Document Title	Specification of Ethernet Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	430
Document Classification	Standard

Document Version	1.5.0
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History

Date	Version	Changed by	Change Description
31.03.2014	1.5.0	AUTOSAR Release Management	• Introduction of periodic call to Eth_SetControllerMode • Support of VLANs (Virtual Local Area Networks) • Editorial changes
31.10.2013	1.4.0	AUTOSAR Release Management	• Introduction of Eth_GeneralTypes.h • Support of API deviation for asynchronous implementation • Changes in API of EthIf_ProvideTxBuffer and EthIf_SetPhysAddr • Editorial changes • Removed chapter(s) on change documentation
28.02.2013	1.3.0	AUTOSAR Administration	• Configurable MAC address based filtering • Detection of lost Ethernet frames • Buffer handling enhancement
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

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

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.

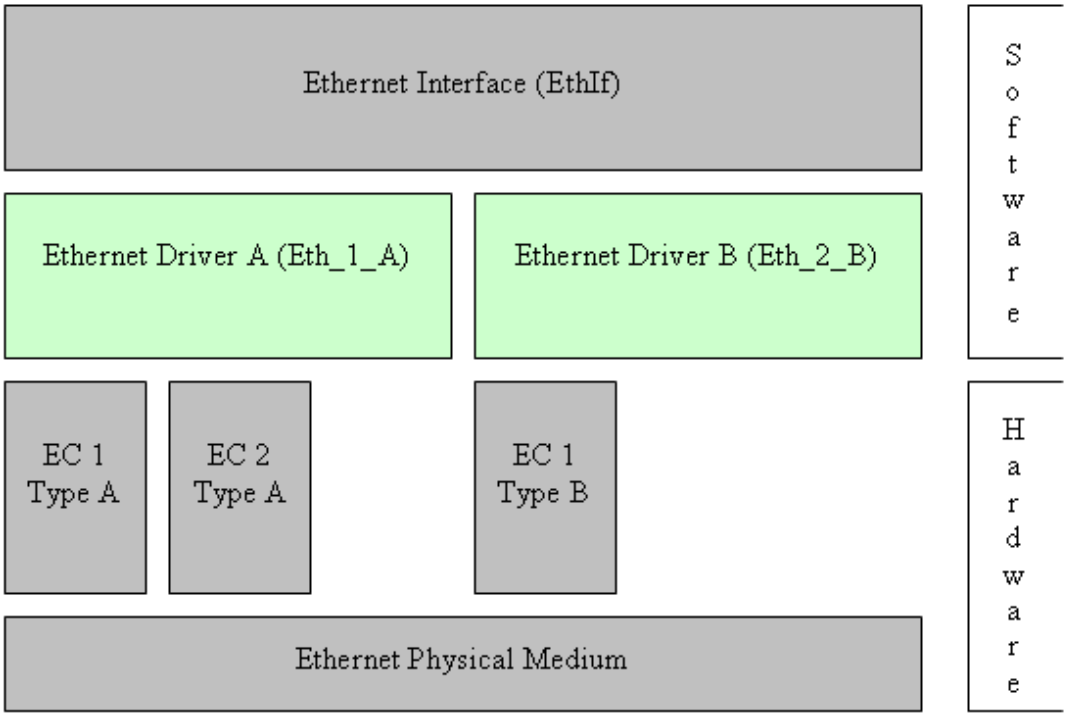

Figure 1.1: Ethernet stack module overview

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

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

3.2 Related standards and norms

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

[21] IEEE 802.3-2006

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [20] (SWS BSW General), which is also valid for Ethernet Driver.

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

4 Constraints and assumptions

4.1 Limitations

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

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).

Depending on the Ethernet hardware, it may become necessary that implementations deviate from API specifications in respect to the asynchronous/synchronous behaviour.

4.2 Applicability to car domains

The Ethernet BSW stack is intended to be used wherever high data rates are required but no hard real-time is required. Of course, it can also be used for less-demanding use cases, i.e. for low data rates.

5 Dependencies to other modules

This chapter lists the modules interacting with the Ethernet Driver module.

Modules that use Ethernet Driver module:

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

Modules used by the Ethernet Driver module:

- BSW Scheduler mechanisms for data consistency and main function handling.

