementation specific
[BSW00333] Documentation of callback function context	Chapter 8.4
[BSW00374] Module vendor identification	Chapter 10.3
[BSW00379] Module identification	Chapter 10.3
[BSW00003] Version identification	Chapter 10.3
[BSW00318] Format of module version numbers	Chapter 10.3
[BSW00321] Enumeration of module version numbers	Implementation specific
[BSW00341] Microcontroller compatibility documentation	Not relevant
[BSW00334] Provision of XML file	Not relevant

Document: AUTOSAR Ethernet Requirements [5]

Requirement	***Satisfied by***
[BSW41900044] The TCP/IP stack is not an AUTOSAR module	Not relevant

[BSW41900045] TCPIP automatic IP address assignment	Not relevant
[BSW41900014] TCPIP IPv4 implementation	Not relevant
[BSW41900015] TCPIP ARP implementation	Not relevant
[BSW41900016] TCPIP ICMP implementation	Not relevant
[BSW41900017] TCPIP TCP implementation	Not relevant
[BSW41900018] TCPIP UDP implementation	Not relevant
[BSW41900019] TCPIP TCP+UDP implementation	Not relevant
[BSW41900020] TCPIP DHCP implementation	Not relevant
[BSW41900021] TCPIP DHCP “host name option” implementation	Not relevant
[BSW41900022] TCPIP link local IP implementation	Not relevant
[BSW41900046] SoAd DoIP implementation	Not relevant
[BSW41900004] SoAd Multi-homed hosts	Not relevant
[BSW41900002] SoAd IP address configuration	Not relevant
[BSW41900001] SoAd TCP connection setup	Not relevant
[BSW41900001] SoAd TCP connection setup	Not relevant
[BSW41900005] SoAd Use of UDP and TCP	Not relevant
[BSW41900009] SoAd Connection shutdown	Not relevant
[BSW41900008] SoAd immediate retry	Not relevant
[BSW41900007] SoAd COTS Compatibility	Not relevant
[BSW41900010] SoAd Resource management	Not relevant
[BSW41900011] SoAd Resource predictability	Not relevant
[BSW41900012] SoAd No buffer memory	Not relevant
[BSW41900013] SoAd Reduced Copy operation	Not relevant
[BSW41900006] SoAd No Protocol overhead	Not relevant
[BSW41900048] SoAd PDU routing	Not relevant
[BSW41900047] DoIp DHCP "host name option" access	Not relevant
[BSW41900024] DoIP routing	Not relevant

[BSW41900025] DoIP message recognition	Not relevant
[BSW41900026] DoIP Vehicle Identification	Not relevant
[BSW41900027] DoIP diagnostic message	Not relevant
[BSW41900028] DoIP Socket handling	Not relevant
[BSW41900029] EthIf: Interface of the module	Not relevant
[BSW41900030] EthIf: Hardware abstraction	Not relevant
[BSW41900031] EthIf: Interrupt / Polling mode	Not relevant
[BSW41900032] EthIf: Hardware configuration and initialization	Not relevant
[BSW41900033] EthIf: Link state change indication	Not relevant
[BSW41900034] Eth: Hardware abstraction	Chapter 8.3
[BSW41900035] Eth: Interrupt / Polling mode	Chapter 8.5, 8.3
[BSW41900036] Eth: Hardware configuration and initialization	Chapter 10.2
[BSW41900038] EthTrcv: Hardware abstraction	Not relevant
[BSW41900039] EthTrcv: Hardware configuration and initialization	Not relevant
[BSW41900040] EthTrcv: Link state change indication	Not relevant
[BSW41900041] EthSM: Network abstraction	Not relevant
[BSW41900043] EthSM: Network configuration and initialization	Not relevant
[BSW41900042] UdpNm: Network abstraction	Not relevant
[BSW41900037] UdpNm: Network management information	Not relevant

