Document Title	Specification of Ethernet Switch Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	656

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R21-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2021-11-25	R21-11	AUTOSAR Release Management	• Added 10BASE-T1S support • Clarified return values and development errors • Removed EthSwtMgmtInfoIndicationTimeout related requirements
2020-11-30	R20-11	AUTOSAR Release Management	• Support for Ethernet wake on data line • Modified description of return values in EthSwt_StoreConfiguration, EthSwt_ResetConfiguration • Updated the types for ReTaggingVlanId and DoubleTaggingVlanId • Fix service IDs for EthSwt_DeletePortMirrorConfiguration, EthSwt_PortLinkStateRequest, EthSwt_GetMaxFIFOBufferFillLevel • Editorial changes

2019-11-28	R19-11	AUTOSAR Release Management	• Possibility to explicitly request or release Ethernet link state added • Replace usage of EthTrcv_ModeType with the Eth_ModeType • Support for 2500 MBit/s Ethernet connection • Fix Ethernet Hardware Initialization • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Clarified Port Mirroring concepts. • Introduced timeout for ARL table entries • Added counter synchronization for cascaded switches
2017-12-08	4.3.1	AUTOSAR Release Management	• minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	• Restructured VLAN-membership as a port-related configuration parameter • Introduced configuration of rate policers on ingress side • Introduced filter configuration for double tagged frames • Introduced configuration of minimum buffer size for FIFOS • Introduced Types to read HW-statistic by List pointer; reorganized interfaces to read HW-statistics. • Introduced Compensation of Ethernet switch delays for Global Time Synchronization • Add / update elements to describe MAC interface and physical interface • Added testing functionality for diagnostic use cases • Added Possibility to switch off ports and switch instances according to VLAN or PNC. • Introduced interfaces for verification of switch configuration

2015-07-31	4.2.2	AUTOSAR Release Management	• minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2014-10-31	4.2.1	AUTOSAR Release Management	• Initial Release

Disclaimer

Table of Contents

1 Introduction and functional overview

In the AUTOSAR Layered Software Architecture [1], the Ethernet Switch Driver belongs to the Communication Hardware Abstraction.

This indicates the main task of the Ethernet Switch Driver:

Provide to the upper layers (e.g. Ethernet Interface [2]) a hardware independent interface comprising a switch with several ports. This interface shall be uniform for all Ethernet switches. Thus, the upper layers may access the underlying communication technology in a uniform manner.

A single Ethernet Switch Driver module supports only one type of switch hardware. The Ethernet physical layer ports are configured by the Ethernet Transceiver Driver[3]. The Ethernet Switch Driver's prefix generates a unique namespace. The Ethernet Interface can access different Ethernet controller types using different Ethernet Switch Drivers using this prefix. The decision which driver to use to access a particular transceiver is a configuration parameter of the Ethernet Interface.

Figure 1.1 depicts the lower part of the Ethernet stack. Accesses via an SPI- and MII/MDIO-Hardware-Interface for switch specific configuration or functions are directly done via the Ethernet Driver [4] or the SPI driver [5].

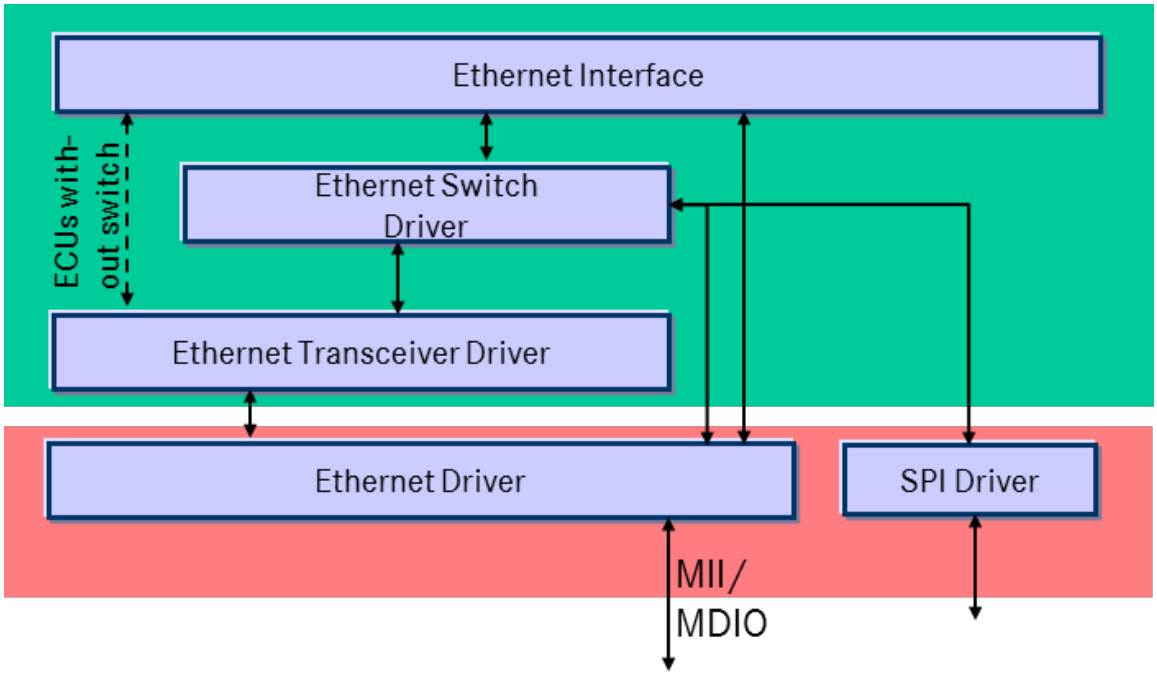

Figure 1.1: Ethernet Switch Driver in layer architecture

2 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations and terms relevant to the Network Management Interface module that are not included in the [6, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
DEM	Diagnostic Event Manager module
EcuM	ECU State Manager module
Eth	Ethernet Controller Driver (AUTOSAR BSW module)
EthIf	Ethernet Interface (AUTOSAR BSW module)
EthTrcv	Ethernet Transceiver Driver (AUTOSAR BSW module)
MII	Media Independent Interface (standardized interface provided by Ethernet controllers to access Ethernet transceivers)
MDIO	Management Data Input/Output
OA TC10	Open Alliance TC10 specification (see [7])

3 Related documentation

3.1 Input documents & related standards and norms

[1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture

[2] Specification of Ethernet Interface
AUTOSAR_SWS_EthernetInterface

[3] Specification of Ethernet Transceiver Driver
AUTOSAR_SWS_EthernetTransceiverDriver

[4] Specification of Ethernet Driver
AUTOSAR_SWS_EthernetDriver

[5] Specification of SPI Handler/Driver
AUTOSAR_SWS_SPIHandlerDriver

[6] Glossary
AUTOSAR_TR_Glossary

[7] OPEN Sleep/Wake-up Specification for Automotive Ethernet
http://www.opensig.org/Automotive-Ethernet-Specifications/

[8] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral

[9] Requirements on Ethernet Support in AUTOSAR
AUTOSAR_SRS_Ethernet

[10] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral

[11] IEEE 802.1Q-2011 - IEEE Standard for Local and metropolitan area networks - Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks

[12] Specification of Time Synchronization over Ethernet
AUTOSAR_SWS_TimeSyncOverEthernet

[13] Specification of NVRAM Manager
AUTOSAR_SWS_NVRAMManager

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software [8, SWS_BSWGeneral] which is also valid for Ethernet Switch Driver.

Thus, the specifications [SWS_BSWGeneral] [8], SRS_Ethernet [9] shall be considered as additional and required specification for Ethernet Switch Driver.

4 Constraints and assumptions

4.1 Limitations

The Ethernet Switch 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/s, 100MBit/s and 1000Mbit/s Ethernet and transceivers connected via (gigabit) Media Independent Interface (xMII).

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

The switch driver does not support the following features:

- MAC-based Ingress Filtering: No filtering options for Ethernet frames based on MAC-addresses is supported.

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 Switch Driver module.

Modules that use the Ethernet Switch Driver module:

- Ethernet Interface (EthIf) calls the Ethernet Switch driver for initializing and accessing the switch device.

Modules used by the Ethernet Switch Driver module:

- Ethernet Controller Driver (Eth) for transceiver access via Media Independent Interface (MII).
- Ethernet Transceiver Driver (EthTrcv) for configuring the PHY ports and controlling/checking the ports.
- The configuration of the Ethernet Switch device can be either via MDIO or SPI. In case of an SPI interface access to SPI module is necessary.

Dependencies to other Modules:

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

6 Requirements Tracing

The following tables reference the requirements specified in [9] as well as [10] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00003]	All software modules shall provide version and identification information	[SWS_EthSwt_00131]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_EthSwt_00006] [SWS_EthSwt_00007] [SWS_EthSwt_00008] [SWS_EthSwt_00011]
[SRS_BSW_00161]	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	[SWS_EthSwt_00099] [SWS_EthSwt_00130]
[SRS_BSW_00162]	The AUTOSAR Basic Software shall provide a hardware abstraction layer	[SWS_EthSwt_00099] [SWS_EthSwt_00130]

Requirement	Description	Satisfied by
[SRS_BSW_00171]	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	[SWS_EthSwt_00022] [SWS_EthSwt_00029] [SWS_EthSwt_00035] [SWS_EthSwt_00042] [SWS_EthSwt_00049] [SWS_EthSwt_00056] [SWS_EthSwt_00058] [SWS_EthSwt_00090] [SWS_EthSwt_00095] [SWS_EthSwt_00124] [SWS_EthSwt_00129] [SWS_EthSwt_00177] [SWS_EthSwt_00186] [SWS_EthSwt_00191] [SWS_EthSwt_00202] [SWS_EthSwt_00210] [SWS_EthSwt_00215] [SWS_EthSwt_00220] [SWS_EthSwt_00225] [SWS_EthSwt_00229] [SWS_EthSwt_00230] [SWS_EthSwt_00240] [SWS_EthSwt_00243] [SWS_EthSwt_00249] [SWS_EthSwt_00253] [SWS_EthSwt_00257] [SWS_EthSwt_00261] [SWS_EthSwt_00264] [SWS_EthSwt_00268] [SWS_EthSwt_00273] [SWS_EthSwt_00287] [SWS_EthSwt_00291] [SWS_EthSwt_00297] [SWS_EthSwt_00303] [SWS_EthSwt_00308] [SWS_EthSwt_00312] [SWS_EthSwt_00317] [SWS_EthSwt_00322] [SWS_EthSwt_00327] [SWS_EthSwt_00332] [SWS_EthSwt_00338] [SWS_EthSwt_00344] [SWS_EthSwt_00350] [SWS_EthSwt_00362] [SWS_EthSwt_00370] [SWS_EthSwt_00379] [SWS_EthSwt_00403] [SWS_EthSwt_00405] [SWS_EthSwt_00427] [SWS_EthSwt_00432] [SWS_EthSwt_00441] [SWS_EthSwt_00443]
[SRS_BSW_00347]	A Naming seperation of different instances of BSW drivers shall be in place	[SWS_EthSwt_00131]

Requirement	Description	Satisfied by
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/disabling of detection and reporting of development errors.	[SWS_EthSwt_00386] [SWS_EthSwt_00387] [SWS_EthSwt_00389] [SWS_EthSwt_00390] [SWS_EthSwt_00391] [SWS_EthSwt_00392] [SWS_EthSwt_00393]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_EthSwt_00128] [SWS_EthSwt_00164]
[SRS_BSW_00375]	Basic Software Modules shall report wake-up reasons	[SWS_EthSwt_00098]
[SRS_BSW_00385]	List possible error notifications	[SWS_EthSwt_00001] [SWS_EthSwt_00113] [SWS_EthSwt_00395]
[SRS_BSW_00386]	The BSW shall specify the configuration for detecting an error	[SWS_EthSwt_00016] [SWS_EthSwt_00164]
[SRS_BSW_00395]	The Basic Software Module specifications shall list all configuration parameter dependencies	[SWS_EthSwt_00165]
[SRS_BSW_00406]	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	[SWS_EthSwt_00123]
[SRS_BSW_00413]	An index-based accessing of the instances of BSW modules shall be done	[SWS_EthSwt_00120] [SWS_EthSwt_00154] [SWS_EthSwt_00156] [SWS_EthSwt_00157] [SWS_EthSwt_00180]
[SRS_BSW_00433]	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	[SWS_EthSwt_00114] [SWS_EthSwt_00115]
[SRS_BSW_323]	No description	[SWS_EthSwt_00009] [SWS_EthSwt_00154] [SWS_EthSwt_00156] [SWS_EthSwt_00157] [SWS_EthSwt_00180]
[SRS_BSW_369]	No description	[SWS_EthSwt_00009] [SWS_EthSwt_00154] [SWS_EthSwt_00156] [SWS_EthSwt_00157] [SWS_EthSwt_00180]

Requirement	Description	Satisfied by
[SRS_ETH_00087]	Semi-Static Auto-Configuration	[SWS_EthSwt_00031] [SWS_EthSwt_00032] [SWS_EthSwt_00060] [SWS_EthSwt_00061] [SWS_EthSwt_00086] [SWS_EthSwt_00087] [SWS_EthSwt_00091] [SWS_EthSwt_00092] [SWS_EthSwt_00098] [SWS_EthSwt_00111] [SWS_EthSwt_00117] [SWS_EthSwt_00118] [SWS_EthSwt_00125] [SWS_EthSwt_00126] [SWS_EthSwt_00127] [SWS_EthSwt_00136] [SWS_EthSwt_00162] [SWS_EthSwt_00181] [SWS_EthSwt_00182] [SWS_EthSwt_00183] [SWS_EthSwt_00187] [SWS_EthSwt_00188] [SWS_EthSwt_00193] [SWS_EthSwt_00194] [SWS_EthSwt_00196] [SWS_EthSwt_00197] [SWS_EthSwt_00203] [SWS_EthSwt_00204] [SWS_EthSwt_00226] [SWS_EthSwt_00227] [SWS_EthSwt_00228] [SWS_EthSwt_00235]
[SRS_ETH_00114]	Ethernet Switch Filtering and Policing	[SWS_EthSwt_00134] [SWS_EthSwt_00172] [SWS_EthSwt_00173]
[SRS_ETH_00118]	Transparent interface to underlying EthTrcv module(s)	[SWS_EthSwt_00018] [SWS_EthSwt_00019] [SWS_EthSwt_00023] [SWS_EthSwt_00025] [SWS_EthSwt_00026] [SWS_EthSwt_00038] [SWS_EthSwt_00044] [SWS_EthSwt_00045] [SWS_EthSwt_00051] [SWS_EthSwt_00052] [SWS_EthSwt_00098] [SWS_EthSwt_00154] [SWS_EthSwt_00156] [SWS_EthSwt_00157] [SWS_EthSwt_00164] [SWS_EthSwt_00217] [SWS_EthSwt_00222] [SWS_EthSwt_00398] [SWS_EthSwt_00440]