Dependencies to other Modules:

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

5.1 File structure

5.1.1 Header file structure

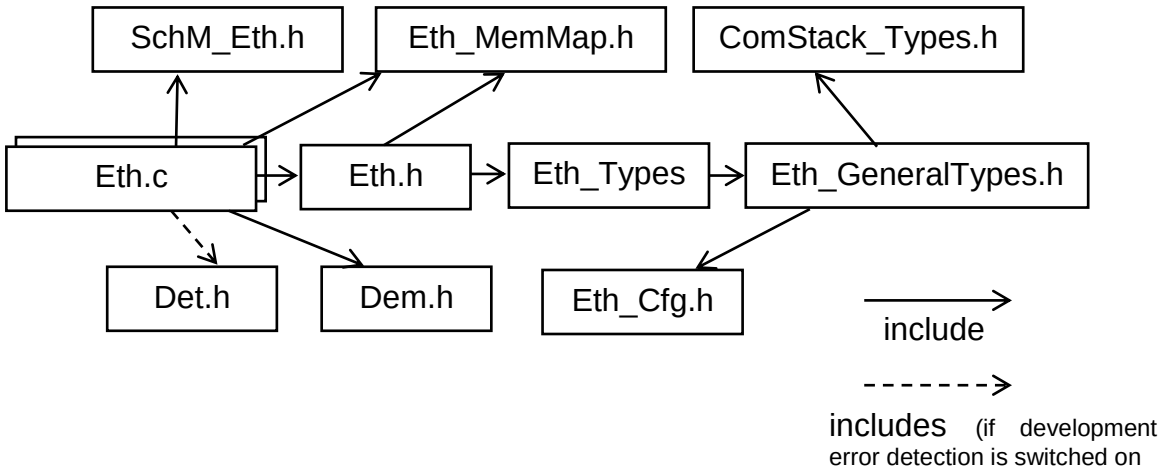

Figure 5.1 Ethernet Driver file structure

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_Eth_00139
-	-	SWS_Eth_00140
-	-	SWS_Eth_00141
-	-	SWS_Eth_00142
-	-	SWS_Eth_00143
-	-	SWS_Eth_00144
-	-	SWS_Eth_00146
-	-	SWS_Eth_00147
-	-	SWS_Eth_00148
-	-	SWS_Eth_00149
-	-	SWS_Eth_00150
-	-	SWS_Eth_00151
-	-	SWS_Eth_00152
-	-	SWS_Eth_00154
-	-	SWS_Eth_00164
-	-	SWS_Eth_00165
-	-	SWS_Eth_00166
-	-	SWS_Eth_00167

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.