7 Functional specification

7.1 Ethernet BSW stack

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

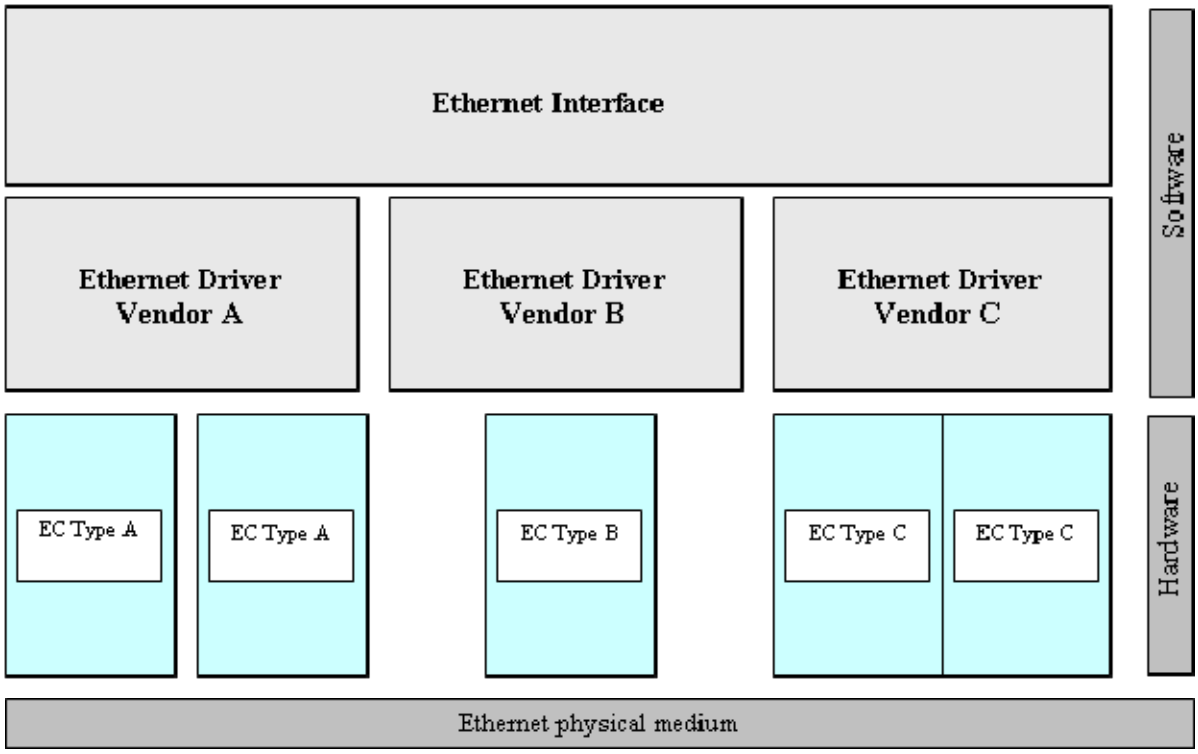

Figure 7.1: Basic Structure of the Ethernet BSW stack

7.1.1 Indexing scheme

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

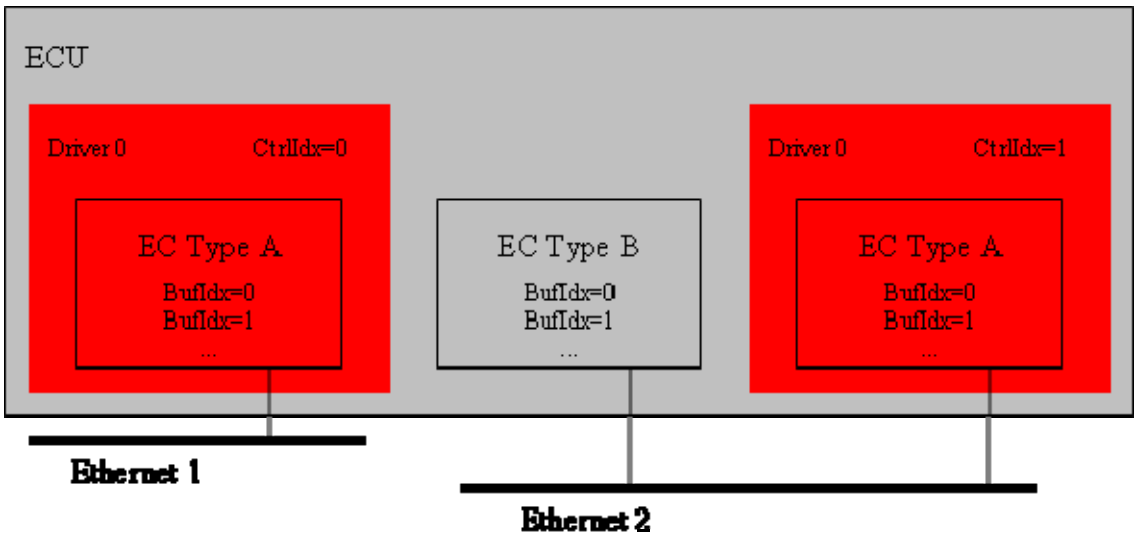

Figure 7.2: Ethernet Driver indexing scheme

ETH003:
The Ethernet Driver is using a zero-based index to abstract the access for upper software layers. The parameter Eth_CtrlIdx within configuration corresponds to parameter CtrlIdx used in the API.

ETH004:
A buffer index (BufIdx) indentifies an Ethernet buffer processed by Ethernet Driver API functions. Each controller's buffers are identified by buffer indexes 0 to (n-1) where n is the number of buffers processed by the corresponding controller. Buffer indexes are valid within a tuple <CtrlIdx, BufIdx> only. A BufIdx uniquely identifies the buffer used for an Ethernet Driver.

7.1.2 Requirements

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

ETH005:
The Ethernet Driver module shall support pre-compile time, link time and post-build time configuration.

ETH006:
The header file *Eth.h* shall include a software and specification version number.

ETH007:
The Ethernet Driver module shall perform a consistency check between code files and header files based on pre-process-checking the version numbers of related code files and header files.

ETH008:

In case development error detection is enabled for the Ethernet Driver module: The Ethernet Driver module shall check API parameters for validity and report detected errors to the DET.