Requirement	Description	Satisfied by
[SRS_ETH_00119]	Access to hardware status of ports	[SWS_EthSwt_00037] [SWS_EthSwt_00038] [SWS_EthSwt_00098] [SWS_EthSwt_00117] [SWS_EthSwt_00118] [SWS_EthSwt_00154] [SWS_EthSwt_00203] [SWS_EthSwt_00204] [SWS_EthSwt_00430] [SWS_EthSwt_00431]
[SRS_ETH_00120]	Hardware access via MII and/or SPI	[SWS_EthSwt_00098] [SWS_EthSwt_00217] [SWS_EthSwt_00222]
[SRS_ETH_00121]	Configuration of forwarding rules	[SWS_EthSwt_00132] [SWS_EthSwt_00133] [SWS_EthSwt_00134] [SWS_EthSwt_00135] [SWS_EthSwt_00172] [SWS_EthSwt_00173] [SWS_EthSwt_00178] [SWS_EthSwt_00179] [SWS_EthSwt_00234]
[SRS_ETH_00122]	Persistent storage of configurations	[SWS_EthSwt_00086] [SWS_EthSwt_00087] [SWS_EthSwt_00091] [SWS_EthSwt_00092] [SWS_EthSwt_00125] [SWS_EthSwt_00126] [SWS_EthSwt_00127] [SWS_EthSwt_00136] [SWS_EthSwt_00182] [SWS_EthSwt_00183] [SWS_EthSwt_00193] [SWS_EthSwt_00194] [SWS_EthSwt_00196]
[SRS_ETH_00123]	Testing and diagnostic of switch ports	[SWS_EthSwt_00416] [SWS_EthSwt_00417] [SWS_EthSwt_00418] [SWS_EthSwt_00419] [SWS_EthSwt_00420] [SWS_EthSwt_91017] [SWS_EthSwt_91020]
[SRS_ETH_00125]	The Ethernet Switch Driver shall support switch frame management	[SWS_EthSwt_00098] [SWS_EthSwt_00240] [SWS_EthSwt_00241] [SWS_EthSwt_00242] [SWS_EthSwt_00243] [SWS_EthSwt_00245] [SWS_EthSwt_00378]
[SRS_ETH_00126]	Independent reset of host ECU and switch hardware	[SWS_EthSwt_00181] [SWS_EthSwt_91012] [SWS_EthSwt_91013]

Requirement	Description	Satisfied by
[SRS_ETH_00128]	The Ethernet Switch Driver shall provide statistic counter values per port	[SWS_EthSwt_00106] [SWS_EthSwt_00199] [SWS_EthSwt_00372] [SWS_EthSwt_00373]
[SRS_ETH_00458]	No description	[SWS_EthSwt_00128]
[SRS_Eth_00107]	The Ethernet Transceiver Driver shall support access to the wake up reason.	[SWS_EthSwt_00442] [SWS_EthSwt_91040]
[SRS_Eth_00114]	Ethernet Switch Filtering and Policing	[SWS_EthSwt_00233]
[SRS_Eth_00118]	Transparent interface to underlying EthTrcv module(s)	[SWS_EthSwt_91003]
[SRS_Eth_00120]	Hardware access via MII and/or SPI	[SWS_EthSwt_00206] [SWS_EthSwt_00207] [SWS_EthSwt_00211] [SWS_EthSwt_00212] [SWS_EthSwt_00216] [SWS_EthSwt_00221]
[SRS_Eth_00122]	Persistent storage of configurations	[SWS_EthSwt_00098] [SWS_EthSwt_00192]
[SRS_Eth_00123]	Testing and diagnostic of switch ports	[SWS_EthSwt_00293] [SWS_EthSwt_00299] [SWS_EthSwt_00305] [SWS_EthSwt_00309] [SWS_EthSwt_00313] [SWS_EthSwt_00318] [SWS_EthSwt_00323] [SWS_EthSwt_00328] [SWS_EthSwt_00334] [SWS_EthSwt_00340] [SWS_EthSwt_00346] [SWS_EthSwt_00421] [SWS_EthSwt_00422] [SWS_EthSwt_00424] [SWS_EthSwt_00425] [SWS_EthSwt_00426] [SWS_EthSwt_91014] [SWS_EthSwt_91015] [SWS_EthSwt_91016] [SWS_EthSwt_91018] [SWS_EthSwt_91019] [SWS_EthSwt_91021] [SWS_EthSwt_91022] [SWS_EthSwt_91023] [SWS_EthSwt_91024] [SWS_EthSwt_91025] [SWS_EthSwt_91029] [SWS_EthSwt_91030] [SWS_EthSwt_91031] [SWS_EthSwt_91032]

Requirement	Description	Satisfied by
[SRS_Eth_00125]	The Ethernet Switch Driver shall support switch frame management	[SWS_EthSwt_91002] [SWS_EthSwt_91004] [SWS_EthSwt_91005] [SWS_EthSwt_91006] [SWS_EthSwt_91007] [SWS_EthSwt_91008] [SWS_EthSwt_91009] [SWS_EthSwt_91010] [SWS_EthSwt_91028]
[SRS_Eth_00126]	Independent reset of host ECU and switch hardware	[SWS_EthSwt_00292]
[SRS_Eth_00128]	The Ethernet Switch Driver shall provide statistic counter values per port	[SWS_EthSwt_00198] [SWS_EthSwt_00231] [SWS_EthSwt_91000] [SWS_EthSwt_91001]

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 Ethernet BSW stack. The EthIf module accesses several switches using one or more Ethernet Switch Driver modules. The role of the Ethernet transceiver driver is to configure and control the physical layer ports (PHY) integrated into or connected to a switch. Whereas, the role of the Ethernet switch driver is the configuration and control of the switch. In case the Ethernet interface wants to access a PHY, it has to use the APIs of the switch driver which forward the API call to the addressed transceiver driver.

By separating the transceiver driver from the switch driver, different hardware architectures will be supported. In HW-Variant 1, the PHYs are separate devices from different vendors. They are connected via MII and MDIO to a switch which is integrated in to a μC. In HW-Variant 2, the switch has integrated PHYs. In HW-Variant 3, the μC can control the switch via MDIO or SPI and the switch has three external PHYs which can be controlled via MDIO. In this case, different Ethernet transceiver drivers might occur.

Please note that the functional behavior of the ingress and egress port of a switch is implemented in hardware in the switch devices (see [11]). Thus, the configuration from chapter 10 in some parts has to be written to the switch device.

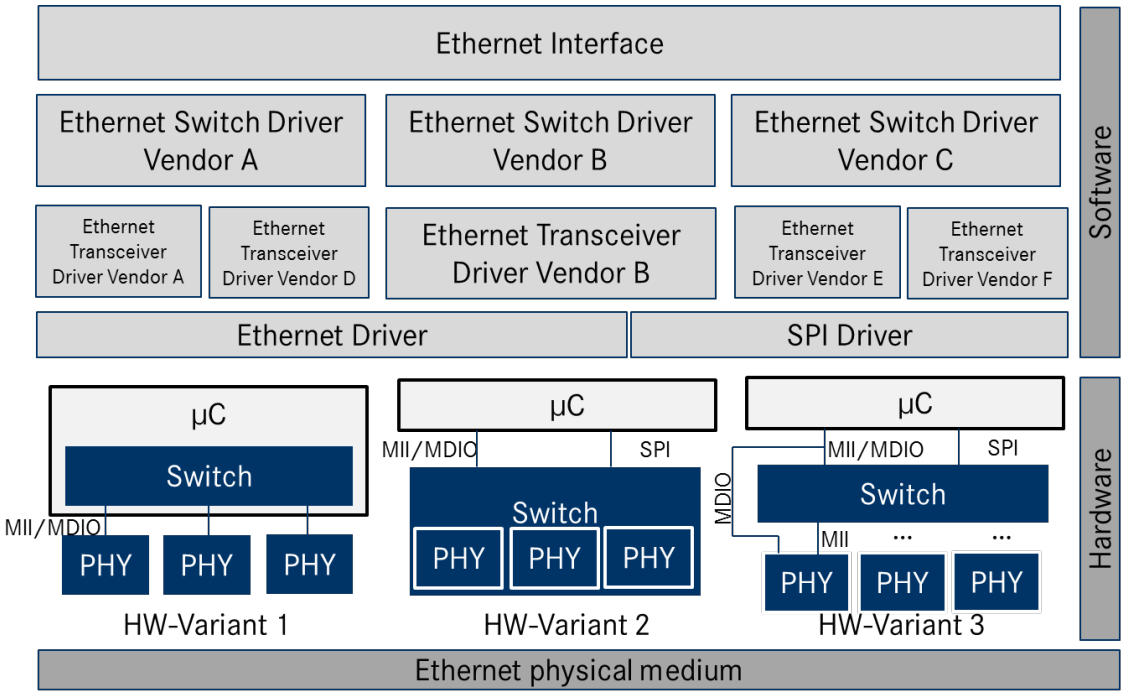

Figure 7.1: Basic Structure of the Ethernet BSW stack.(Note: The different hardware variants are alternative setups)

7.1.1 Indexing scheme

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

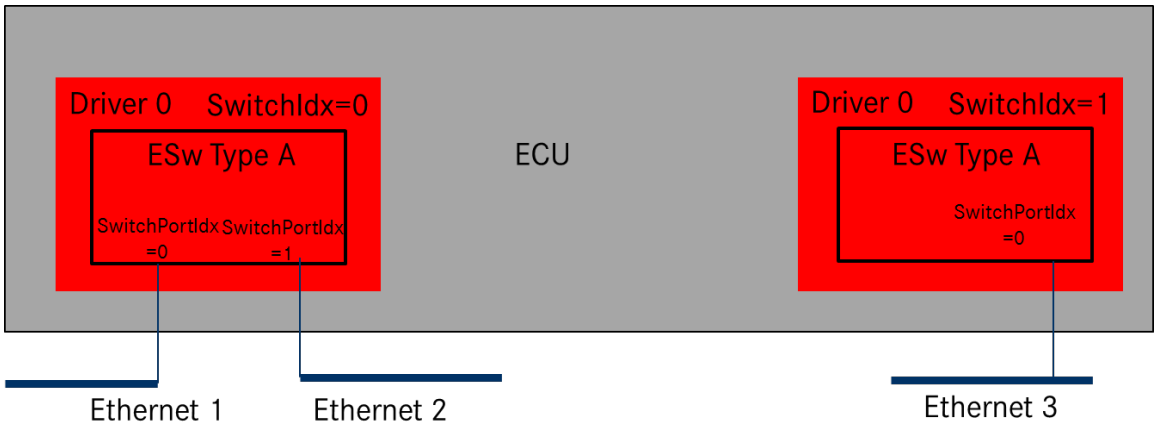

Figure 7.2: Ethernet Switch Driver indexing scheme

[SWS_EthSwt_00099] ⌈The Ethernet Switch Driver shall use a zero-based index to abstract the access for upper software layers.⌋ *(SRS_BSW_00161, SRS_BSW_00162)*

[SWS_EthSwt_00130] ⌈The `SwitchPortIdx` is an index for a port at the switch.⌋ *(SRS_BSW_00161, SRS_BSW_00162)*

[SWS_EthSwt_00120] ⌈The parameter `EthSwtIdx` within the configuration shall correspond to the argument used in the API.⌋*(SRS_BSW_00413)*

[SWS_EthSwt_00180] ⌈The parameter `EthSwtIndex` shall be used to distinguish different instances of a switch driver module in case the API `Det_ReportError (uint16 ModuleId, uint8 InstanceId, uint8 ApiId, uint8 ErrorId)` is called.⌋*(SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)*

[SWS_EthSwt_00131] ⌈In case different Switch devices are used in one ECU, the function names of the different Ethernet Switch drivers must be modified such that no two functions with the same names are generated. It is the responsibility of the user to take care that no two functions with the same names are configured. The names may be extended with a vendor ID or a type ID.⌋*(SRS_BSW_00003, SRS_BSW_00347)*

7.1.2 Ethernet Switch Port Mirroring

Ethernet switch port mirroring use the common established functionality of the Ethernet switch hardware to mirror traffic of one or more Ethernet switch ports (mirrored port) to a another Ethernet switch port (capture port). The mirroring configuration is given by the port mirror configuration (see [SWS_EthSwt_91017]). The port mirror configuration is set up per Ethernet switch. The configuration is stored persistently by the Ethernet switch driver. Therefore a shadow buffer is used to store the port mirror configuration during runtime and stored persistently according to the NvM storing strategy (e.g. store the shadow buffer persistently upon ECU shutdown). The port mirror configuration could be activated and de-activated, respectively, explicitly via dedicated APIs. The port mirroring is controlled by a dedicated diagnostic CDD with receive diagnostic request and forward them to the Ethernet switch driver.

[SWS_EthSwt_00416] ⌈The port mirror configuration (see [SWS_EthSwt_91017]) shall be written to a shadow buffer of the Ethernet switch driver per Ethernet Switch by calling `EthSwt_WritePortMirrorConfiguration`.⌋*(SRS_ETH_00123)*

Note: One port mirror configuration is maintained per Ethernet switch.