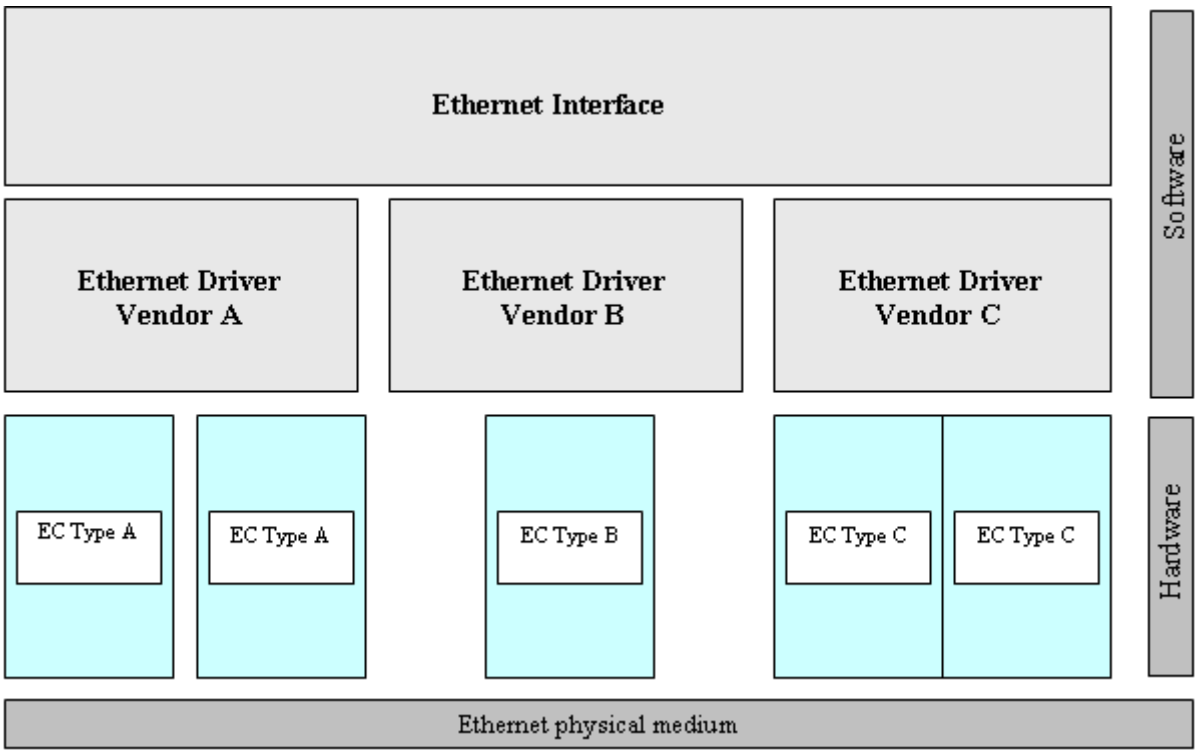

Figure 7.1: Basic Structure of the Ethernet BSW stack

7.1.1 Indexing scheme

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

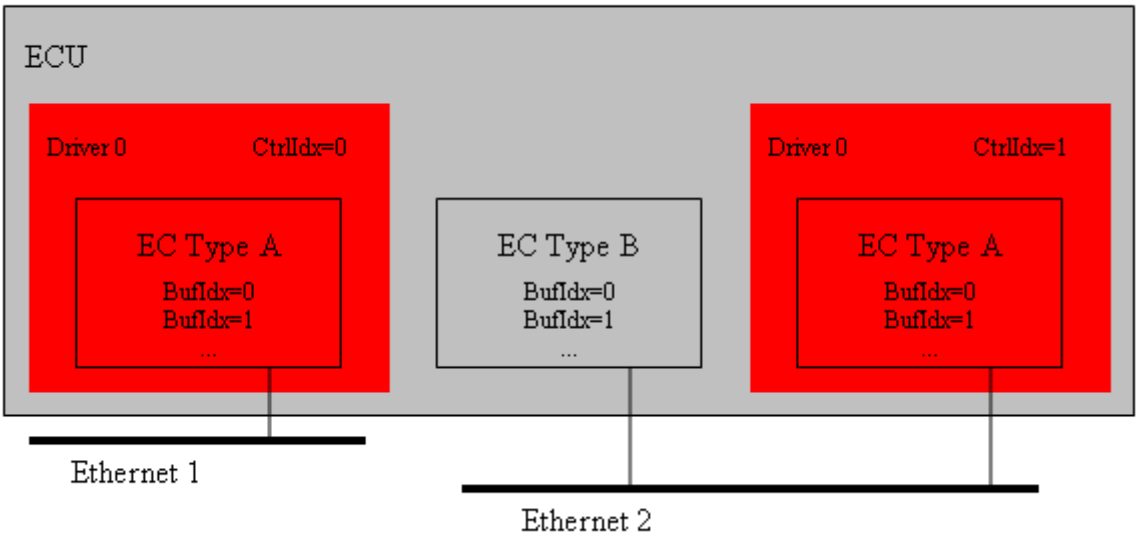

Figure 7.2: Ethernet Driver indexing scheme

SWS_Eth_00003:
The Ethernet Driver is using a zero-based index to abstract the access for upper software layers. The parameter Eth_CtrlIdx within configuration corresponds to parameter CtrlIdx used in the API.

SWS_Eth_00004:
A buffer index (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.

SWS_Eth_00005:
The Ethernet Driver module shall support pre-compile time, link time and post-build time configuration.

SWS_Eth_00006:
The header file *Eth.h* shall include a software and specification version number.

SWS_Eth_00007:

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

SWS_Eth_00008:
In case 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].

SWS_Eth_00009:
The Ethernet Driver module implementation shall conform to the HIS subset of the MISRA C Standard (see document [18]).

SWS_Eth_00011:
None of the Ethernet Driver module header files shall define global variables.

7.1.3 Configuration description

SWS_Eth_00012:
The Ethernet Driver module shall provide an XML file that contains the data, which is required for the SW identification (it shall contain the vendor identification, module ID and software version information), configuration and integration process. This file should describe vendor specific configuration parameters as well as it should contain recommended configuration parameter values.

SWS_Eth_00125:
The MCG shall read the ECU configuration description of the Ethernet Driver module(s). Ethernet Driver related configuration data is contained in the Ethernet Driver module configuration description.

SWS_Eth_00126:
The MCG shall ensure the consistency of the generated configuration data.

SWS_Eth_00013:
The configuration of the Ethernet Driver module shall be calculated at ECU configuration time. None of the communication parameters shall be calculated at runtime.

SWS_Eth_00014:
The start address of post-build time configuration data shall be passed during module initialization (see chapter 8.3.1).

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

A detailed description of all Ethernet Driver related configuration parameters can be found in chapter 10 of this document.

7.2 Development Errors

SWS_Eth_00016:

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
Ethernet frame lost	Development	ETH_E_FRAMES_LOST	0x07

7.3 Production Errors

None

7.4 Extended Production Errors

[SWS_Eth_00154]⌈

Type or error	*Related error code*	*Value [hex]*
Ethernet Controller Access Failure	ETH_E_ACCESS	Assigned by DEM

⌋()

8 API specification

8.1 Imported types

This chapter lists all types included from the following files:

SWS_Eth_00026:

Module	*Imported Type*
ComStack_Types	BufReq_ReturnType
Dem	Dem_EventIdType
	Dem_EventStatusType
Eth_GeneralTypes	Eth_DataType
	Eth_FilterActionType
	Eth_FrameType
	Eth_ModeType
	Eth_ReturnType
	Eth_RxStatusType
	Eth_ConfigType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Type definitions

[SWS_Eth_00148]⌈
Eth.h shall include Eth_GeneralTypes.h for the include of general Eth type declarations.⌋()

[SWS_Eth_00149]⌈
The types specified in SWS_EthernetDriver shall be declared in Eth_GeneralTypes.h.⌋()

8.2.1 Eth_ConfigType

[SWS_Eth_00156]

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

8.2.2 Eth_ReturnType

[SWS_Eth_00157]

Name:	Eth_ReturnType	
Type:	Enumeration	
Range:	ETH_OK	success
	ETH_E_NOT_OK	general failure
	ETH_E_NO_ACCESS	Ethernet hardware access failure
Description:	Ethernet Driver specific return type.	