DET API functions are specified in [16].

ETH009:
The Ethernet Driver module implementation shall conform to the HIS subset of the MISRA C Standard (see document [18]).

ETH010:
The Ethernet Driver module shall implement the API functions specified by the Ethernet Driver SWS as real C-code functions and shall not implement the API as macros for object code deliveries.

ETH011:
None of the Ethernet Driver module header files shall define global variables.

7.1.3 Configuration description

ETH012:
The Ethernet Driver module shall provide an XML file that contains the data, which is required for the SW identification (it shall contain the vendor identification, module ID and software version information), configuration and integration process. This file should describe vendor specific configuration parameters as well as it should contain recommended configuration parameter values.

ETH125:
The MCG shall read the ECU configuration description of the Ethernet Driver module(s). Ethernet Driver related configuration data is contained in the Ethernet Driver module configuration description.

ETH126:
The MCG shall ensure the consistency of the generated configuration data.

ETH013:
The configuration of the Ethernet Driver module shall be calculated at ECU configuration time. None of the communication parameters shall be calculated at runtime.

ETH014:
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 Driver related configuration parameters can be found in chapter 10 of this document.

7.2 Error classification

ETH015:
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.

ETH016:
Development error values are of type uint8.

Type or error	***Relevance***	***Related error code***	***Value [hex]***
Invalid controller index	Development	ETH_E_INV_CTRL_IDX	0x01
Eth module or controller was not initialized	Development	ETH_E_NOT_INITIALIZED	0x02
Invalid pointer in parameter list	Development	ETH_E_INV_POINTER	0x03
Invalid parameter	Development	ETH_E_INV_PARAM	0x04
Invalid configuration	Development	ETH_E_INV_CONFIG	0x05
Invalid mode	Development	ETH_E_INV_MODE	0x06
Controller access failed	Production	ETH_E_ACCESS	Assigned by DEM

7.3 Error detection

ETH017:
The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch *EthDevErrorDetect* (see chapter 10) shall activate or deactivate the detection of all development errors.

ETH018:
The *EthDevErrorDetect* switch enables API parameter checking. Chapter 7.2 and 8 contain the detailed description of the detected errors.

ETH019:
Switching off the detection of production code errors shall not be possible.

7.4 Error notification

ETH020:
The module shall report development errors to the *Det_ReportError* service of the Development Error Tracer (DET) if the pre-processor switch *EthDevErrorDetect* is set (see chapter 10).

ETH021:
The module shall report production errors to the Diagnostic Event Manager.

7.5 Debugging

ETH022:
Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable.

ETH023:
All type definitions of variables, which shall be debugged, shall be accessible by the header file Eth.h.

ETH024:
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”.

ETH025:
Variables available for debugging shall be described in the respective Basic Software Module Description.

7.6 Version checking

ETH135:
The Ethernet Driver module shall perform inter-module checks to avoid integration of incompatible files.
The imported include files shall be checked by preprocessing directives.

The Ethernet Driver 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 Driver module.

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

8 API specification

8.1 Imported types

This chapter lists all types included from the following files:

ETH026:

Module	*Imported Type*
ComStack_Types	BufReq_ReturnType
Dem	Dem_EventIdType
	Dem_EventStatusType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Type definitions

8.2.1 Eth_ConfigType

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

8.2.2 Eth_ModeType

Name:	`Eth_ModeType`	
Type:	`Enumeration`	
Range:	`ETH_MODE_DOWN`	Controller disabled
	`ETH_MODE_ACTIVE`	Controller enabled
Description:	This type defines the controller modes	

8.2.3 Eth_StateType

Name:	`Eth_StateType`	
Type:	`Enumeration`	
Range:	`ETH_STATE_UNINIT`	Driver is not yet configured
	`ETH_STATE_INIT`	Driver is configured
	`ETH_STATE_ACTIVE`	Driver is active
Description:	Status supervision used for Development Error Detection. The state shall be available for debugging.	

8.2.4 Eth_FrameType

Name:	Eth_FrameType		
Type:	--		
Range:	uint16	0x0000 - 0xFFFF	See [16]
Description:	This type defines the Ethernet frame type used in the Ethernet frame header		

8.2.5 Eth_DataType

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

8.3 Function definitions

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

8.3.1 Eth_Init

ETH027:

Service name:	Eth_Init

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

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

ETH028:
The function shall store the access to the configuration structure for subsequent API calls.

ETH029:

The function shall change the state of the component from ETH_STATE_UNINIT to ETH_STATE_INIT.

ETH030:
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 ETH_E_INV_POINTER.

ETH031:
Caveat: The API has to be called during initialization.

ETH032:
Configuration: The user shall pass the post-build configuration or a NULL_PTR as parameter depending on the configuration variant.