[SWS_EthSwt_00417] ⌈The port mirror configuration shall be enabled and disabled, respectively, per Ethernet Switch by calling `EthSwt_SetPortMirrorState`. The current state of the stored port mirror configuration shall be stored persistently, to outlast an ECU reset and to restore the port mirroring activities after an ECU reset.⌋ *(SRS_ETH_00123)*

[SWS_EthSwt_00418] ⌈The stored port mirror configuration shall be marked as "to be deleted" by calling `EthSwt_DeletePortMirrorConfiguration`, if the port mirroring of the given Ethernet switch index is disabled (see [SWS_EthSwt_91022]. Otherwise the request to delete the port mirror configuration shall be rejected.⌋*(SRS_ETH_00123)*

Note: The shadow buffer is stored persistently according to the NvM storing strategy, e.g. store the shadow buffer persistently upon ECU shutdown.

[SWS_EthSwt_00419] ⌈The current port mirroring state shall be returned by calling EthSwt_GetPortMirrorState.⌋*(SRS_ETH_00123)*

[SWS_EthSwt_00420] ⌈The port mirror configuration per Ethernet switch shall be returned by calling EthSwt_ReadPortMirrorConfiguration.⌋*(SRS_ETH_00123)*

7.1.3 State Handling

[SWS_EthSwt_00435] ⌈All functions apart from EthSwt_SetSwitchPortMode, EthSwt_GetSwitchPortMode, EthSwt_StartSwitchPortAutoNegotiation, EthSwt_GetLinkState, EthSwt_GetBaudRate, EthSwt_GetDuplexMode, EthSwt_ReadTrcvRegister, EthSwt_WriteTrcvRegister, EthSwt_Init, EthSwt_MainFunction and EthSwt_BackgroundTask may only be called in state ETHSWT_STATE_ACTIVE.

If a function which can only run (succeed with E_OK) in the states ETHSWT_STATE_PORTINIT_COMPLETED and ETHSWT_STATE_ACTIVE is called before state ETHSWT_STATE_PORTINIT_COMPLETED is reached, the Ethernet switch driver shall raise the runtime error ETHSWT_INIT_NOT_COMPLETED.⌋*()*

[SWS_EthSwt_00436] ⌈ETHSWT_STATE_PORTINIT_COMPLETED shall be reached as soon as the port initialization has finished.⌋*()*

Note: ETHSWT_STATE_PORTINIT_COMPLETED can be reached either by the function EthSwt_Init or by a background task (see [SWS_EthSwt_91104]).

[SWS_EthSwt_00437] ⌈ETHSWT_STATE_ACTIVE shall be reached, when the Ethernet switch initialization has finished.⌋*()*

Note: The initialization of the Ethernet switch takes longer than the initialization of the Ethernet switch ports.

7.1.4 Handling of cable diagnostic

Cable diagnostic measurement is triggered by calling EthSwt_RunPortCableDiagnostic. The current state of the cable diagnostic measurement is polled by calling EthSwt_GetPortCableDiagnosticsResult. If EthSwt_GetPortCableDiagnosticsResult return with other value then ETHTRCV_CABLEDIAG_PENDING, then the cable diagnostic has finished.

Its up to the caller to re-trigger cable diagnostic again, if the measurement failed by returning ETHTRCV_CABLEDIAG_ERROR.

[SWS_EthSwt_00428] ⌈The cable diagnostic APIs (EthSwt_RunPortCableDiagnostic, EthSwt_GetPortCableDiagnosticsResult) shall only be called for Ethernet switch ports of a Ethernet switch, where the Ethernet switch ports reference an Ethernet transceiver.⌋*()*

Note: The upper layer is a CDD that triggers the cable diagnostic measurement and maintains the cable diagnostic result. The EthSwt forwards the API calls to the EthTrcv (see [SWS_EthSwt_00429] and [SWS_EthSwt_00346]).

7.1.5 Functional Description

7.1.5.1 Learning Phase at Start-up

[SWS_EthSwt_00226] ⌈The switch driver shall support a learning phase which can be divided into several sequential steps.⌋ *(SRS_ETH_00087)*

Note: After assembly and initial power-up of the network, three learning phases follow which include MAC-Learning and IP-Address Assignment. Afterwards the learned parameters are stored to one or several non-volatile memories to make them available for subsequent start-ups. This process is shown in Figure 7.3. As an example for triggering this process, the DCM receives a diagnostic request via a bus system or a broadcast message in the Ethernet network. This diagnostic request can be forwarded to an SWC or CCD which triggers the auto-configuration process. However, the trigger is not part of this specification.

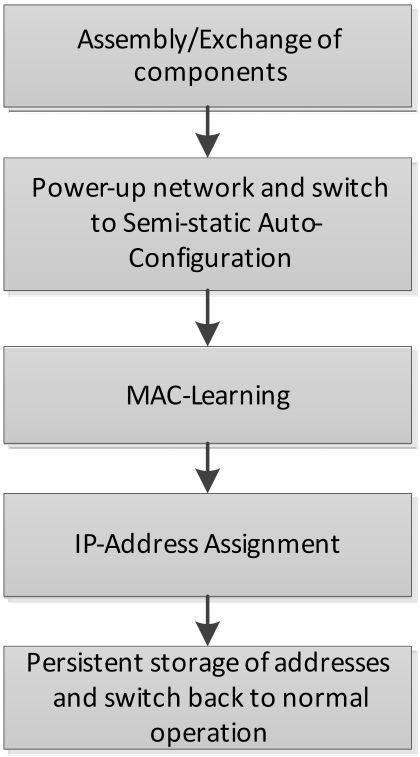

Figure 7.3: Learning Process

MAC-Learning (Optional Step): In this phase, messages need to be sent through the network and the switch will learn new MAC addresses (cf. Figure 7.4). These MAC-addresses will be stored in addition to predefined addresses, e.g. multicast MAC addresses which are configured during the vehicle network design. If static learning is

executed, i.e. MAC address will be persistently stored, it might be possible to add dynamically learned entries in the tables.

If software MAC learning is supported by switch hardware and the switch hardware expects an external μC (see Variant 2 and 3 in Figure 7.1), packets with unknown MAC Source Address will be routed to this μC. The MAC learning is done by integration code. It is intentionally not defined where this algorithm is located within the AUTOSAR stack as this might need a very time-optimized solution.

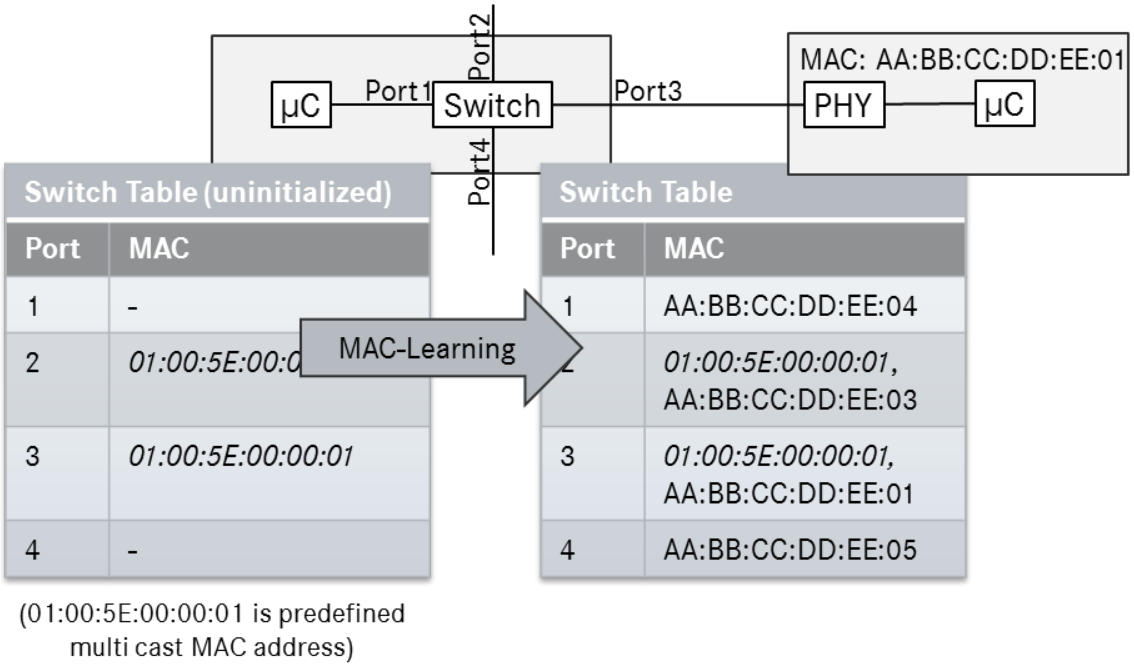

Figure 7.4: MAC-learning within the switch

IP-Address Assignment: In this phase, ECUs without a predefined IP-address will start to acquire an IP-address via DHCP (cf. Figure 7.5). Thus, these ECUs will run a DHCP-client while the ECU with the switch will run a DHCP server. In order to be able to assign always the same IP-address to a certain node, the DHCP server needs the information at which port the MAC address has been received. This port information can be interpreted as a "domain name" in the internet which is resolved to an IP address using a domain name server (DNS). With this port information the DHCP-server will assign the IP-address according to the IP-Assignment Table to the node. As mentioned above, this allows the assignment of MAC addresses by the Tier 1 and assignment of IP addresses by the OEM. With this mechanism it is also possible to assign different IP addresses to several VLANs at the same port. For this purpose, the IP-Assignment Table needs to be extended with a VLAN-column. Please note that the MAC-Learning-Phase can be combined with this phase.

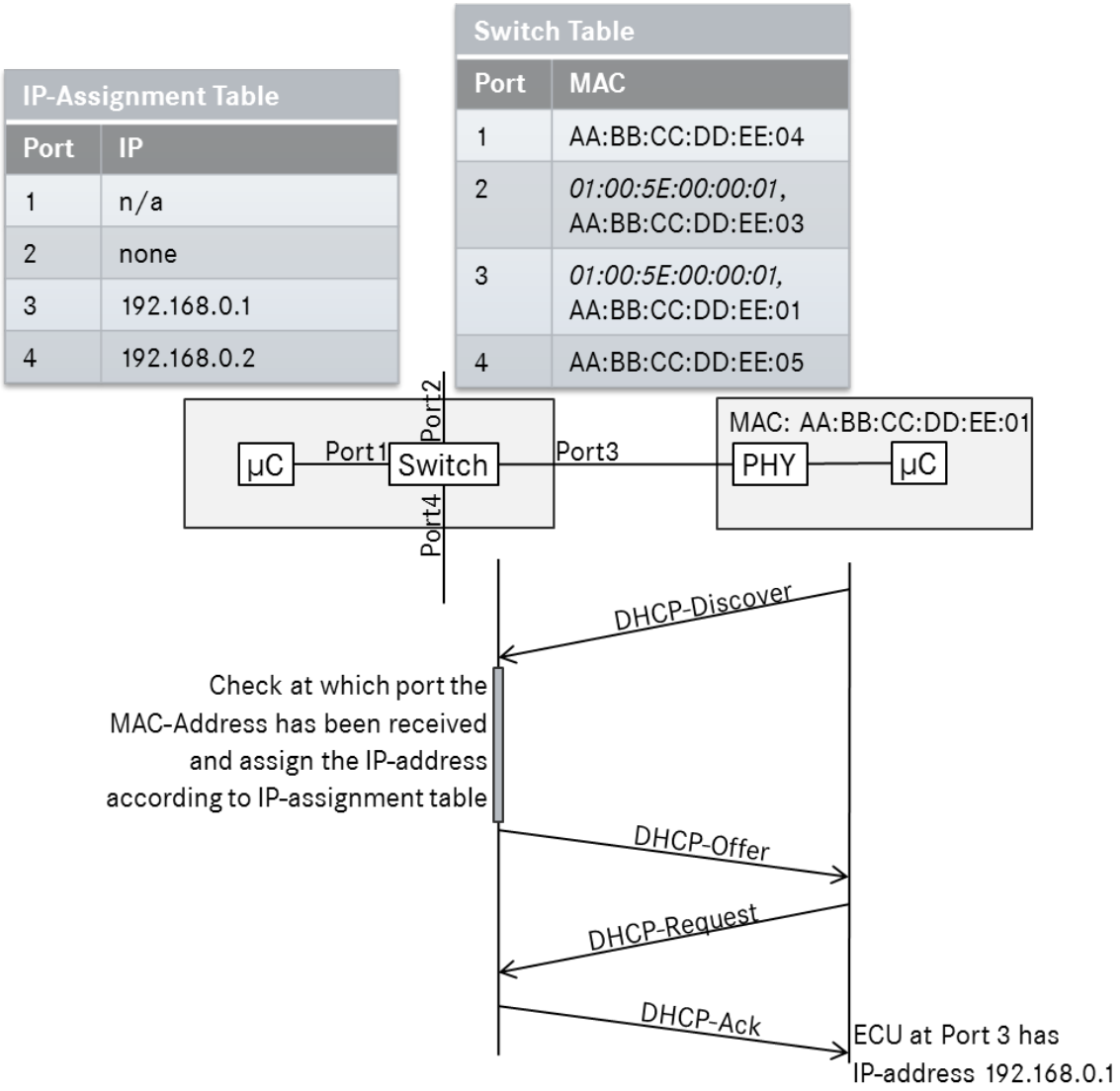

Figure 7.5: IP-address assignment via DHCP

[SWS_EthSwt_00136] ⌈The Ethernet Switch driver shall support an API which allows to store learned parameters like address resolution tables in a persistent manner by using the API `EthSwt_StoreConfiguration`. This persistent storage can be done in an NVRAM of the host CPU which runs the Ethernet Switch driver. Alternatively, this can be done in a memory of the switch itself. The trigger for storing the learned configuration or resetting the stored configuration can be done e.g. by a DCM.⌋*(SRS_ETH_00122, SRS_ETH_00087)*

[SWS_EthSwt_00181] ⌈The Ethernet Switch driver shall support an API which allows to reset learned parameters like address resolution tables by using the API `EthSwt_ResetConfiguration`.⌋*(SRS_ETH_00126, SRS_ETH_00087)*

[SWS_EthSwt_00162] ⌈The switch driver shall provide APIs to read the MAC-address to switch port mapping from the switch device to support the IP-address assignment by using the API `EthSwt_GetPortMacAddr`.⌋*(SRS_ETH_00087)*

[SWS_EthSwt_00407] ⌈Unused ARL table entries shall be removed from the ARL table after the timeout configured via `EthSwtArlTableEntryTimeout`, if this parameters is present in the configuration.⌋ *()*

7.1.5.2 Configuration of Egress Port Structure

As shown in Figure 7.6, the switch consists of a certain number of ports. Each port has its own set of egress FIFOs in which the incoming packets will be buffered. How the messages in the FIFOs will be forwarded depends mainly on the shaping and port scheduling mechanisms. Thus, the parameterization of the egress port influences the latency of messages within the network. Please note that the egress port structures in Figure 7.6 are meant as an example. Other structures with different FIFO numbers are possible.