8.2.3 Eth_ModeType

[SWS_Eth_00158]

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

8.2.4 Eth_StateType

[SWS_Eth_00159]

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.	

8.2.5 Eth_FrameType

[SWS_Eth_00160]

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

8.2.6 Eth_DataType

[SWS_Eth_00161]

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.		

8.2.7 Eth_RxStatusType

[SWS_Eth_00162]

Name:	Eth_RxStatusType	
Type:	Enumeration	
Range:	ETH_RECEIVED	Ethernet frame has been received, no further frames available
	ETH_NOT_RECEIVED	Ethernet frame has not been received, no further frames available
	ETH_RECEIVED_MORE_DATA_AVAILABLE	Ethernet frame has been received, more frames are available
	ETH_RECEIVED_FRAMES_LOST	Ethernet frame has been received, some frames got lost
Description:	Used as out parameter in Eth_Receive() indicates whether a frame has been received and if so, whether more frames are available or frames got lost.	

8.2.8 Eth_FilterActionType

[SWS_Eth_00163]

Name:	Eth_FilterActionType	
Type:	Enumeration	
Range:	ETH_ADD_TO_FILTER	add the MAC address to the filter, meaning allow reception
	ETH_REMOVE_FROM_FILTER	remove the MAC address from the filter, meaning reception is blocked in the lower layer
Description:	The Enumeration Type Eth_FilterActionType describes the action to be taklen for the MAC address given in *PhysAddrPtr.	

8.3 Function definitions

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

8.3.1 Eth_Init

SWS_Eth_00027:

Service name:	Eth_Init	
Syntax:	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	None	

(inout):	
Parameters (out):	None
Return value:	None
Description:	Initializes the Ethernet Driver

SWS_Eth_00028:
The function shall store the access to the configuration structure for subsequent API calls.

SWS_Eth_00029:
The function shall change the state of the component from ETH_STATE_UNINIT to ETH_STATE_INIT.

SWS_Eth_00030:
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.

SWS_Eth_00031:
Caveat: The API has to be called during initialization.

SWS_Eth_00032:
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

SWS_Eth_00033:

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	

SWS_Eth_00034:
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

SWS_Eth_00035:
The function shall change the state of the component from ETH_STATE_INIT to ETH_STATE_ACTIVE.

SWS_Eth_00036:
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.

SWS_Eth_00037:
If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETH_E_INV_CTRL_IDX and return E_NOT_OK.

SWS_Eth_00038:
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.

SWS_Eth_00039:
The function shall check the access to the Ethernet controller. If the check fails, the function shall raise the production error ETH_E_ACCESS and return E_NOT_OK.

SWS_Eth_00040:
Caveat: The function requires previous initialization (Eth_Init).

8.3.3 Eth_SetControllerMode

SWS_Eth_00041:

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: controller mode could not be changed
Description:	Enables / disables the indexed controller	

SWS_Eth_00042:
The function shall:

- Put the controller in the specified mode given in the parameter 'CtrlMode'
  - Upon mode ETH_MODE_DOWN the driver shall:
    - Disable the Ethernet controller
    - Reset all transmit and receive buffers (i.e. ignore all pending transmission and reception requests)
  - Upon mode ETH_MODE_ACTIVE:
    - Enable all transmit and receive buffers
    - Enable the Ethernet controller

SWS_Eth_00043:
If 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.

SWS_Eth_00044:
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.

[SWS_Eth_00168]:
The function shall check the access to the Ethernet controller. If the check fails, the function shall raise the production error ETH_E_ACCESS and return E_NOT_OK.

SWS_Eth_00045:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.4 Eth_GetControllerMode

SWS_Eth_00046:

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

SWS_Eth_00047:
The function shall read the current controller mode.

SWS_Eth_00048:
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.

SWS_Eth_00049:
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.

SWS_Eth_00050:
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.

SWS_Eth_00051:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.5 Eth_GetPhysAddr

SWS_Eth_00052:

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.
Return value:	void	None
Description:	Obtains the physical source address used by the indexed controller	

SWS_Eth_00053:
The function shall read the source address used by the indexed controller.

SWS_Eth_00054:
If 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.

SWS_Eth_00055:
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.

SWS_Eth_00056:
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.