8.3.2 Eth_ControllerInit

ETH033:

Service name:	Eth_ControllerInit	

Syntax:	`Std_ReturnType Eth_ControllerInit(`
	`  uint8 CtrlIdx,`
	`  uint8 CfgIdx`
	`)`	

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

ETH034:
The function shall:

- Disable the controller
- Clear pending Ethernet interrupts
- Configure all controller configuration parameters (e.g. interrupts, frame length, frame filter, ...)
- Configure all transmit / receive resources (e.g. buffer initialization)
- delete all pending transmit and receive requests

ETH035:
The function shall change the state of the component from ETH_STATE_INIT to ETH_STATE_ACTIVE.

ETH036:
If development error detection is enabled: the function shall check that the service Eth_Init was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED and return E_NOT_OK.

ETH037:
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX and return E_NOT_OK.

ETH038:
If development error detection is enabled: the function shall check the parameter CfgIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CONFIG and return E_NOT_OK.

ETH039:
The function shall check the access to the Ethernet controller. If the check fails, the function shall raise the production error ETH_E_ACCESS and return E_NOT_OK.

ETH040:
Caveat: The function requires previous initialization (Eth_Init).

8.3.3 Eth_SetControllerMode

ETH041:

Service name:	Eth_SetControllerMode	

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

Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	CtrlMode	ETH_MODE_DOWN: disable the controller ETH_MODE_ACTIVE: enable the controller
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: transceiver could not be initialized
Description:	Enables / disables the indexed controller	

ETH042:
The function shall:

- Put the controller in the specified mode given in the parameter ‘CtrlMode’
  - Upon mode ETH_MODE_DOWN the driver shall:
    - Disable the Ethernet controller

- Reset all transmit and receive buffers (i.e. ignore all pending transmission and reception requests)

o Upon mode ETH_MODE_ACTIVE:

- Enable all transmit and receive buffers
- Enable the Ethernet controller

ETH043:
If development error detection is enabled: the function shall check that the service Eth_ControllerInit was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED and return E_NOT_OK.

ETH044:
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX and return E_NOT_OK.

ETH045:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.4 Eth_GetControllerMode

ETH046:

Service name:	Eth_GetControllerMode	

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

Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
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 mode could not be obtained
Description:	Obtains the state of the indexed controller	

ETH047:
The function shall read the current controller mode.

ETH048:
If development error detection is enabled: the function shall check that the service Eth_ControllerInit was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED and return E_NOT_OK.

ETH049:

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

ETH050:
If development error detection is enabled: the function shall check the parameter CtrlModePtr for being valid. If the check fails, the function shall raise the development error ETH_E_INV_POINTER and return E_NOT_OK.

ETH051:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.5 Eth_GetPhysAddr

ETH052:

Service name:	Eth_GetPhysAddr	

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

Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
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:	void	None
Description:	Obtains the physical source address used by the indexed controller	

ETH053:
The function shall read the source address used by the indexed controller.

ETH054:
If development error detection is enabled: the function shall check that the service Eth_ControllerInit was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED.

ETH055:
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.

ETH056:
If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETH_E_INV_POINTER.

ETH057:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.6 Eth_WriteMii

ETH058:

Service name:	Eth_WriteMii	

Syntax:	`void Eth_WriteMii(`
	`  uint8 CtrlIdx,`
	`  uint8 TrcvIdx,`
	`  uint8 RegIdx,`
	`  uint16 RegVal`
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	TrcvIdx	Index of the transceiver on the MII (see [16] for details)
	RegIdx	Index of the transceiver register on the MII (see [16] for details)
	RegVal	Value to be written into the indexed register (see [16] for details)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	void	None
Description:	Configures a transceiver register or triggers a function offered by the receiver	

ETH059:
The function shall write the specified transceiver register through the MII of the indexed controller.

ETH060:
If development error detection is enabled: the function shall check that the service Eth_ControllerInit was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED.

ETH061:
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.

ETH128:
If development error detection is enabled: the function shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function shall raise the development error ETH_E_INV_MODE.

ETH062:

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

ETH063:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.7 Eth_ReadMii

ETH064:

Service name:	Eth_ReadMii	
Syntax:	`void Eth_ReadMii( uint8 CtrlIdx, uint8 TrcvIdx, uint8 RegIdx, uint16* RegValPtr )`	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	TrcvIdx	Index of the transceiver on the MII (see [16] for details)
	RegIdx	Index of the transceiver register on the MII (see [16] for details)
Parameters (inout):	None	
Parameters (out):	RegValPtr	Filled with the register content of the indexed register (see [16] for details)
Return value:	void	None
Description:	Reads a transceiver register	

ETH065:
The function shall read the specified transceiver register through the MII of the indexed controller.

ETH066:
If development error detection is enabled: the function shall check that the service Eth_ControllerInit was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED.

ETH067:
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.

ETH068:
If development error detection is enabled: the function shall check the parameter RegValPtr for being valid. If the check fails, the function shall raise the development error ETH_E_INV_POINTER.

ETH127:
If development error detection is enabled: the function shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function shall raise the development error ETH_E_INV_MODE.

ETH069:
The function shall be pre compile time configurable On/Off by the configuration parameter: EthCtrlEnableMii.