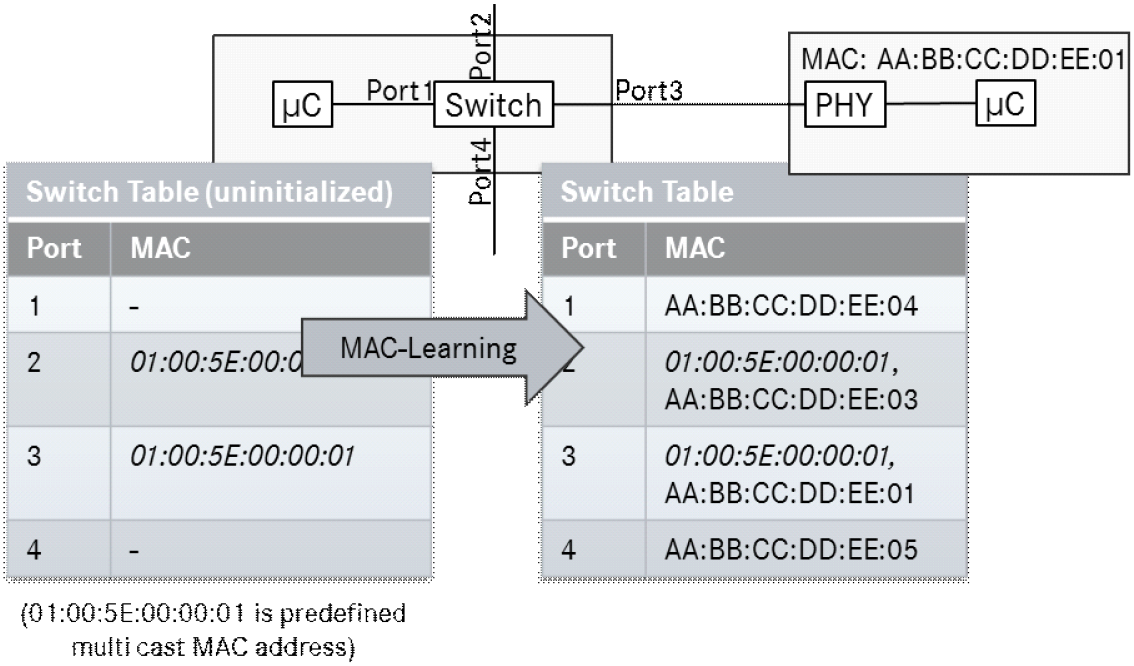

Figure 7.6: Ethernet Egress Port Structure

Considering the limitations of the hardware, such port structures shall be configurable within e.g. an initialization phase of the Ethernet Switch (see Section 10.1.6ff.)

[SWS_EthSwt_00132] ⌈The configuration of the Ethernet switch driver shall support different Ethernet egress port structures by the configuration `EthSwtPortEgress`.⌋ *(SRS_ETH_00121)*

Implementation note:

As Switch HW has very vendor specific structure a more specific description of the applying algorithms is not feasible here. The Routing takes place inside the Switch HW.

The configuration of the Schedulers is done with the container EthSwtPortEgress and its sub-container EthSwtPortScheduler with multiplicity 1 to *. As depicted in Figure 7.6 multiple schedulers on one port are possible. The Scheduler link which points from Predecessor to Predecessor is done with the references EthSwtPortEgressPredecessorRef. Shaper Algorithms are configured with the container EthSwtPortShaper and are linked to their FIFO through the EthSwtPortEgressPredecessorFifoRef. One FIFO can have multiple schedulers additionally to a shaper algorithm and the port-scheduler.

EthSwtPortSchedulerPredecessorOrder could be used to define the service weight of a FIFO if EthSwtPortSchedulerAlgorithm is configured to ETHSWT_SCHEDULER_DEFICIT_ROUND_ROBIN or ETHSWT_SCHEDULER_STRICT_PRIORITY. In case of EthSwtPortSchedulerAlgorithm is ETHSWT_SCHEDULER_STRICT_PRIORITY the EthSwtPortSchedulerPredecessorOrder defines the order of the schedulers,

The FIFO which is referenced (via Shaper or directly) by the Scheduler with the highest value of EthSwtPortSchedulerPredecessorOrder is the FIFO with the highest priority.

Besides the modeling of egress ports, it is necessary to specify how incoming packets are forwarded to the egress ports. For this purpose, different assignment policies of packets to egress port FIFOs are implemented in switches. As an example, the Ethernet priority field can be evaluated and mapped to a so-called traffic class. Such a traffic class is again mapped to an egress FIFO. Other header information of the Ethernet frame can be also used for the assignment of Ethernet frames to egress FIFOs. For the mapping to a certain traffic class, the following tables are necessary. While the first table shows the mapping of ingress-ports to traffic classes, the second table shows the priority-based mapping which can be defined per ingress port. Both tables are in conflict with each other, i.e. it has to be decided which mapping is applied.

Ingress-Port to Traffic Class Mapping:

Port-based Mapping	Traffic Class
e.g. Port2, Port3, Port4	7
e.g. Port1	6
-	5
-	4
-	3
-	2
-	1
-	0

PCP-field (Priority Code Point) to Traffic Class Mapping:

PCP-based Mapping	Traffic Class
Prio 0	7

△

Prio 1	6
Prio 2-7	5
-	4
-	3
-	2
-	1
-	0

After mapping the packets to a traffic class, they will be mapped to a certain FIFO at the egress side of the switch. This mapping can vary from egress port to egress port.

Traffic Class to FIFO Mapping:

Traffic Class	FIFO (if 4 FIFOs available)
7	3
6	2
5-0	1
-	0

While the frame forwarding is a hardware mechanism of the switch, the tables how the frames will be forwarded shall be configurable (see Section 10.1.12ff.).

Please note that the traffic class assignment is done after the priority regeneration.

[SWS_EthSwt_00133] ⌈The switch configuration shall support to configure the Ethernet frame forwarding mechanisms of a switch by the configuration parameters `EthSwtPortTrafficClassAssignment`, `EthSwtPriorityTrafficClassAssignment`, `EthSwtPortFifoTrafficClassAssignment`.⌋*(SRS_ETH_00121)*

[SWS_EthSwt_00234] ⌈The Parameter `EthSwtPortFifoMinimumLength` shall define the minimum length for one dedicated FIFO on one Switch.⌋*(SRS_ETH_00121)*

Note: The actual length can be longer. The decision on the length is very likely to be taken by the Switch HW or fixed by the Switch design. To define the minimum in ECUC is supposed to guarantee that some priorities have enough egress buffer.

7.1.5.3 Vlan-Membership on Switch-ports

Every Port holds the list of Vlan-Memberships. This Membership describe ingress and egress behavior in terms of filtering, and rate policing and tagging or untagging.

For each VLAN identifier a table is necessary which stores at which egress port the corresponding VLAN is tagged or untagged. For an 8-port switch, this table could look like the following example where T stands for tagging and U for untagging:

VLAN Forwarding Table								
VLAN-ID	Port Number							
	1	2	3	4	5	6	7	8
1	T	T	-	U	-	-	-	T
2	T	U	-	T	-	-	-	T
...								
4094								

Incoming packets which contain a VLAN-ID of e.g. 1 can be forwarded to the ports 1, 2, 4, and 8. At ports 1, 2, and 8 these packets will be transmitted with the VLAN tag and at port 4 the tag will be removed. If a broadcast message with e.g. VLAN-ID 2 will be received at port 2. It will be forwarded to port 1, 4, and 8. The other ports 3, 5, 6, and 7 are not in the same VLAN. Thus, the packet will not be forwarded to these egress ports. The table considers only messages, which contain a VLAN-ID within the switch. (see also 10.1.12).

[SWS_EthSwt_00134] ⌈The switch configuration shall support the configuration how packets will be forwarded with respect to configured VLANs by using the configuration parameters of the subcontainer `EthSwtPortVlanMembership`.⌋*(SRS_ETH_00121, SRS_ETH_00114)*

Note: VLAN-Memberships of a port are modeled with the container `EthSwtPortVlanMembership` where the `EthSwtPortVlanDefaultPriority` and `EthSwtPortVlanForwardingType` are configured.

7.1.5.4 Rate Policing on Ingress Side

If HW supports Rate Policing a policer can be configured using the parameters `EthSwtPortRatePolicedTimeInterval`, `EthSwtPortRatePolicedByteCount`, `EthSwtPortRatePolicedPriority` and `EthSwtPortRateVlanMembershipRef`.

It is possible to rate only on Priority configuring no membership reference. If the policing shall only check the Vlan, the Priority can be omitted.

An example of a rate limit definition might be a maximum number of data Bytes inside the payload allowed to pass over a 5 ms period.

The policer can either drop the violating frame or block the violating Source based on the MAC-Address depending on the configuration parameter `EthSwtPortRateViolationAction`.

7.1.5.5 VLAN-modification at ingress side

Another table specifies a port-based modification of the VLAN-ID or an insertion of the VLAN-ID into the Ethernet message:

Ingress VLAN Modification/Insertion Table								
Port Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
VLAN-ID	2	-	-	6	-	-	-	-

In this example, all incoming messages at port one will get the VLAN-ID 2 no matter they already had one before. At port 4, all incoming messages will get a 6 as their VLAN-ID. At the remaining ports, no VLAN-IDs will be inserted and an existing VLAN-ID in the Ethernet-message will remain without modification.

[SWS_EthSwt_00135] ⌈The switch configuration shall support the configuration how VLANs will be inserted into packets or existing VLANs will be modified by the configuration `EthSwtPortIngressVlanModification`.⌋*(SRS_ETH_00121)*

7.1.5.6 Priority-Code-Point-Regeneration

Within the VLAN-tag, the PCP-field (priority code point) is another parameter which can be modified at an ingress port of an Ethernet switch. For this purpose a so-called priority regeneration table has to be defined:

Priority Regeneration Table								
Ingress PCP	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Regenerated PCP	0	1	2	3	4	5	6	7

Table 7.1: In this table, the "Ingress PCP" is mapped to the "Regenerated PCP".

[SWS_EthSwt_00178] ⌈The switch configuration shall support the configuration how the PCP field of incoming packets will be modified before they are forwarded to the egress port, i.e. a priority regeneration table can be configured (Please also refer to `EthSwtPriorityRegeneration`,`EthSwtPriorityRegenerationIngressPriority` and `EthSwtPriorityRegenerationRegeneratedPriority`.⌋*(SRS_ETH_00121)*

7.1.5.7 Direct Traffic Class Assignement

[SWS_EthSwt_00179] ⌈The switch configuration shall support the configuration of a default traffic class for incoming frames (Please also refer to `EthSwtPortTrafficClassAssignment`).⌋*(SRS_ETH_00121)*

7.1.5.8 Behavior in case of untagged frames in a VLAN network

There are three ways to handle untagged frames:

- Drop all untagged frames at ingreess side of the port
- Forward untagged
- Tag all untagged frames with a default Vlan and default priority.

Implementation Hint: If there is a Vlan-Tag there is also a priority.

To drop all untagged frames at the ingress side of the port the parameter `EthSwtPortIngressDropUntagged` need be set to `TRUE` and the parameters `EthSwtPortIngressDefaultVlan` and `EthSwtPortIngressDefaultPriority` need not be set by setting the multiplicity of both parameters to `0`. To add a Default Tag to the untagged frame the parameter `EthSwtPortIngressDropUntagged` need be set to `FALSE` and the parameters `EthSwtPortIngressDefaultVlan` and `EthSwtPortIngressDefaultPriority` need be set to the intended tag.

To forward untagged frames from ingress to egress side the parameter `EthSwtPortIngressDropUntagged` need be set to `FALSE` and the parameters `EthSwtPortIngressDefaultVlan` and if `EthSwtPortIngressDefaultPriority` can be set according to internal forwarding rules. This Default Vlan than needs to be configured to be untagged on the egress port by `EthSwtPortVlanForwardingType` set to `ETHSWT_SENT_UNTAGGED`.

Note: The handling of untagged frames by the HW is expected to be located before all other modifications of the VLan and the Priority and before the Traffic Class assignement.

7.1.5.9 Behavior in case of double tagged frames in a VLAN network

[SWS_EthSwt_00233] ⌈The Switch Driver shall support the configuration of dropping double tagged frames via the configuration parameter `EthSwtDropDoubleTagged` if the Switch hardware supports dropping of double tagged frames.⌋*(SRS_Eth_00114)*

7.1.5.10 Switch Management support

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

For an introduction of the basic HW architecture and interaction, please refer to [4, SWS_EthernetDriver].

[SWS_EthSwt_00240] ⌈The Switch driver shall offer Switch management APIs

- EthSwt_EthRxProcessFrame
- EthSwt_EthRxFinishedIndication
- EthSwt_EthTxAdaptBufferLength
- EthSwt_EthTxPrepareFrame
- EthSwt_SetMgmtInfo
- EthSwt_EthTxProcessFrame and
- EthSwt_EthTxFinishedIndication

if EthSwtManagementSupportApi is set to TRUE.⌋(*SRS_BSW_00171*, *SRS_ETH_00125*)

Note: Switch management APIs support the EthIf to gather / modify Switch-Port specific communication attributes.