SWS_Eth_00057:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.6 Eth_SetPhysAddr

[SWS_Eth_00151] ⌈

Service name:	Eth_SetPhysAddr	
Syntax:	`void Eth_SetPhysAddr( uint8 CtrlIdx, const uint8* PhysAddrPtr )`	
Service ID[hex]:	0x13	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Driver.
	PhysAddrPtr	Pointer to memory containing the physical source address (MAC address) in network byte order.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Sets the physical source address used by the indexed controller	

⌋()

[SWS_Eth_00139]⌈

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

[SWS_Eth_00140]⌈

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.⌋()

[SWS_Eth_00141]⌈

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.」()

[SWS_Eth_00142]「
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.」()

[SWS_Eth_00143]「
Caveat: The function requires previous controller initialization (Eth_ControllerInit).」()

8.3.7 Eth_UpdatePhysAddrFilter

[SWS_Eth_00152]「

Service name:	Eth_UpdatePhysAddrFilter	
Syntax:	`Std_ReturnType Eth_UpdatePhysAddrFilter( uint8 CtrlIdx, uint8* PhysAddrPtr, Eth_FilterActionType Action )`	
Service ID[hex]:	0x12	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Driver
	PhysAddrPtr	Pointer to memory containing the physical source address (MAC address) in network byte order.
	Action	Add or remove the address from the Ethernet controllers filter.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: filter was successfully changed E_NOT_OK: filter could not be changed
Description:	Update the physical source address to/from the indexed controller filter. If the Ethernet Controller is not capable to do the filtering, the software has to do this.	

」()

[SWS_Eth_00150]「 The function shall update the physical address receive filter of the indexed controller.」 ()

[SWS_Eth_00164]「
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.」()

[SWS_Eth_00165]⌈
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.⌋()

[SWS_Eth_00166]⌈
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.⌋()

[SWS_Eth_00167]⌈
Caveat: The function requires previous controller initialization (Eth_ControllerInit).⌋()

[SWS_Eth_00144]⌈If the physical source address (MAC address) is set to FF:FF:FF:FF:FF:FF, this shall completely open the filter.⌋()

[SWS_Eth_00146]⌈ If this API is used and the hardware does not support filtering, promiscuous mode shall be enabled during initialization.⌋()

[SWS_Eth_00147]⌈ If the physical source address (MAC address) is set to 00:00:00:00:00:00, this shall reduce the filter to the controllers unique unicast MAC address and end promiscuous mode if it was turned on.⌋()

8.3.8 Eth_WriteMii

SWS_Eth_00058:

Service name:	Eth_WriteMii	
Syntax:	`Eth_ReturnType 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 [21] for details)
	RegIdx	Index of the transceiver register on the MII (see [21] for details)
	RegVal	Value to be written into the indexed register (see [21] for details)
Parameters (inout):	None	
Parameters (out):	None	

Return value:	Eth_ReturnType	ETH_OK: MII register written ETH_E_NOT_OK: MII register write failure ETH_E_NO_ACCESS: Ethernet transceiver access failure
Description:	Configures a transceiver register or triggers a function offered by the receiver	

SWS_Eth_00059:
The function shall write the specified transceiver register through the MII of the indexed controller.

SWS_Eth_00060:
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.

SWS_Eth_00061
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.

SWS_Eth_00062
The function shall be pre compile time configurable On/Off by the configuration parameter: EthCtrlEnableMii.

SWS_Eth_00063
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.9 Eth_ReadMii

SWS_Eth_00064:

Service name:	Eth_ReadMii	
Syntax:	`Eth_ReturnType 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 [21] for details)
	RegIdx	Index of the transceiver register on the MII (see [21] for details)
Parameters (inout):	None	
Parameters (out):	RegValPtr	Filled with the register content of the indexed register (see [21] for details)
Return value:	Eth_ReturnType	ETH_OK: MII register read ETH_E_NOT_OK: MII register read failure ETH_E_NO_ACCESS: Ethernet transceiver access failure
Description:	Reads a transceiver register	

SWS_Eth_00065:
The function shall read the specified transceiver register through the MII of the indexed controller.

SWS_Eth_00066:
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.

SWS_Eth_00067:
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.

SWS_Eth_00068:
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.

SWS_Eth_00069:
The function shall be pre compile time configurable On/Off by the configuration parameter: EthCtrlEnableMii.

SWS_Eth_00070:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.10 Eth_GetCounterState

SWS_Eth_00071:

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	

SWS_Eth_00072:

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

SWS_Eth_00073:
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.

SWS_Eth_00074:
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.

SWS_Eth_00075:
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.