ETH070:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.8 Eth_GetCounterState

ETH071:

Service name:	Eth_GetCounterState	

Syntax:	`void Eth_GetCounterState(`
	`  uint8 CtrlIdx,`
	`  uint16 CtrOffs,`
	`  uint32* CtrValPtr`
	`)`	

Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	CtrOffs	Memory offset of the counter. The offset is controller specific.
Parameters (inout):	None	
Parameters (out):	CtrValPtr	Filled with the content of the specified counter
Return value:	void	None
Description:	Reads the value of a counter specified with its memory offset	

ETH072:
The function shall read the specified counter register of the indexed controller.

ETH073:
If development error detection is enabled: the function shall check that the service Eth_ControllerInit was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED.

ETH074:
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.

ETH075:

If development error detection is enabled: the function shall check the parameter CtrValPtr for being valid. If the check fails, the function shall raise the development error ETH_E_INV_POINTER.

ETH076:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.9 Eth_ProvideTxBuffer

ETH077:

Service name:	Eth_ProvideTxBuffer	

Syntax:	`BufReq_ReturnType Eth_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 controller within the context of the Ethernet Driver
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 controller	

ETH078:
The function shall provide a transmit buffer resource. The Ethernet Driver shall lock the buffer until it receives a subsequent call of Eth_Transmit service with the buffer index returned in the BufIdxPtr parameter.

ETH137:
All locked transmit buffers shall be released if the controller is disabled via Eth_SetControllerMode.

ETH079:
If the requested buffer length is larger than the available buffer length the component shall lock the available buffer and return BUFREQ_OK.

ETH080:
If all available buffers are in use the component shall return BUFREQ_E_BUSY.

ETH081:

If development error detection is enabled: the function shall check that the service Eth_ControllerInit was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED and return BUFREQ_E_NOT_OK.

ETH082:
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX and return BUFREQ_E_NOT_OK.

ETH083:
If development error detection is enabled: the function shall check the parameter BufIdxPtr for being valid. If the check fails, the function shall raise the development error ETH_E_INV_POINTER and return BUFREQ_E_NOT_OK.

ETH084:
If development error detection is enabled: the function shall check the parameter BufPtr for being valid. If the check fails, the function shall raise the development error ETH_E_INV_POINTER and return BUFREQ_E_NOT_OK.

ETH085:
If development error detection is enabled: the function shall check the parameter LenBytePtr for being valid. If the check fails, the function shall raise the development error ETH_E_INV_POINTER and return BUFREQ_E_NOT_OK.

ETH086:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.10 Eth_Transmit

ETH087:

Service name:	Eth_Transmit	

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

Service ID[hex]:	0xA	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the controller within the context of the Ethernet Driver
	BufIdx	Index of the buffer resource
	FrameType	Ethernet frame type
	TxConfirmation	Activates transmission confirmation
	LenByte	Data length in byte
	PhysAddrPtr	Physical target address (MAC address) in network byte order
Parameters (inout):	None	

Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: transmission failed
Description:	Triggers transmission of a previously filled transmit buffer	

ETH088:
The function shall build the Ethernet header with the given physical target address (MAC address) and trigger the transmission of a previously filled transmit buffer.

ETH089:
If TxConfirmation is false, the function shall release the buffer resource.

ETH138:
All pending transmit buffers shall be released if the controller is disabled via Eth_SetControllerMode.

ETH090:
If development error detection is enabled: the function shall check that the service Eth_ControllerInit was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED and return E_NOT_OK.

ETH091:
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX and return E_NOT_OK.

ETH092:
If development error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_PARAM and return E_NOT_OK.

ETH093:
If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETH_E_INV_POINTER and return E_NOT_OK.

ETH129:
If development error detection is enabled: the function shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function shall raise the development error ETH_E_INV_MODE.

ETH094:
Caveat: The function requires previous buffer request (Eth_ProvideTxBuffer).

8.3.11 Eth_Receive

ETH095:

Service name:	Eth_Receive

Syntax:	`void Eth_Receive(`
	`  uint8 CtrlIdx`
	`)`

Service ID[hex]:	0xB
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	CtrlIdx: Index of the controller within the context of the Ethernet Driver
Parameters (inout):	None
Parameters (out):	None
Return value:	void: None
Description:	Triggers frame reception

ETH096:
The function shall read the frames of all filled receive buffers. The function passes each received frame to the Ethernet interface using the callback function EthIf_Cbk_RxIndication.

ETH097:
If development error detection is enabled: the function shall check that the service Eth_ControllerInit was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED.

ETH098:
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.

ETH132:
If development error detection is enabled: the function shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function shall raise the development error ETH_E_INV_MODE.

ETH099:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.12 Eth_TxConfirmation

ETH100:

Service name:	Eth_TxConfirmation

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

Service ID[hex]:	0xC
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	CtrlIdx: Index of the controller within the context of the Ethernet Driver
Parameters	None

(inout):	
Parameters (out):	None
Return value:	void None
Description:	Triggers frame transmission confirmation

ETH101:
The function shall check all filled transmit buffers for successful transmission. The function issues transmit confirmation for each transmitted frame using the callback function EthIf_Cbk_TxConfirmation if requested by the previous call of Eth_Transmit service.