[SWS_EthSwt_00241] ⌈The Switch Driver management APIs

- EthSwt_EthRxProcessFrame
- EthSwt_EthRxFinishedIndication
- EthSwt_EthTxAdaptBufferLength
- EthSwt_EthTxPrepareFrame
- EthSwt_SetMgmtInfo
- EthSwt_EthTxProcessFrame and
- EthSwt_EthTxFinishedIndication

shall support the Ethernet Driver to gather the Switch specific management information out of an Ethernet frame for reception or to prepare an Ethernet frame for management mode conformant frame transmission, e.g. the egress route of a frame.⌋(*SRS_ETH_00125*)

[SWS_EthSwt_00242] ⌈The Switch Driver management APIs EthSwt_EthTxProcessFrame and EthSwt_EthTxFinishedIndication shall return immediately, if EthSwt_SetMgmtInfo has not been called before a call of EthSwt_EthTxProcessFrame.⌋(*SRS_ETH_00125*)

7.1.5.11 Global Time support

For more details regarding time measurement with Switches, please refer to [12, SWS_TimeSyncOverEthernet].

[SWS_EthSwt_00243] ⌈The Switch driver shall access the port specific hardware time stamps if EthSwtPortTimeStampSupport of the port is set to TRUE.⌋*(SRS_BSW_00171, SRS_ETH_00125)*

[SWS_EthSwt_00378] ⌈If EthSwt_PortEnableTimeStamp is called for a PortIdx, the switch driver shall enable the time-stamping for this port if EthSwtPortTimeStampSupport is set to TRUE for this port.⌋*(SRS_ETH_00125)*

[SWS_EthSwt_00245] ⌈The Switch driver shall inform the EthIf about the availability of port specific ingress and egress timestamps using the APIs EthIf_SwitchIngressTimeStampIndication and EthIf_SwitchEgressTimeStampIndication , if EthSwtGlobalTimeSupportApi is set to TRUE.⌋*(SRS_ETH_00125)*

Note: Global Time support typically requires the activation of the Switch management support functionality within the Switch device.

7.1.5.12 Counter synchronization of Ethernet switches which are connected via uplink ports

Some Ethernet Switches provide the possibility to synchronize their internal clock. For Ethernet switches which are connected via uplink ports it is not necessary to measure the delay between the connected uplink ports, if the clock synchronization clock is activated (EthSwtClockSynchronizationSupport set to TRUE).

[SWS_EthSwt_00408] ⌈The Switch driver shall enable clock synchronization with another Ethernet switch to which it is connected via uplink port, if EthSwtClockSynchronizationSupport is set to TRUE.⌋*()*

[SWS_EthSwt_CONSTR_00409] ⌈The port specific timestamping (EthSwtPortTime StampSupport) can be set to TRUE, if clock synchronization for connected Ethernet switches is deactivated (EthSwtClockSynchronizationSupport set to FALSE).⌋*()*

[SWS_EthSwt_CONSTR_00410] ⌈The port specific timestamping (EthSwtPortTimeStampSupport) can be set to TRUE, if EthSwtClockSynchronizationSupport is activated and EthSwtPortRole is not ETHSWT_UP_LINK_PORT. Eth SwtPorts with EthSwtPortRole ETHSWT_UP_LINK_PORT are connected to another Ethernet switch and not considered for the time delay compensation, if EthSwtClockSynchronizationSupport is activated.⌋*()*

7.1.5.13 Verification of Configuration

There are some situations where the Host controller needs to verify the Switch configuration.

[SWS_EthSwt_00292] ⌈If the parameter EthSwtVerifyConfigApi is set to TRUE the function EthSwt_VerifyConfig shall be used to verify switch configuration.⌋ *(SRS_Eth_00126)*

Implementation hint: As Switch configuration is highly HW-Architecture dependent the steps inside the function are implementation specific.

In some use cases, it is necessary to stop frame forwarding during the verification using the optional function EthSwt_SetForwardingMode

The function EthSwt_VerifyConfig could for example do the following steps:

- Stop frame forwarding by calling EthSwt_SetForwardingMode(FALSE).
- Verify the switch configuration
- In case the switch configuration is valid then frame forwarding shall be enabled by calling EthSwt_SetForwardingMode(TRUE) (if disabled in step 1).
- In case the switch configuration is not valid then the switch shall be reset and reconfigured.

Note: Please note that a reset of the Host Controller does not necessarily need a reset of the connected Switch HW. This needs to be evaluated individually very carefully as a reset raises the risk of uncontrolled communication during reset phase of the host controller.

Note: The Verification of the Switch Configuration as described above is just an example how and when this Verification may be done. It is very dependent on the used switch HW as well as the individual HW-Architecture and even Power supply and Reset strategy of the Switch of the ECU how the Configuration is verified or even how it can be verified. The only thing what this Module specifies is the interface to the upper layer to apply some verification on the switch configuration.

7.1.5.14 Testing and Diagnostic of Switch Ports

If configured, the Ethernet Switch Driver provides following interfaces to apply Testing and diagnostic functionalities

- EthSwt_GetPortSignalQuality
- EthSwt_GetPortIdentifier
- EthSwt_GetSwitchIdentifier
- EthSwt_WritePortMirrorConfiguration
- EthSwt_ReadPortMirrorConfiguration
- EthSwt_GetPortMirrorState
- EthSwt_SetPortMirrorState

- EthSwt_SetPortTestMode
- EthSwt_SetPortLoopbackMode
- EthSwt_SetPortTxMode
- EthSwt_GetPortCableDiagnosticsResult
- EthSwt_GetCfgDataRaw
- EthSwt_GetCfgDataInfo

The Availabiliy of these functions is strongly depending on the possibilities of the used Transceiver-(Phy)-HW.

7.1.5.15 Low Power Mode Support

[SWS_EthSwt_00376] ⌈If EthSwtLowPowerModeSupport is set to TRUE and at least one EthSwtPort of a Ethernet switch is enabled and the corresponding Ethernet switch HW is in an inactive or low power mode the Ethernet switch HW shall be set to an active mode in which forwarding of Ethernet frames is possible.⌋*()*

[SWS_EthSwt_00377] ⌈If EthSwtLowPowerModeSupport is set to TRUE and no Eth SwtPort for a certain Ethernet switch is enabled, the corresponding Ethernet switch HW shall be set to an inactive or low power mode.⌋*()*

7.2 Error Classifications

Section 7.2 "Error Handling" of the document [8, SWS_BSW General] describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below

7.2.1 Development Errors

[SWS_EthSwt_00001] ⌈

Type of error	*Related error code*	*Error value*
Invalid switch index	ETHSWT_E_INV_SWITCH_IDX	0x01
EthSwt module was not initialized	ETHSWT_E_UNINIT	0x02
Invalid pointer in parameter list	ETHSWT_E_PARAM_POINTER	0x03
Invalid API which is not available by another module	ETHSWT_E_INV_API	0x05

▽

△

Type of error	Related error code	Error value
Invalid switch port index	ETHSWT_E_INV_SWITCHPORT_IDX	0x06
Invalid Controller Index	ETHSWT_E_INV_CTRL_IDX	0x07
Invalid input parameter	ETHSWT_E_INV_PARAM	0x08
Invalid configuration	ETHSWT_E_INIT_FAILED	0x09

⌋(*SRS_BSW_00385*)

[SWS_EthSwt_00009] ⌈If development error detection is enabled, the function `EthSwt_Init` shall check the parameter `CfgPtr` for being valid. If the check fails, `EthSwt_Init` shall raise the development error ETHSWT_E_INIT_FAILED.⌋(*SRS_BSW_323*, *SRS_BSW_369*)

Note: Please note that in case of variant pre-compile `NULL_PTR` is allowed.

[SWS_EthSwt_00164] ⌈The switch driver shall check whether the lower layer driver, i.e. the EthTrcv provides the APIs which can be called by an upper layer module (Eth If) of the switch driver and will be forwarded to the lower layer. In case of missing APIs, the switch driver shall raise the development error ETHSWT_E_INV_API if APIs are missing in the lower layer module.⌋(*SRS_BSW_00369*, *SRS_BSW_00386*, *SRS_ETH_00118*)

Note: This check will be performed upon calling a certain API. For this check the input parameter `SwitchPortIdx` and a configuration table which needs to be derived from the configuration of the Ethernet transceiver drivers which are attached to the Ethernet switch driver are necessary. This functionality is necessary if development error tracing is activated. This check is necessary because an Ethernet switch driver API can be called by an upper layer module with the argument `SwitchPortIdx`. This value of this `SwitchPortIdx` can be in a valid range, but some Ethernet transceiver driver which are used by the switch driver support the API and some do not support this API. In order to resolve this conflict, this check has been implemented.

[SWS_EthSwt_00156] ⌈The function `EthSwt_SetSwitchPortMode` shall check whether the `EthTrcv_SetTransceiverMode` API of the indexed transceiver driver is available by checking whether for this `SwitchPortIdx` the corresponding `EthTrcv` API is available. If this is not the case, the function shall raise the development error ETHSWT_E_INV_API.⌋(*SRS_BSW_00413*, *SRS_BSW_323*, *SRS_BSW_369*, *SRS_ETH_00118*)

[SWS_EthSwt_00157] ⌈The function `EthSwt_GetSwitchPortMode` shall check whether the `EthTrcv_GetTransceiverMode` API of the indexed transceiver driver is available by checking whether for this `SwitchPortIdx` the corresponding `EthTrcv` API is available. If this is not the case, the function shall raise the development error ETHSWT_E_INV_API.⌋(*SRS_BSW_00413*, *SRS_BSW_323*, *SRS_BSW_369*, *SRS_ETH_00118*)

[SWS_EthSwt_00386] ⌈If development error detection is activated by `EthSwtDevErrorDetect`, all functions except `EthSwt_Init` shall check that the service `Eth-`

Swt_Init was previously called. If the check fails, the function shall raise the development error ETHSWT_E_UNINIT.⌋(*SRS_BSW_00350*)

[SWS_EthSwt_00387] ⌈If development error detection is activated by EthSwtDevErrorDetect, all functions with input parameter SwitchIdx shall check the parameter for being valid. If the check fails, the functions shall raise the development error ETHSWT_E_INV_SWITCH_IDX.⌋(*SRS_BSW_00350*)

[SWS_EthSwt_00389] ⌈If development error detection is enabled, all functions with input parameter SwitchPortIdx or PortIdx shall check the parameter for being valid. If the check fails, the functions shall raise the development error ETHSWT_E_INV_SWITCH_IDX.⌋(*SRS_BSW_00350*)

[SWS_EthSwt_00390] ⌈If development error detection is enabled, all functions with input parameter CtrlIdx shall check the parameter for being valid. If the check fails, the functions shall raise the development error ETHSWT_E_INV_CTRL_IDX.⌋(*SRS_BSW_00350*)

[SWS_EthSwt_00391] ⌈If development error detection is enabled, all functions with input parameter BufIdx shall check the parameter for being valid. If the check fails, the functions shall raise the development error ETHSWT_E_INV_PARAM.⌋(*SRS_BSW_00350*)

[SWS_EthSwt_00392] ⌈If development error detection is enabled, all functions with inout or output pointer parameter shall check the parameter for being valid. If the check fails, the functions shall raise the development error ETHSWT_E_PARAM_POINTER.⌋(*SRS_BSW_00350*)

[SWS_EthSwt_00393] ⌈If development error tracing is activated by EthSwtDevErrorDetect, the functions which call an Ethernet Transceiver API and do not obtain the functionality directly from the switch port interface shall check whether the API of the indexed transceiver driver is available. If this is not the case, the functions shall raise the development error ETHSWT_E_INV_API.⌋(*SRS_BSW_00350*)

[SWS_EthSwt_00154] ⌈If development error detection is activated by EthSwtDevErrorDetect, the function EthSwt_GetLinkState shall check whether the EthTrcv_GetLinkState API of the indexed transceiver driver is available by checking whether for this SwitchPortIdx the corresponding EthTrcv API is available. If this is not the case, the function shall raise the development error ETHSWT_E_INV_API.⌋(*SRS_ETH_00118, SRS_ETH_00119, SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369*)

7.2.2 Runtime Errors

[SWS_EthSwt_00434] ⌈

Type of error	*Related error code*	*Error value*
Initialization of ports is not finished	ETHSWT_INIT_NOT_COMPLETED	0x01

⌋()

7.2.3 Transient Faults

There are no transient faults.

7.2.4 Production Errors

There are no production errors.