SWS_Eth_00076:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.11 Eth_ProvideTxBuffer

SWS_Eth_00077:

Service name:	Eth_ProvideTxBuffer	
Syntax:	`BufReq_ReturnType Eth_ProvideTxBuffer( uint8 CtrlIdx, uint8* BufIdxPtr, uint8** 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 BUFREQ_E_OVFL: requested buffer too large
Description:	Provides access to a transmit buffer of the specified controller	

SWS_Eth_00078:
The function shall provide a transmit buffer resource. The Ethernet Driver shall lock the buffer until it receives a subsequent call of Eth_Transmit service with the buffer index returned in the BufIdxPtr parameter.

SWS_Eth_00137:
All locked transmit buffers shall be released if the controller is disabled via Eth_SetControllerMode.

SWS_Eth_00079:
If a buffer is requested with Eth_ProvideTxBuffer that is larger than the available buffer length, the buffer shall not be locked but return the available length and BUFREQ_E_OVFL.

SWS_Eth_00080:
If all available buffers are in use the component shall return BUFREQ_E_BUSY.

SWS_Eth_00081:
If 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.

SWS_Eth_00082:
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.

SWS_Eth_00083:
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.

SWS_Eth_00084:
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.

SWS_Eth_00085:
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.

SWS_Eth_00086:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.12 Eth_Transmit

SWS_Eth_00087:

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	

SWS_Eth_00088:
The function shall build the Ethernet header with the given physical target address (MAC address) and trigger the transmission of a previously filled transmit buffer.

SWS_Eth_00089:
If TxConfirmation is false, the function shall release the buffer resource.

SWS_Eth_00138:
All pending transmit buffers shall be released if the controller is disabled via Eth_SetControllerMode.

SWS_Eth_00090:
If 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.

SWS_Eth_00091:
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.

SWS_Eth_00092:
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.

SWS_Eth_00093:
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.

SWS_Eth_00129:

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.

SWS_Eth_00094:
Caveat: The function requires previous buffer request (Eth_ProvideTxBuffer).

8.3.13 Eth_Receive

SWS_Eth_00095:

Service name:	Eth_Receive	
Syntax:	`void Eth_Receive( uint8 CtrlIdx, Eth_RxStatusType* RxStatusPtr )`	
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):	RxStatusPtr	Indicates whether a frame has been received and if so, whether more frames are available or frames got lost.
Return value:	None	
Description:	Triggers frame reception	

SWS_Eth_00096:
The function shall read the next frame from the receive buffers. The function passes the received frame to the Ethernet interface using the callback function EthIf_RxIndication and indicates if there are more frames in the receive buffers.

SWS_Eth_00097:
If 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.

SWS_Eth_00098:
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.

SWS_Eth_00132:
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.

[SWS_Eth_00155]

If development error detection is enabled: the function shall check for lost frames. If frames got lost, the function shall raise the development error ETH_E_FRAMES_LOST.

[SWS_Eth_00153]
When calling the callback function EthIf_RxIndication broadcast frames shall be indicated to the Ethernet Interface (see [6]).

SWS_Eth_00099:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

8.3.14 Eth_TxConfirmation

SWS_Eth_00100:

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

SWS_Eth_00101:
The function shall check all filled transmit buffers for successful transmission. The function issues transmit confirmation for each transmitted frame using the callback function EthIf_TxConfirmation if requested by the previous call of Eth_Transmit service.

SWS_Eth_00102:
If transmission confirmation was enabled by a previous call to Eth_Transmit function the function shall release the buffer resource.

SWS_Eth_00103:
If 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.

SWS_Eth_00104:
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.

SWS_Eth_00134:

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.

SWS_Eth_00105:
Caveat: The function requires previous initialization (Eth_ControllerInit).

8.3.15 Eth_GetVersionInfo

SWS_Eth_00106:

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	

SWS_Eth_00136:
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>

SWS_Eth_00109:

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	

SWS_Eth_00110:
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_RxIndication.

SWS_Eth_00111:
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.

SWS_Eth_00112:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

SWS_Eth_00113:
Caveat: The function shall be callable on interrupt level.

8.5.2 Eth_TxIrqHdlr_<CtrlIdx>

SWS_Eth_00114:

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	

SWS_Eth_00115:
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_TxConfirmation if requested by the previous call of Eth_Transmit service.

SWS_Eth_00116:

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.

SWS_Eth_00117:
Caveat: The function requires previous controller initialization (Eth_ControllerInit).

SWS_Eth_00118:
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.

SWS_Eth_00119:

API function	*Description*
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. OBD Events Suppression shall be ignored for this computation.
EthIf_RxIndication	Handles a received frame received by the indexed controller
EthIf_TxConfirmation	Confirms frame transmission by the indexed controller
SchM_Enter_Eth	Invokes the SchM_Enter function to enter a module local exclusive area.
SchM_Exit_Eth	Invokes the SchM_Exit function to exit an exclusive area.