ETH102:
If transmission confirmation was enabled by a previous call to Eth_Transmit function the function shall release the buffer resource.

ETH103:
If development error detection is enabled: the function shall check that the service Eth_ControllerInit was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED.

ETH104:
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX.

ETH134:
If development error detection is enabled: the function shall check the controller mode for being active (ETH_MODE_ACTIVE). If the check fails, the function shall raise the development error ETH_E_INV_MODE.

ETH105:
Caveat: The function requires previous initialization (Eth_ControllerInit).

8.3.13 Eth_GetVersionInfo

ETH106:

Service name:	Eth_GetVersionInfo
Syntax:	`void Eth_GetVersionInfo( Std_VersionInfoType* VersionInfoPtr )`
Service ID[hex]:	0xD
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	VersionInfoPtr Version information of this module
Parameters (inout):	None
Parameters (out):	None
Return value:	void None

Description:	Returns the version information of this module

ETH107:
The function Eth_ 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.

ETH108:
The function Eth_ GetVersionInfo shall be pre compile time configurable On/Off by the configuration parameter: EthVersionInfoApi using the keyword ETH_GET_VERSION_INFO.

ETH136:
If development error detection is enabled: the function shall check the parameter VersionInfoPtr for being valid. If the check fails, the function shall raise the development error ETH_E_INV_POINTER.

8.4 Callback notifications

The Ethernet Driver does not provide any callback functions.

8.5 Interrupt service routines

This is a list of functions provided for interrupt handling.

8.5.1 Eth_RxIrqHdlr_<CtrlIdx>

ETH109:

Service name:	Eth_RxIrqHdlr_<CtrlIdx>

Syntax:	`void Eth_RxIrqHdlr_<CtrlIdx>(`
	`  void`
	`)`

Service ID[hex]:	0x10
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None

Return value:	void	None
Description:	Handles frame reception interrupts of the indexed controller	

ETH110:
The function shall clear the interrupt and read the frames of all filled receive buffers. The function passes each received frame to the Ethernet interface using the callback function EthIf_Cbk_RxIndication.

ETH111:
If development error detection is enabled: the function shall check that the service Eth_ControllerInit was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED.

ETH112:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

ETH113:
Caveat: The function shall be callable on interrupt level.

8.5.2 Eth_TxIrqHdlr_<CtrlIdx>

ETH114:

Service name:	Eth_TxIrqHdlr_<CtrlIdx>	

Syntax:	`void Eth_TxIrqHdlr_<CtrlIdx>(`
	`  void`
	`)`	

Service ID[hex]:	0x11	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	void	None
Description:	Handles frame transmission interrupts of the indexed controller	

ETH115:
The function shall clear the interrupt and check all filled transmit buffers for successful transmission. The function issues transmit confirmation for each transmitted frame using the callback function EthIf_Cbk_TxConfirmation if requested by the previous call of Eth_Transmit service.

ETH116:
If development error detection is enabled: the function shall check that the service Eth_ControllerInit was previously called. If the check fails, the function shall raise the development error ETH_E_NOT_INITIALIZED.

ETH117:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

ETH118:
Caveat: The function shall be callable on interrupt level.

8.6 Scheduled functions

The Ethernet Driver runs in the context of the Ethernet Interface and has thus no scheduled functions.

8.7 Expected Interfaces

This chapter lists all interfaces required from other modules.

8.7.1 Mandatory Interfaces

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

ETH119:

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.
EthIf_Cbk_RxIndication	Handles a received frame received by the indexed controller
EthIf_Cbk_TxConfirmation	Confirms frame transmission by the indexed controller
SchM_Enter_Eth	Invokes the SchM_Enter function to enter a module local exclusive area.
SchM_Exit_Eth	Invokes the SchM_Exit function to exit an exclusive area.

8.7.2 Optional Interfaces

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

ETH120:

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

8.7.3 Configurable interfaces

The Ethernet Driver does not use configurable interfaces.

Terms and definitions:

Reentrant: interface is expected to be reentrant
Don't care: reentrancy of interface not relevant for this module (in general it is in this case not reentrant).

9 Sequence diagrams

The usage of the Ethernet Driver is depicted in the sequence diagrams of the Ethernet Interface.

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

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

Chapter 10.3 specifies published information of the module Ethernet Driver.

10.1 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.