7.2.5 Extended Production Errors

[SWS_EthSwt_00113] ⌈

Error Name:	ETHSWT_E_ACCESS	
Short Description:	Ethernet Switch Access Failure	
Long Description:	This production error shall be issued when the switch is not accessible.	
Recommended DTC:	N/A	
Detection Criteria:	Fail	When access to the Ethernet Switch fails the module shall report the extended production error with event status DEM_EVENT_STATUS_ PREFAILED to DEM.
	Pass	When access to the Ethernet Switch succeeds the module shall report the extended production error with event status DEM_EVENT_STATUS_ PREPASSED to DEM.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

Table 7.2: ETHSWT_E_ACCESS

⌋(*SRS_BSW_00385*)

[SWS_EthSwt_00395] ⌈

Error Name:	ETHSWT_E_SYNCPORT2PHY

▽

△

Short Description:	Ethernet switch port and the referenced Ethernet transceiver are in contradicting modes.	
Long Description:	While getting the Ethernet switch port mode, the Ethernet switch driver detected an inconsistent state between Ethernet switch port and the referenced Ethernet transceiver Mode.	
Recommended DTC:	N/A	
Detection Criteria:	Fail	When getting the Ethernet switch port mode together with the Ethernet transceiver mode and the mode of the two referenced modules was found inconsistent the module shall report the extended production error with event status DEM_EVENT_STATUS_ PREFAILED to DEM.
	Pass	When getting the Ethernet switch port mode together with the Ethernet transceiver mode and the mode of the two referenced modules was found consistent the module shall report the extended production error with event status DEM_EVENT_STATUS_ PREPASSED to DEM.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

Table 7.3: ETHSWT_E_SYNCPORT2PHY

⌋(*SRS_BSW_00385*)

8 API specification

8.1 Imported types

This chapter lists all types included from the following files:

[SWS_EthSwt_00002] ⌈

Module	*Header File*	*Imported Type*
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Eth	Eth_GeneralTypes.h	Eth_BufIdxType
	Eth_GeneralTypes.h	Eth_CounterType
	Eth_GeneralTypes.h	Eth_DataType
	Eth_GeneralTypes.h	Eth_MacVlanType
	Eth_GeneralTypes.h	Eth_ModeType (draft)
	Eth_GeneralTypes.h	Eth_RxStatsType
	Eth_GeneralTypes.h	Eth_TimeStampType

▽

△

Module	Header File	Imported Type
	Eth_GeneralTypes.h	Eth_TxErrorCounterValuesType
	Eth_GeneralTypes.h	Eth_TxStatsType
EthTrcv	Eth_GeneralTypes.h	EthTrcv_BaudRateType
	Eth_GeneralTypes.h	EthTrcv_CableDiagResultType
	Eth_GeneralTypes.h	EthTrcv_DuplexModeType
	Eth_GeneralTypes.h	EthTrcv_LinkStateType
	Eth_GeneralTypes.h	EthTrcv_PhyLoopbackModeType
	Eth_GeneralTypes.h	EthTrcv_PhyTestModeType
	Eth_GeneralTypes.h	EthTrcv_PhyTxModeType
	Eth_GeneralTypes.h	EthTrcv_WakeupReasonType
NvM	Rte_NvM_Type.h	NvM_BlockIdType
	Rte_NvM_Type.h	NvM_BlockRequestType
	Rte_NvM_Type.h	NvM_RequestResultType
Spi	Spi.h	Spi_AsyncModeType
	Spi.h	Spi_ChannelType
	Spi.h	Spi_DataBufferType
	Spi.h	Spi_NumberOfDataType
	Spi.h	Spi_SequenceType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋()

8.2 Type definitions

8.2.1 EthSwt_StateType

[SWS_EthSwt_00123] ⌈

Name	EthSwt_StateType		
Kind	Enumeration		
Range	ETHSWT_STATE_UNINIT	0x00	Switch is not yet configured
	ETHSWT_STATE_INIT	0x01	Switch driver is initialized
	ETHSWT_STATE_ PORTINIT_COMPLETED	0x02	Port initialization is completed
	ETHSWT_STATE_ACTIVE	0x03	Switch is active
Description	Status supervision used for Development Error Detection. The state shall be available for debugging.		
Available via	Eth_GeneralTypes.h		

⌋(*SRS_BSW_00406*)

8.2.2 EthSwt_ConfigType

[SWS_EthSwt_00165] ⌈

Name	EthSwt_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	–
Description	Implementation specific structure of the post build configuration.	
Available via	EthSwt.h	

⌋(*SRS_BSW_00395*)

8.2.3 EthSwt_MacLearningType

[SWS_EthSwt_00227] ⌈

Name	EthSwt_MacLearningType		
Kind	Enumeration		
Range	ETHSWT_ MACLEARNING_ HWDISABLED	–	If hardware learning disabled, the switch must not learn new MAC addresses
	ETHSWT_ MACLEARNING_ HWENABLED	–	If hardware learning enabled, the switch learns new MAC addresses
	ETHSWT_ MACLEARNING_ SWENABLED	–	If software learning enabled, the hardware learning is disabled and the switch forwards packets with an unknown source address to a host CPU
Description	The interpretation of this value		
Available via	Eth_GeneralTypes.h		

⌋(*SRS_ETH_00087*)

8.2.4 EthSwt_MgmtInfoType

[SWS_EthSwt_91002] ⌈

Name	EthSwt_MgmtInfoType	
Kind	Structure	
Elements	SwitchIdx	
	Type	uint8
	Comment	Switch index
	SwitchPortIdx	

▽

△

	Type	uint8
	Comment	Port index of the switch
Description	Type for holding the management information received/transmitted on Switches (ports).	
Available via	Eth_GeneralTypes.h	

⌋(*SRS_Eth_00125*)

8.2.5 EthSwt_PortMirrorCfgType

[SWS_EthSwt_91017] ⌈

Name	EthSwt_PortMirrorCfgType	
Kind	Structure	
Elements	srcMacAddrFilter	
	Type	Array of uint8
	Size	6
	Comment	Specifies the source MAC address [0..255,0..255,0..255,0..255,0..255,0..255] that should be mirrored. If set to 0,0,0,0,0,0, no source MAC address filtering shall take place.
	dstMacAddrFilter	
	Type	Array of uint8
	Size	6
	Comment	Specifies the destination MAC address [0..255,0..255,0..255,0..255,0..255,0..255] that should be mirrored. If set to 0,0,0,0,0,0, no destination MAC address filtering shall take place.
	VlanIdFilter	
	Type	uint16
	Comment	Specifies the VLAN address 0..4094 that should be mirrored. If set to 65535, no VLAN filtering shall take place.
	MirroringPacketDivider	
	Type	uint8
	Comment	Divider if only a subset of received frames should be mirrored. E.g. MirroringPacketDivider = 2 means every second frames is mirrored
	MirroringMode	
	Type	uint8
	Comment	specifies the mode how the mirrored traffic should be tagged : 0x00 == No VLAN retagging; 0x01 == VLAN retagging; 0x02 == VLAN Double tagging
	TrafficDirectionIngressBitMask	
	Type	uint32

▽

△

	Comment	Specifies the bit mask of Ethernet switch ingress port traffic direction to be mirrored. The bit mask is calculated depending of the values of Eth SwtPortIdx. (e.g. set EthSwtPortIdx == 2 => TrafficDirectionIngressBit Mask = 0b0000 0000 0000 0000 0000 0000 0000 0100). 0b0 == enable ingress port mirroring 0b1 == disable ingress port mirroring Example: TrafficDirectionIngressBitMask = 0b0000 0000 0000 0000 0000 0000 0000 0100 => Ingress traffic mirroring is enabled of Ethernet switch port with EthSwtPortIdx=2
	TrafficDirectionEgressBitMask	
	Type	uint32
	Comment	Specifies the bit mask of Ethernet switch egress port traffic direction to be mirrored. The bit mask is calculated depending of the values of Eth SwtPortIdx. (e.g. set EthSwtPortIdx == 2 => TrafficDirectionEgressBit Mask = 0b0000 0000 0000 0000 0000 0000 0000 0100). 0b0 == enable egress port mirroring 0b1 == disable egress port mirroring Example: TrafficDirectionEgressBitMask = 0b0000 0000 0000 0000 0000 0000 0000 0001 => Egress traffic mirroring is enabled of Ethernet switch port with EthSwtPortIdx=0
	CapturePortIdx	
	Type	uint8
	Comment	Specifies the Ethernet switch port which capture the mirrored traffic
	ReTaggingVlanId	
	Type	uint16
	Comment	Specifies the VLAN address 0..4094 which shall be used for re-tagging if MirroringMode is set to 0x01 (VLAN re-tagging). If the value is set to 65535, the value shall be ignored, because the VLAN address for re-tagging is provided by the Ethernet switch configuration
	DoubleTaggingVlanId	
	Type	uint16
	Comment	Specifies the VLAN address 0..4094 which shall be used for double-tagging if MirroringMode is set to 0x02 (VLAN double tagging). If the value is set to 65535, the value shall be ignored, because the VLAN address for double tagging is provided by the Ethernet switch configuration
Description	The EthSwt_PortMirrorCfgType specify the port mirror configuration which is set up per Ethernet switch. The configuration is written to the Ethernet switch driver by calling EthSwt_WritePortMirror Configuration. One port mirror configuration is maintained per Ethernet Switch.	
Available via	Eth_GeneralTypes.h	

⌋(*SRS_ETH_00123*)

8.2.6 EthSwt_PortMirrorStateType

[SWS_EthSwt_91020] ⌈

Name	EthSwt_PortMirrorStateType		
Kind	Enumeration		
Range	PORT_MIRRORING_ DISABLED	0x00	port mirroring disabled

▽

△

	PORT_MIRRORING_ ENABLED	0x01	port mirroring enabled
Description	Type to request or obtain the port mirroring state (enable/disable) for a particular port mirror configuration per Ethernet switch.		
Available via	Eth_GeneralTypes.h		

⌋(*SRS_ETH_00123*)

8.2.7 EthSwt_ReturnType

[SWS_EthSwt_91033] ⌈

Range	ETHSWT_PORT_MIRRORING_ CONFIGURATION_NOT_ SUPPORTED	0x02	port mirroring configuration is not supported by Ethernet switch driver or by the Ethernet switch hardware
Description	Overlayed return value of Std_ReturnType for Ethernet switch driver API EthSwt_WritePortMirror Configuration, if the port mirroring configuration is not supported by Ethernet switch driver or by the Ethernet switch hardware (e.g. the configured mirrored traffic direction (see SWS_EthSwt_91017 "TrafficDirectionIngressBitMask" and "TrafficDirectionEgressBitMask") for ingress and egress traffic of the same port is not supported, or the addressed Ethernet switch ports within the port mirror configuration are not accessible by the Ethernet switch driver)		
Available via	Eth_GeneralTypes.h		

⌋()

8.2.8 EthSwt_MgmtOwner

[SWS_EthSwt_91035] ⌈

Name	EthSwt_MgmtOwner		
Kind	Enumeration		
Range	ETHSWT_MGMT_OBJ_ UNUSED	0x00	Object unused
	ETHSWT_MGMT_OBJ_ OWNED_BY_ETHSWT	0x01	Object used and EthSwt collects needed data
	ETHSWT_MGMT_OBJ_ OWNED_BY_UPPER_ LAYER	0x02	Object used and the upper layer does calculations
Description	Holds information if upper layer or EthSwt is owner of mgmt_obj.		
Available via	Eth_GeneralTypes.h		

⌋()

8.2.9 EthSwt_Mgmt_ObjectType

[SWS_EthSwt_91037] ⌈

Name	EthSwt_MgmtObjectType	
Kind	Structure	
Elements	Validation	
	Type	EthSwt_MgmtObjectValidType
	Comment	The validation information for the mgmt_obj.
	IngressTimestamp	
	Type	Eth_TimeStampType
	Comment	The ingress timestamp value out of the switch.
	EgressTimestamp	
	Type	Eth_TimeStampType
	Comment	The egress timestamp value out of the switch.
	MgmtInfo	
	Type	EthSwt_MgmtInfoType
	Comment	Received/Transmitted Management information of the switches.
	Ownership	
	Type	EthSwt_MgmtOwner
	Comment	The ownership of MgmtObj.
Description	Provides information about all struct member elements. The ownership gives information whether EthSwt has finished its activities in providing all struct member elements.	
Available via	Eth_GeneralTypes.h	

⌋()

[SWS_EthSwt_00433] ⌈A MgmtObject is just allowed to be owned between EthSwt and only one <UPPER_LAYER>. The structure element can be identified unambiguously using the DataPtr in Rx- and BufIdx in Tx-context, because both elements are definitively unique within the RxIndication() / TxConfirmation() context.⌋()

8.2.10 EthSwt_MgmtObjectValidType

[SWS_EthSwt_91036] ⌈

Name	EthSwt_MgmtObjectValidType	
Kind	Structure	
Elements	IngressTimestampValid	
	Type	Std_ReturnType
	Comment	IngressTimestampValid shall be set to E_NOT_OK if ingress timestamp is not available
	EgressTimestampValid	
	Type	Std_ReturnType

▽

△

	Comment	EgressTimestampValid shall be set to E_NOT_OK if ingress timestamp is not available.
	MgmtInfoValid	
	Type	Std_ReturnType
	Comment	MgmtInfoValid shall be set to E_NOT_OK if ingress timestamp is not available(e.g. timeout).
Description	Will be set from EthSwt and marks EthSwt_MgmtObject as valid or not. So the upper layer will be able to detect inconsistencies.	
Available via	Eth_GeneralTypes.h	

⌋()

8.3 Function definitions

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

8.3.1 EthSwt_Init

[SWS_EthSwt_00006] ⌈

Service Name	EthSwt_Init	
Syntax	`void EthSwt_Init ( const EthSwt_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 Switch Driver	
Available via	EthSwt.h	

⌋(*SRS_BSW_00101*)

[SWS_EthSwt_00007] ⌈The function `EthSwt_Init` shall store the access to the configuration structure for subsequent API calls.⌋(*SRS_BSW_00101*)

[SWS_EthSwt_00008] ⌈The function `EthSwt_Init` shall change the state of all switches controlled by this Switch Driver from `ETHSWT_STATE_UNINIT` to `ETHSWT_STATE_INIT`.⌋(*SRS_BSW_00101*)

[SWS_EthSwt_00421] ⌈The EthSwt shall check for enabled port mirror configuration. The enabled port mirror configuration shall be activated by reconfiguring the Ethernet

switch hardware according to the port mirror configuration, before frame forwarding is being enabled.⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00422] ⌈If the PortMirrorState is set to 0x01 (port mirroring enabled), then the stored port mirror configuration for the given Ethernet switch shall be written to hardware registers of the given Ethernet switch and enable port mirroring.⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00423] ⌈If the PortMirrorState is set to 0x00 (port mirroring disabled) the corresponding hardware registers of the given Ethernet switch shall be reset (to the HW's default values) and the port mirroring shall be disabled.⌋*()*

[SWS_EthSwt_00011] ⌈After initialization of the Ethernet switch within the `EthSwt_BackgroundTask`, the Ethernet switch shall enter an inactive or low power mode if `EthSwtLowPowerModeSupport` is set to TRUE. If `EthSwtLowPowerModeSupport` is not defined or set to FALSE the Ethernet switch shall enter an active state.⌋*(SRS_BSW_00101)*

Note: The execution of this function may take a long time (e.g. port structure, VLAN configuration, internal Ethernet switch engine ... a.s.o.) and therefore cannot be called by EcuM or BswM. Instead it should be called e.g. by a background task (see `EthSwt_BackgroundTask`).