8.7.2 Optional Interfaces

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

SWS_Eth_00120:

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

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 Containers and configuration parameters

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

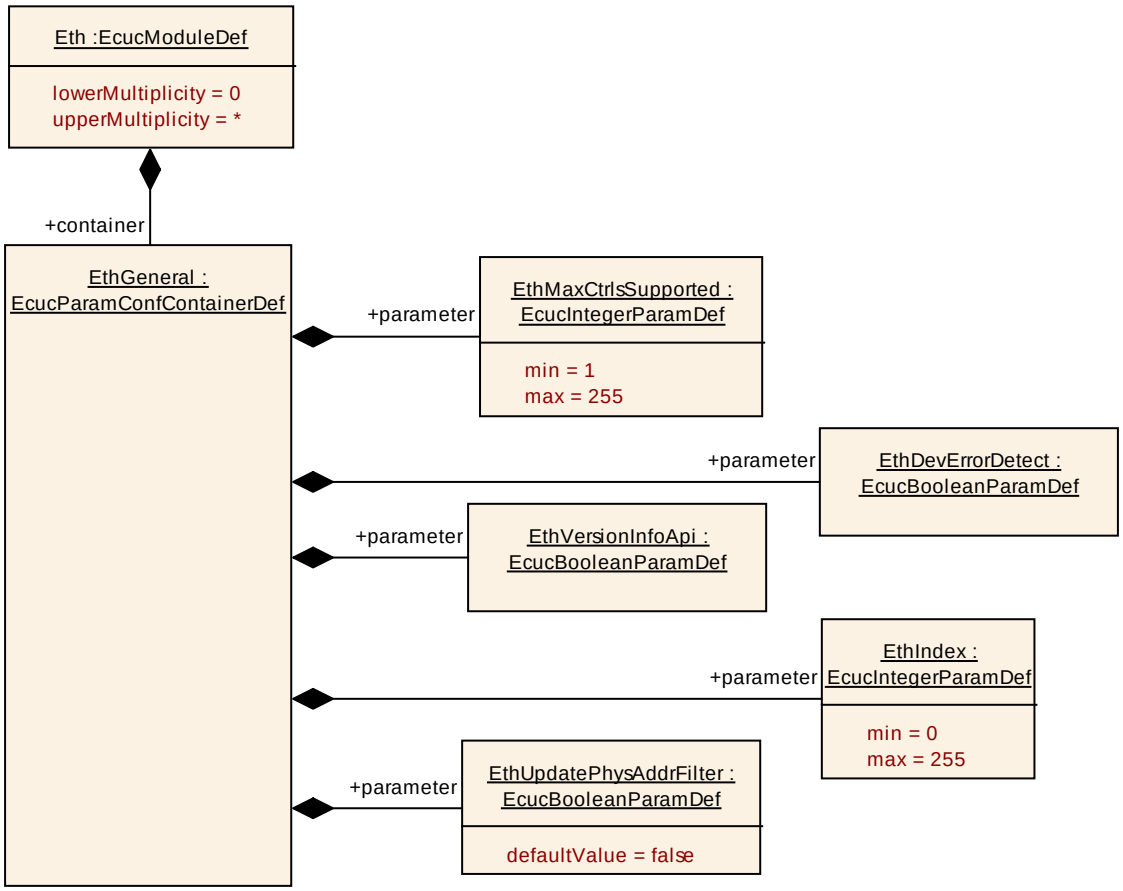

Figure 10.1: Ethernet Driver configuration structure

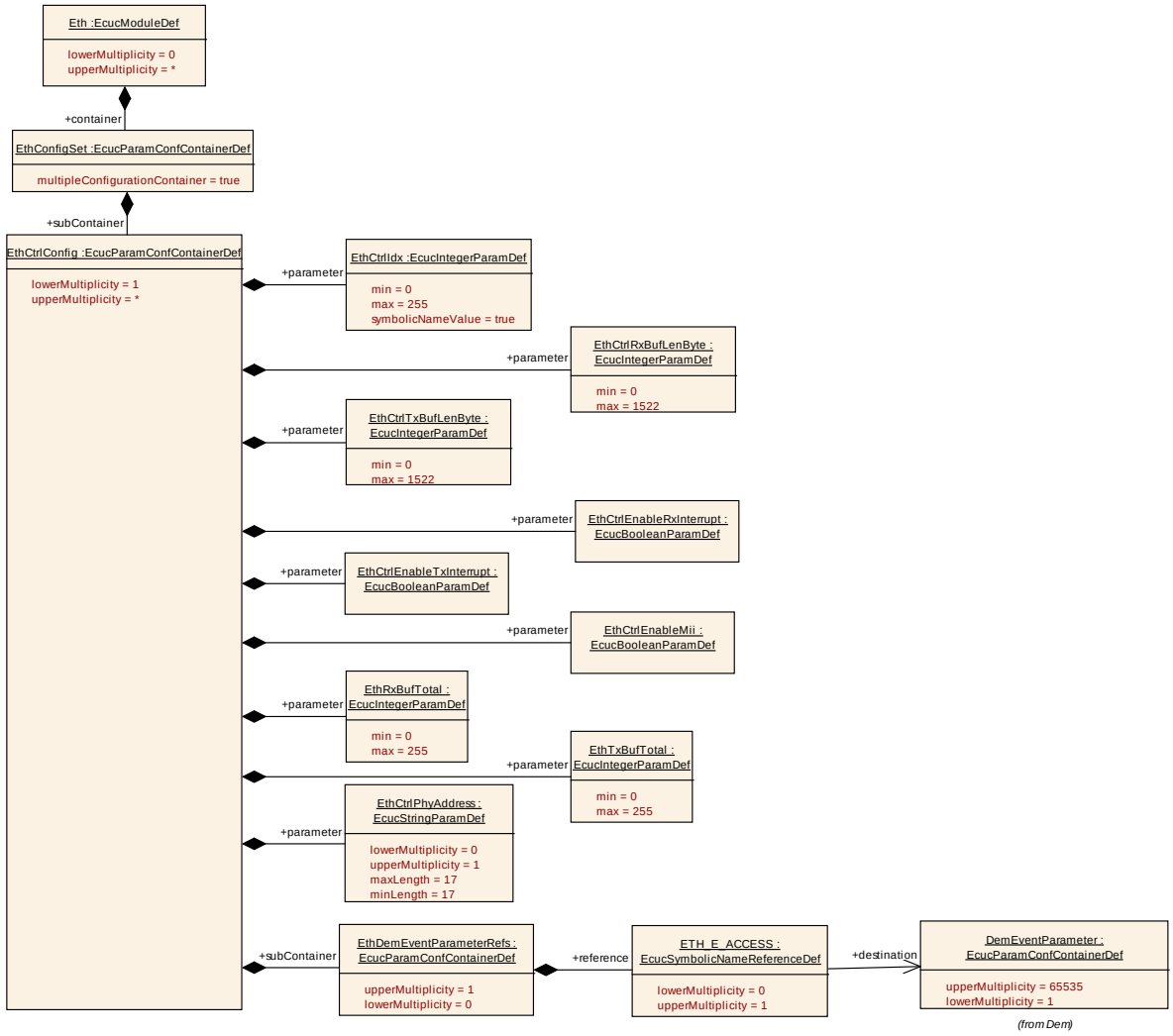

Figure 10.2: Ethernet Driver Controller configuration structure

10.1.1 Variants

SWS_Eth_00121:
VARIANT-POST-BUILD: All configuration parameters in container ‘EthGeneral’ shall be configurable at pre-compile time.
Use case: Object code delivery, selectable configuration

SWS_Eth_00122:
VARIANT-LINK-TIME: All configuration parameters in container ‘EthGeneral’ shall be configurable at pre-compile time.
Use case: Object code delivery, single configuration

SWS_Eth_00123:
VARIANT-PRE-COMPILE: All configuration parameters shall be configurable at pre-compile time.
Use case: Execution time optimizations, fix configuration

10.1.2 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

10.1.3 EthConfigSet

SWS Item	**ECUC_Eth_00015 :**
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

10.1.4 EthCtrlConfig

SWS Item	**ECUC_Eth_00006 :**
Container Name	EthCtrlConfig
Description	Configuration of the individual controller
Configuration Parameters	

SWS Item	**ECUC_Eth_00012 :**		
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: local		

SWS Item	**ECUC_Eth_00010 :**		
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: local		

SWS Item	**ECUC_Eth_00011 :**		
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: local		

SWS Item	**ECUC_Eth_00007 :**		
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: ECU		