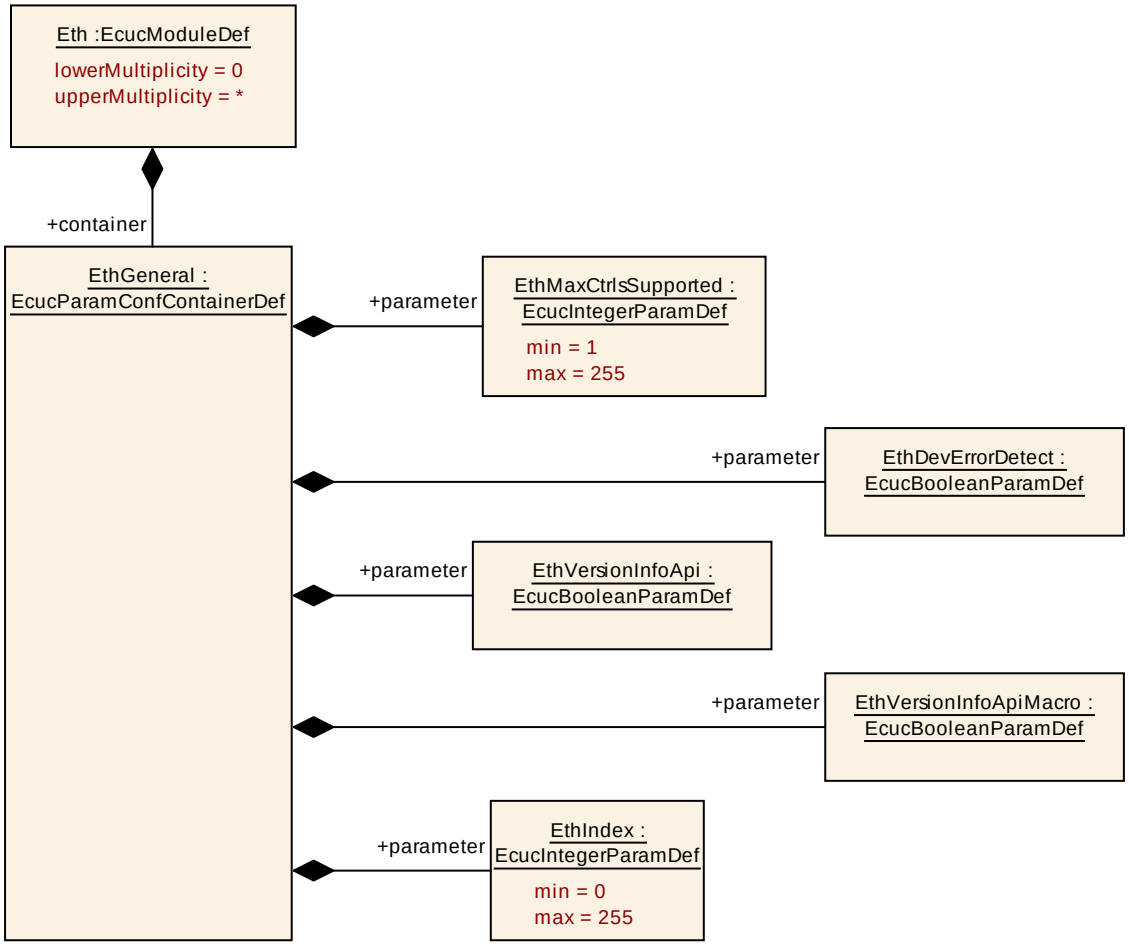

Figure 10.1: Ethernet Driver configuration structure

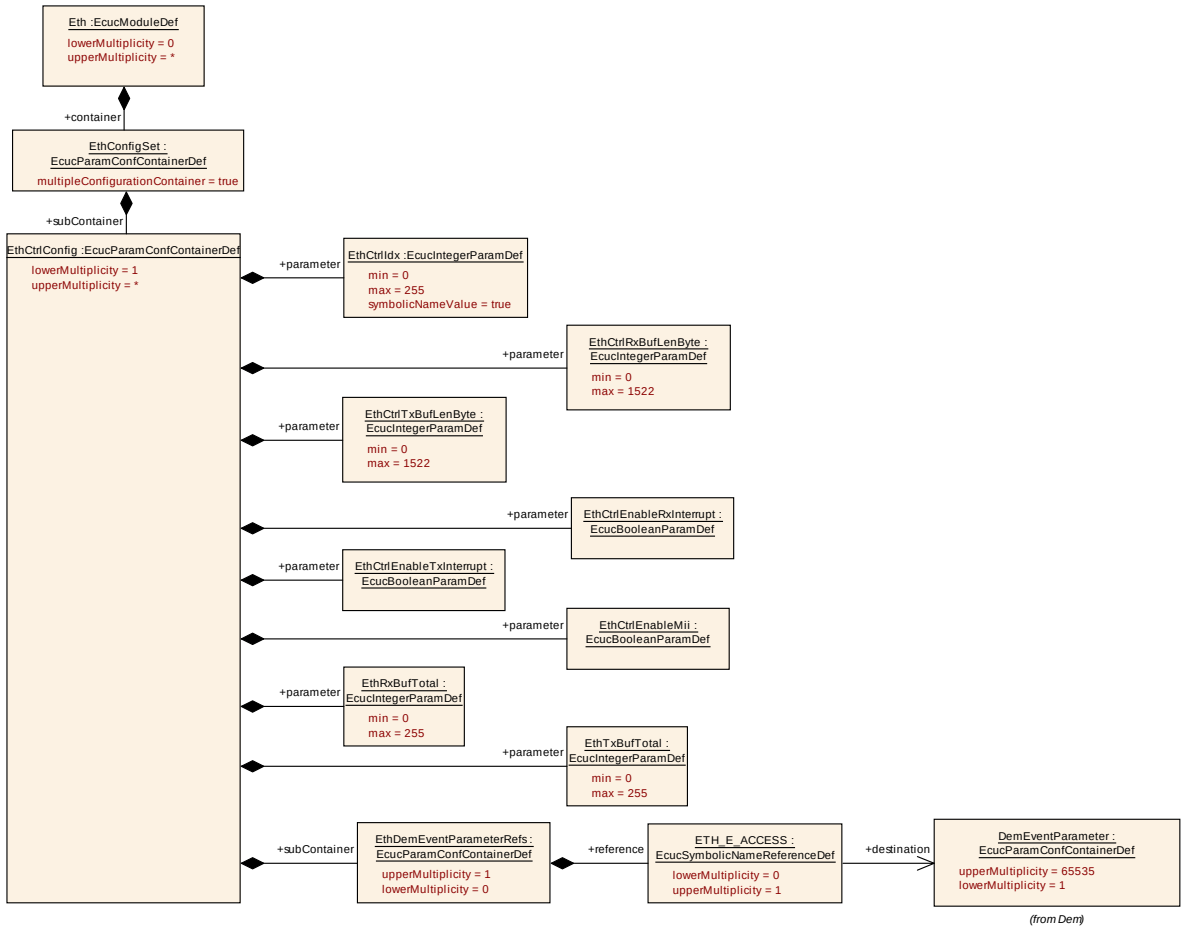

Figure 10.2: Ethernet Driver Controller configuration structure

10.2.1 Variants

ETH121:
VARIANT-POST-BUILD: All configuration parameters in container ‘EthGeneral’ shall be configurable at pre-compile time.
Use case: Object code delivery, selectable configuration

ETH122:
VARIANT-LINK-TIME: All configuration parameters in container ‘EthGeneral’ shall be configurable at pre-compile time.
Use case: Object code delivery, single configuration

ETH123:
VARIANT-PRE-COMPILE: All configuration parameters shall be configurable at pre-compile time.
Use case: Execution time optimizations, fix configuration

10.2.2 Eth

Module Name	*Eth*
Module Description	Configuration of the Eth (Ethernet Driver) module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthConfigSet	1	All included containers and parameters that may be part of a multiple configuration set.
EthGeneral	1	General configuration of Ethernet Driver module

10.2.3 EthConfigSet

SWS Item	**ETH015_Conf :**
Container Name	EthConfigSet [Multi Config Container]
Description	All included containers and parameters that may be part of a multiple configuration set.
Configuration Parameters	

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

10.2.4 EthCtrlConfig

SWS Item	**ETH006_Conf :**
Container Name	EthCtrlConfig
Description	Configuration of the individual controller
Configuration Parameters	

SWS Item	**ETH012_Conf :**		
Name	EthCtrlEnableMii		
Description	Enables / Disables Media Independent Interface (MII) for transceiver access		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ETH010_Conf :**
Name	EthCtrlEnableRxInterrupt
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	**ETH011_Conf :**		
Name	EthCtrlEnableTxInterrupt		
Description	Enables / Disables 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	**ETH007_Conf :**		
Name	EthCtrlIdx		
Description	Specifies the instance ID of the configured controller.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
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		