[SWS_EthSwt_00374] ⌈All Ethernet switch HW ports which are not configured as a EthSwtPort shall be switched off during initialization. This Ethernet switch HW ports shall never be switched on during runtime⌋*()*

[SWS_EthSwt_00375] ⌈All EthSwtPorts shall be set to `ETH_MODE_DOWN` during initialization.⌋*()*

[SWS_EthSwt_00016] ⌈The function `EthSwt_SwitchInit` shall check the access to the Ethernet Switch hardware, i.e. by trying to read or write registers during the configuration of the switch. If the access to the registers fails, the function shall raise the extended production error `ETHSWT_E_ACCESS` and return `E_NOT_OK`.⌋*(SRS_BSW_00386)*

Note: Access to the Ethernet Switch hardware is device dependent, e.g. access through the Ethernet Controller Mii, access through SPI, ... etc.

8.3.2 EthSwt_SetSwitchPortMode

[SWS_EthSwt_00018] ⌈

Service Name	EthSwt_SetSwitchPortMode

▽

△

Syntax	`Std_ReturnType EthSwt_SetSwitchPortMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_ModeType PortMode )`	
Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
	PortMode	ETH_MODE_DOWN: Disable the addressed Ethernet switch port at the given Ethernet switch ETH_MODE_ACTIVE: Enable the addressed Ethernet switch port at the given Ethernet switch ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST: Enable the addressed Ethernet switch port at the given Ethernet switch and request to trigger a wake-up on the network. (This could be used e.g. for Ethernet hardware which is compatible with the OA TC10)
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: The indexed switch port could not be set to Port Mode, or the function is called in state ETHSWT_STATE_UNINIT or ETHSWT_STATE_INIT.
Description	Enables/disables the indexed switch port	
Available via	EthSwt.h	

⌋*(SRS_ETH_00118)*

[SWS_EthSwt_00019] ⌈The function `EthSwt_SetSwitchPortMode` shall put the indexed port of the switch into the specified mode. If EthSwtPort references an EthTrcv then the function `EthTrcv_SetTransceiverMode` of the Ethernet Transceiver Driver shall additionally be called with the corresponding transceiver mode.⌋*(SRS_ETH_00118)*

[SWS_EthSwt_00396] ⌈When calling the function `EthSwt_SetSwitchPortMode` with mode `ETH_MODE_DOWN`, the EthSwt shall disable the Ethernet switch port directly for reduction of power consumption, if it is possible.⌋*()*

[SWS_EthSwt_00397] ⌈When calling the function `EthSwt_SetSwitchPortMode`, the function shall check the access to the Ethernet switch driver. If the check fails, the function shall raise the extended production error `ETHSWT_E_ACCESS` and return `E_NOT_OK`, otherwise pass the extended production error `ETHSWT_E_ACCESS` and return `E_OK`.⌋*()*

[SWS_EthSwt_00398] ⌈If EthSwtPort does not references an EthTrcv, EthSwt shall indicate a mode of the port by the API `EthIf_SwitchPortModeIndication` latest during the next `EthSwt_MainFunction`.⌋*(SRS_ETH_00118)*

[SWS_EthSwt_00022] ⌈The function EthSwt_SetSwitchPortMode shall be pre compile time configurable On/Off by the configuration parameter: EthSwt-SetSwitchPortModeApi.⌋*(SRS_BSW_00171)*

[SWS_EthSwt_00023] ⌈If the switch is already in the requested mode E_OK shall be returned and no development error shall be raised.⌋*(SRS_ETH_00118)*

8.3.3 EthSwt_GetSwitchPortMode

[SWS_EthSwt_00025] ⌈

Service Name	EthSwt_GetSwitchPortMode	
Syntax	`Std_ReturnType EthSwt_GetSwitchPortMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_ModeType* SwitchModePtr )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	SwitchModePtr	ETH_MODE_DOWN: The Ethernet switch port of the given Ethernet switch is disabled ETH_MODE_ACTIVE: The Ethernet switch port of the given Ethernet switch is enabled
Return value	Std_ReturnType	E_OK: success E_NOT_OK: The mode of the indexed switch port could not be obtained, or the function is called in state ETHSWT_STATE_ UNINIT or ETHSWT_STATE_INIT.
Description	Obtains the mode of the indexed switch port	
Available via	EthSwt.h	

⌋*(SRS_ETH_00118)*

[SWS_EthSwt_00026] ⌈The function EthSwt_GetSwitchPortMode shall read the mode of the indexed port of the switch. If EthSwtPort references an EthTrcv then the function shall additionally call the corresponding function EthTrcv_Get-TransceiverMode of the Ethernet Transceiver Driver.⌋*(SRS_ETH_00118)*

[SWS_EthSwt_00439] ⌈The function shall report the active mode always as ETH_MODE_ACTIVE, even though the previous requested (via EthSwt_SetSwitch-PortMode) mode was ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST.⌋*()*

[SWS_EthSwt_00399] ⌈If the obtained modes of the EthSwtPort and the EthTrcv are not aligned, the function EthSwt_GetSwitchPortMode shall raise the extended production error ETHSWT_E_SYNCPORT2PHY and return E_NOT_OK.

If EthTrcv_GetTransceiverMode returns E_NOT_OK, the EthSwt_GetSwitch-PortMode shall also return E_NOT_OK without raising an error.⌋*()*

[SWS_EthSwt_00400] ⌈If the function EthSwt_GetSwitchPortMode is called, the function shall check the access to the Ethernet Switch Driver. If the check fails, the function shall raise the extended production error ETHSWT_E_ACCESS and return E_NOT_OK, otherwise pass the production error ETHSWT_E_ACCESS and return E_OK.⌋*()*

[SWS_EthSwt_00029] ⌈The function EthSwt_GetSwitchPortMode shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetSwitchPortModeApi.⌋*(SRS_BSW_00171)*

8.3.4 EthSwt_StartSwitchPortAutoNegotiation

[SWS_EthSwt_00031] ⌈

Service Name	EthSwt_StartSwitchPortAutoNegotiation	
Syntax	`Std_ReturnType EthSwt_StartSwitchPortAutoNegotiation ( uint8 SwitchIdx, uint8 SwitchPortIdx )`	
Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: Automatic negotiation could not be started for the indexed switch port, or the function is called in state ETHSWT_STATE_UNINIT or ETHSWT_STATE_INIT.
Description	Starts the auto-negotiation of the indexed switch port	
Available via	EthSwt.h	

⌋*(SRS_ETH_00087)*

[SWS_EthSwt_00032] ⌈The function EthSwt_StartSwitchPortAutoNegotiation shall restart the automatic negotiation of the used transmission parameters of the referenced Ethernet transceiver driver by calling the function EthTrcv_StartAutoNegotiation.⌋*(SRS_ETH_00087)*

[SWS_EthSwt_00035] ⌈The function EthSwt_StartSwitchPortAutoNegotiation shall be pre compile time configurable On/Off by the configuration parameter: EthSwtStartSwitchPortAutoNegotiationApi.⌋*(SRS_BSW_00171)*

8.3.5 EthSwt_CheckWakeup

[SWS_EthSwt_91003] ⌈

Service Name	EthSwt_CheckWakeup	
Syntax	`Std_ReturnType EthSwt_CheckWakeup ( uint8 SwitchIdx )`	
Service ID [hex]	0x4c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: request to check for a wake-up is accepted E_NOT_OK: request to check for a wake-up is not accepted
Description	API is called by EthIf. The Ethernet switch driver request to check for a wake-up at all Ethernet switch ports which reference an EthTrcv. For those Ethernet switch ports the call is forwarded to the referenced EthTrcv. The function could be called in context of an interrupt service routine or on task level Note: Interrupt service routine consuming time has to be considered, since all EthSwtPorts of the maintained Ethernet switches has to be checked. Therefore the call is forwarded to the referred EthTrcv where the request to check for wake-up is stored. The check of the Ethernet hardware is done asynchronously in the context of the EthTrcv_MainFunction.	
Available via	EthSwt.h	

⌋(*SRS_Eth_00118*)

[SWS_EthSwt_00440] ⌈The function `EthSwt_CheckWakeup` shall iterate over the Ethernet switch ports of the indexed Ethernet switch and forward the call to `EthTrcv_CheckWakeup` for those Ethernet switch ports, which reference an EthTrcv.⌋(*SRS_ETH_00118*)

[SWS_EthSwt_00441] ⌈The function `EthSwt_CheckWakeup` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtCheckWakeupApi`⌋ (*SRS_BSW_00171*)

8.3.6 EthSwt_GetSwitchPortWakeupReason

[SWS_EthSwt_91040] ⌈

Service Name	EthSwt_GetSwitchPortWakeupReason	
Syntax	`Std_ReturnType EthSwt_GetSwitchPortWakeupReason ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_WakeupReasonType Reason )`	
Service ID [hex]	0x4b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch driver

▽

△

	SwitchPortIdx	Index of the Ethernet switch port index in the context of the Ethernet switch driver
Parameters (inout)	None	
Parameters (out)	Reason	Pointer to structure of least recent wakeup event, which was detected by the Ethernet switch port
Return value	Std_ReturnType	E_OK: Ethernet switch port wake up reason request has been accepted. E_NOT_OK: Ethernet switch port wake up reason request has not been accepted.
Description	This function obtains the wake up reasons of the the indexed Ethernet switch port by calling Eth Trcv_GetBusWuReason(...) of the referenced EthTrcv	
Available via	EthSwt.h	

⌋(*SRS_Eth_00107*)

[SWS_EthSwt_00442] ⌈The function `EthSwt_GetSwitchPortWakeupReason` shall read the current wake-up reason of the indexed Ethernet switch port by forwarding the call to `EthTrcv_GetBusWuReason` of the referenced EthTrcv. If the indexed Ethernet switch port has no reference to an EthTrcv, the function shall return `E_NOT_OK`.⌋ (*SRS_Eth_00107*)

[SWS_EthSwt_00443] ⌈The function `EthSwt_GetSwitchPortWakeupReason` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetSwitchPortWakeupReasonApi`⌋(*SRS_BSW_00171*)

8.3.7 EthSwt_GetLinkState

[SWS_EthSwt_00037] ⌈

Service Name	EthSwt_GetLinkState	
Syntax	`Std_ReturnType EthSwt_GetLinkState ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_LinkStateType* LinkStatePtr )`	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	LinkStatePtr	ETHTRCV_LINK_STATE_DOWN: Switch port is disconnected ETHTRCV_LINK_STATE_ACTIVE: Switch port is connected
Return value	Std_ReturnType	E_OK: success E_NOT_OK: Link state of the indexed switch port could not be obtained, or the function is called in state ETHSWT_STATE_UNINIT or ETHSWT_STATE_INIT.
Description	Obtains the link state of the indexed switch port	

▽

△

Available via	EthSwt.h

⌋(*SRS_ETH_00119*)

[SWS_EthSwt_00038] ⌈The function EthSwt_GetLinkState shall read the current (link) state of the indexed switch port. If the indexed Ethernet port references an Ethernet transceiver, the link state shall be obtained by calling the function EthTrcv_GetLinkState of the Ethernet Transceiver Driver. If the indexed Ethernet Switch port does not reference an Ethernet transceiver, the state shall be obtained from the MAC interface of the Switch port. If the MAC interface is not able to provide a link state (e.g. Ethernet hardware does not support a link state of the MAC interface), the API shall return the following state which is derived from the current mode:

- If the current mode of the indexed switch port is ETH_MODE_ACTIVE, then ETHTRCV_LINK_STATE_ACTIVE shall be returned
- If the current mode of the indexed switch port is ETH_MODE_DOWN, then ETHTRCV_LINK_STATE_DOWN shall be returned

⌋(*SRS_ETH_00118, SRS_ETH_00119*)

[SWS_EthSwt_00042] ⌈The function EthSwt_GetLinkState shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetLinkStateApi.⌋(*SRS_BSW_00171*)

8.3.8 EthSwt_GetBaudRate

[SWS_EthSwt_00044] ⌈

Service Name	EthSwt_GetBaudRate	
Syntax	`Std_ReturnType EthSwt_GetBaudRate ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_BaudRateType* BaudRatePtr )`	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	BaudRatePtr	ETHTRCV_BAUD_RATE_10MBIT: 10MBit connection ETHTRCV_BAUD_RATE_100MBIT: 100MBit connection ETHTRCV_BAUD_RATE_1000MBIT: 1000MBit connection ETHTRCV_BAUD_RATE_2500MBIT: 2500MBit connection

▽

△

Return value	Std_ReturnType	E_OK: success E_NOT_OK: Baud rate of the indexed switch port could not be obtained, or the function is called in state ETHSWT_STATE_UNINIT or ETHSWT_STATE_INIT.
Description	Obtains the baud rate of the indexed switch port	
Available via	EthSwt.h	

⌋(*SRS_ETH_00118*)

[SWS_EthSwt_00045] ⌈The function `EthSwt_GetBaudRate` shall read the current baud rate of the indexed switch port. If the indexed Ethernet port reference an Ethernet transceiver, the baud rate shall be obtained by the function `EthTrcv_GetBaudRate` of the Ethernet Transceiver Driver. If the indexed Ethernet Switch port does not reference an Ethernet transceiver, the baud rate shall be obtained from the MAC interface of the Switch port.⌋(*SRS_ETH_00118*)

[SWS_EthSwt_00049] ⌈The function `EthSwt_GetBaudRate` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetBaudRateApi`.⌋ (*SRS_BSW_00171*)

8.3.9 EthSwt_GetDuplexMode

[SWS_EthSwt_00051] ⌈

Service Name	EthSwt_GetDuplexMode	
Syntax	`Std_ReturnType EthSwt_GetDuplexMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_DuplexModeType* DuplexModePtr )`	
Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	DuplexModePtr	ETHTRCV_DUPLEX_MODE_HALF: half duplex connections ETHTRCV_DUPLEXMODE_FULL: full duplex connection
Return value	Std_ReturnType	E_OK: success E_NOT_OK: duplex mode of the indexed switch port could not be obtained, or the function is called in state ETHSWT_STATE_UNINIT or ETHSWT_STATE_INIT.
Description	Obtains the duplex mode of the indexed switch port	
Available via	EthSwt.h	

⌋(*SRS_ETH_00118*)