SWS Item	**ECUC_Eth_00020 :**		
Name	EthCtrlPhyAddress		
Description	Specifies the unique 48-bit physical address (MAC address) of the controller in network byte order. Regular Expression: [0-9a-fA-F]{2}[[:-][0-9a-fA-F]{2}]{5}		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	17		
minLength	17		
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: local		

SWS Item	**ECUC_Eth_00008 :**		
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: local		

SWS Item	**ECUC_Eth_00009 :**		

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: local		

SWS Item	**ECUC_Eth_00013 :**		
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: local		

SWS Item	**ECUC_Eth_00014 :**		
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: local		

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.

10.1.5 EthDemEventParameterRefs

SWS Item	**ECUC_Eth_00016 :**
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	ECUC_Eth_00017 :		
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	Symbolic name 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: local		

No Included Containers

10.1.6 EthGeneral

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

SWS Item	ECUC_Eth_00003 :		
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: local		

SWS Item	ECUC_Eth_00018 :		
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: local		

SWS Item	ECUC_Eth_00002 :	
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: local		

SWS Item	**ECUC_Eth_00019 :**		
Name	EthUpdatePhysAddrFilter		
Description	Enables/Disables optional API Eth_UpdatePhysAddrFilter.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Eth_00004 :**		
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: local		

No Included Containers