SWS Item	**ETH008_Conf :**		
Name	EthCtrlRxBufLenByte		
Description	Limits the maximum receive buffer length (frame length) in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 1522		
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	**ETH009_Conf :**		
Name	EthCtrlTxBufLenByte		
Description	Limits the maximum transmit buffer length (frame length) in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 1522		
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	ETH013_Conf :		
Name	EthRxBufTotal		
Description	Configures the number of receive buffers.		
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		

SWS Item	ETH014_Conf :		
Name	EthTxBufTotal		
Description	Configures the number of transmit buffers.		
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		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
EthDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.

10.2.5 EthDemEventParameterRefs

SWS Item	ETH016_Conf :
Container Name	EthDemEventParameterRefs
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
Configuration Parameters	

SWS Item	ETH017_Conf :		
Name	ETH_E_ACCESS		
Description	Reference to the DemEventParameter which shall be issued when the error "Controller access failed" has occured.		
Multiplicity	0..1		
Type	Reference to [DemEventParameter]		
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 EthGeneral

SWS Item	**ETH001_Conf :**
Container Name	EthGeneral
Description	General configuration of Ethernet Driver module
Configuration Parameters	

SWS Item	**ETH003_Conf :**		
Name	EthDevErrorDetect		
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	**ETH018_Conf :**		
Name	EthIndex		
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ETH002_Conf :**		
Name	EthMaxCtrlsSupported		
Description	Limits the total number of supported controllers.		
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	**ETH004_Conf :**
Name	EthVersionInfoApi
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	**ETH005_Conf :**		
Name	EthVersionInfoApiMacro		
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

10.3 Published Information

ETH124 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.