[SWS_EthSwt_00052] ⌈The function `EthSwt_GetDuplexMode` shall read the current duplex mode of the indexed switch port. If the indexed Ethernet port reference

an Ethernet transceiver, the duplex mode shall be obtained by calling the function EthTrcv_ GetDuplexMode of the Ethernet Transceiver Driver. If the indexed Ethernet Switch port does not reference an Ethernet transceiver, the duplex mode shall be obtained from the MAC interface of the Switch port.⌋(*SRS_ETH_00118*)

[SWS_EthSwt_00056] ⌈The function EthSwt_GetDuplexMode shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetDuplexModeApi.⌋(*SRS_BSW_00171*)

8.3.10 EthSwt_GetPortMacAddr

[SWS_EthSwt_00060] ⌈

Service Name	EthSwt_GetPortMacAddr	
Syntax	`Std_ReturnType EthSwt_GetPortMacAddr ( uint8 SwitchIdx, const uint8* MacAddrPtr, uint8* PortIdxPtr )`	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	MacAddrPtr	MAC-address for which a switch port is searched over which the node with this MAC-address can be reached.
Parameters (inout)	None	
Parameters (out)	PortIdxPtr	Pointer to the port index
Return value	Std_ReturnType	E_OK: success E_NOT_OK: multiple ports were found
Description	Obtains the port over which this MAC-address at the indexed switch can be reached. The result might be used for a DHCP-server which will need the port/MAC-resolution. If for the PortIdxPtr the maximal possible value (255) is returned the given MAC address cannot be reached via a port of this switch. If multiple ports were found the API returns E_NOT_OK.	
Available via	EthSwt.h	

⌋(*SRS_ETH_00087*)

[SWS_EthSwt_00061] ⌈The function EthSwt_GetPortMacAddr shall return the port index over which the given MAC-address is reachable within the indexed switch. If for the PortIdxPtr the maximal possible value (255) is returned the given MAC address cannot be reached via a port of this switch. If multiple ports were found the API returns E_NOT_OK.⌋(*SRS_ETH_00087*)

[SWS_EthSwt_00230] ⌈The function EthSwt_GetPortMacAddr shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetPortMacAddrApi.⌋(*SRS_BSW_00171*)

8.3.11 EthSwt_GetArlTable

[SWS_EthSwt_00111] ⌈

Service Name	EthSwt_GetArlTable	
Syntax	`Std_ReturnType EthSwt_GetArlTable ( uint8 switchIdx, uint16* numberOfElements, Eth_MacVlanType* arlTableListPointer )`	
Service ID [hex]	0x0a	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	switchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout)	numberOfElements	In: Maximum number of elements which can be written into the arlTable Out: Number of elements which are currently available in the EthSwitch module.
Parameters (out)	arlTableListPointer	Returns a pointer to the memory where the ARL table of the switch consisting of a list of structs with MAC-address, VLAN-ID and port shall be stored.
Return value	Std_ReturnType	E_OK: success E_NOT_OK: requested switchIdx is not valid or inactive
Description	Obtains the address resolution table of a switch and copies the list into a user provided buffer. The function will copy all or numberOfElements into the output list. If input value of numberOf Elements is 0 the function will not copy any data but only return the number of valid entries in the cache. arlTableListPointer may be NULL_PTR in this case.	
Available via	EthSwt.h	

⌋*(SRS_ETH_00087)*

[SWS_EthSwt_00228] ⌈The function `EthSwt_GetArlTable` shall provide a list of structs with MAC-address, VLAN-ID and port for the indexed switch.⌋*(SRS_ETH_00087)*

[SWS_EthSwt_00197] ⌈If the numberOfElements is greater 0x00, the arlTableList-Pointer shall be filled with up to numberOfElements elements. numberOfElements shall return the number of copied elements.⌋*(SRS_ETH_00087)*

[SWS_EthSwt_00235] ⌈The `EthSwt_GetArlTable` API shall return only the numberOfElements if the numberOfElements is set to 0x00. In this case no data will be copied and a NULLPTR can be used for the arlTableListPointer.⌋*(SRS_ETH_00087)*

[SWS_EthSwt_00229] ⌈The function `EthSwt_GetArlTable` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetArlTableApi`.⌋*(SRS_BSW_00171)*

8.3.12 EthSwt_GetCounterValues

[SWS_EthSwt_00231] ⌈

Service Name	EthSwt_GetCounterValues	
Syntax	`Std_ReturnType EthSwt_GetCounterValues ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_CounterType* CounterPtr )`	
Service ID [hex]	0x0c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	CounterPtr	counter values according to IETF RFC 1757, RFC 1643 and RFC 2233.
Return value	Std_ReturnType	E_OK: success E_NOT_OK: counter values read failure
Description	Reads a list with drop counter values of the corresponding port of the switch. The meaning of these values is described at Eth_CounterType.	
Available via	EthSwt.h	

⌋(*SRS_Eth_00128*)

[SWS_EthSwt_00106] ⌈ `EthSwt_GetCounterValues` shall read a list with drop counter values of the corresponding port of the switch. The meaning of these values is described at `Eth_CounterType`.⌋(*SRS_ETH_00128*)

8.3.13 EthSwt_GetRxStats

[SWS_EthSwt_00198] ⌈

Service Name	EthSwt_GetRxStats	
Syntax	`Std_ReturnType EthSwt_GetRxStats ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_RxStatsType* RxStats )`	
Service ID [hex]	0x0d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	RxStats	List of values according to IETF RFC 2819 (Remote Network Monitoring Management Information Base)
Return value	Std_ReturnType	E_OK: success E_NOT_OK: drop counter could not be obtained
Description	Returns a list of statistic counters defined with Eth_RxTatsType. The majority of these Counters are derived from the IETF RFC2819.	

▽

△

Available via	EthSwt.h

⌋*(SRS_Eth_00128)*

[SWS_EthSwt_00199] ⌈`EthSwt_GetRxStats` shall return a list of statistic counters defined with `Eth_RxStatsType`. The majority of these Counters are derived from the IETF RFC2819.⌋*(SRS_ETH_00128)*

[SWS_EthSwt_00202] ⌈The function `EthSwt_GetRxStats` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetRxStatsApi`.⌋ *(SRS_BSW_00171)*

8.3.14 EthSwt_GetTxStats

[SWS_EthSwt_91001] ⌈

Service Name	EthSwt_GetTxStats	
Syntax	`Std_ReturnType EthSwt_GetTxStats ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_TxStatsType* TxStats )`	
Service ID [hex]	0x20	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	TxStats	List of values to read statistic values for transmission.
Return value	Std_ReturnType	E_OK: success E_NOTOK: Tx-statistics could not be obtained
Description	Returns the list of Transmission Statistics out of IETF RFC1213 defined with Eth_TxStatsType, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.	
Available via	EthSwt.h	

⌋*(SRS_Eth_00128)*

[SWS_EthSwt_00372] ⌈`EthSwt_GetTxStats` shall return the list of Transmission Statistics out of IETF RFC1213 defined with `Eth_TxStatsType`, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.⌋ *(SRS_ETH_00128)*

[SWS_EthSwt_00362] ⌈The function `EthSwt_GetTxStats` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetTxStatsApi`.⌋ *(SRS_BSW_00171)*

8.3.15 EthSwt_GetTxErrorCounterValues

[SWS_EthSwt_91000] ⌈

Service Name	EthSwt_GetTxErrorCounterValues	
Syntax	`Std_ReturnType EthSwt_GetTxErrorCounterValues ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_TxErrorCounterValuesType* TxStats )`	
Service ID [hex]	0x21	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Drive
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	TxStats	List of values to read statistic error counter values for transmission.
Return value	Std_ReturnType	E_OK: success, E_NOTOK: Tx-statistics could not be obtained
Description	Returns the list of Transmission Error Counters out of IETF RFC1213 and RFC1643 defined with Eth_TxErrorCounterValuesType, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.	
Available via	EthSwt.h	

⌋*(SRS_Eth_00128)*

[SWS_EthSwt_00373] ⌈`EthSwt_GetTxErrorCounterValues` returns the list of Transmission Error Counters out of IETF RFC1213 and RFC1643 defined with `Eth_TxErrorCounterValuesType`, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.⌋*(SRS_ETH_00128)*

[SWS_EthSwt_00370] ⌈The function `EthSwt_GetTxErrorCounterValues` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetTxErrorCounterValuesApi`.⌋*(SRS_BSW_00171)*

8.3.16 EthSwt_GetSwitchReg

[SWS_EthSwt_00206] ⌈

Service Name	EthSwt_GetSwitchReg
Syntax	`Std_ReturnType EthSwt_GetSwitchReg ( uint8 SwitchIdx, uint32 page, uint32 register, uint32* registerContent )`
Service ID [hex]	0x0e

▽

△

Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	page	Address of a register page
	register	Address of a register
Parameters (inout)	None	
Parameters (out)	registerContent	Content of the addresses register
Return value	Std_ReturnType	E_OK: success E_NOT_OK: drop counter could not be obtained
Description	Generic API for reading the content of a switch register	
Available via	EthSwt.h	

⌋(*SRS_Eth_00120*)

[SWS_EthSwt_00207] ⌈The function `EthSwt_GetSwitchReg` shall read the content of a switch register.⌋(*SRS_Eth_00120*)

[SWS_EthSwt_00210] ⌈The function `EthSwt_GetSwitchReg` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetSwitchRegApi`.⌋(*SRS_BSW_00171*)

8.3.17 EthSwt_SetSwitchReg

[SWS_EthSwt_00211] ⌈

Service Name	EthSwt_SetSwitchReg	
Syntax	`Std_ReturnType EthSwt_SetSwitchReg ( uint8 SwitchIdx, uint32 page, uint32 register, uint32 registerContent )`	
Service ID [hex]	0x0f	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	page	Address of a register page
	register	Address of a register
	registerContent	Content of the addresses register
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: drop counter could not be obtained
Description	Generic API for writing the content of a switch register	
Available via	EthSwt.h	

⌋(*SRS_Eth_00120*)

[SWS_EthSwt_00212] ⌈The function EthSwt_SetSwitchReg shall write the content of a switch register.⌋ *(SRS_Eth_00120)*

[SWS_EthSwt_00215] ⌈The function EthSwt_SetSwitchReg shall be pre compile time configurable On/Off by the configuration parameter: EthSwtSetSwitchRegApi.⌋ *(SRS_BSW_00171)*

8.3.18 EthSwt_ReadTrcvRegister

[SWS_EthSwt_00216] ⌈

Service Name	EthSwt_ReadTrcvRegister	
Syntax	`Std_ReturnType EthSwt_ReadTrcvRegister ( uint8 SwitchIdx, uint8 SwitchPortIdx, uint8 RegIdx, uint16* RegValPtr )`	
Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
	RegIdx	Index of the register
Parameters (inout)	None	
Parameters (out)	RegValPtr	Pointer to the register content
Return value	Std_ReturnType	E_OK: success E_NOT_OK: Content of the transceiver could not be obtained, or the function is called in state ETHSWT_STATE_UNINIT or ETHSWT_STATE_INIT.
Description	Generic API for reading the content of a transceiver register	
Available via	EthSwt.h	

⌋ *(SRS_Eth_00120)*

[SWS_EthSwt_00217] ⌈The function EthSwt_ReadTrcvRegister shall read the specified transceiver register through the MII or SPI of the indexed switch port.⌋ *(SRS_ETH_00118, SRS_ETH_00120)*

[SWS_EthSwt_00220] ⌈The function EthSwt_ReadTrcvRegister shall be pre compile time configurable On/Off by the configuration parameter: EthSwtReadTrcvRegisterApi.⌋ *(SRS_BSW_00171)*

8.3.19 EthSwt_WriteTrcvRegister

[SWS_EthSwt_00221] ⌈

Service Name	EthSwt_WriteTrcvRegister	
Syntax	`Std_ReturnType EthSwt_WriteTrcvRegister ( uint8 SwitchIdx, uint8 SwitchPortIdx, uint8 RegIdx, uint16 RegVal )`	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
	RegIdx	Index of the register
	RegVal	Content for the indexed register
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: Content given by RegVal could not be written to the given register (RegIdx) of the transceiver, or the function is called in state ETHSWT_STATE_UNINIT or ETHSWT_STATE_INIT.
Description	Generic API for writing the content of a transceiver register	
Available via	EthSwt.h	

⌋(*SRS_Eth_00120*)

[SWS_EthSwt_00222] ⌈The function `EthSwt_WriteTrcvRegister` shall write the specified transceiver register through the MII or SPI of the indexed switch port.⌋(*SRS_ETH_00118*, *SRS_ETH_00120*)

[SWS_EthSwt_00225] ⌈The function `EthSwt_WriteTrcvRegister` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtWriteTrcvRegisterApi`.⌋(*SRS_BSW_00171*)

8.3.20 EthSwt_EnableVlan

[SWS_EthSwt_00172] ⌈

Service Name	EthSwt_EnableVlan
Syntax	`Std_ReturnType EthSwt_EnableVlan ( uint8 SwitchIdx, uint8 SwitchPortIdx, uint16 VlanId, boolean Enable )`
Service ID [hex]	0x12
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
	VlanId	VLAN-ID to a preconfigured configuration on the given ingress port
	Enable	1 = VLAN-configuration enabled 0 = VLAN-configuration disabled (frames with given VLAN-ID will be dropped)
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: buffer level could not be obtained
Description	Enables or disables a pre-configured VLAN at a certain port of a switch.	
Available via	EthSwt.h	

⌋(*SRS_ETH_00121*, *SRS_ETH_00114*)

[SWS_EthSwt_00173] ⌈The function `EthSwt_EnableVlan` shall enable or disable a pre-configured VLAN at a certain port of a switch.⌋(*SRS_ETH_00121*, *SRS_ETH_00114*)

[SWS_EthSwt_00177] ⌈The function `EthSwt_EnableVlan` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtEnableVlanApi`.⌋(*SRS_BSW_00171*)

8.3.21 EthSwt_StoreConfiguration

[SWS_EthSwt_00086] ⌈

Service Name	EthSwt_StoreConfiguration	
Syntax	`Std_ReturnType EthSwt_StoreConfiguration ( uint8 SwitchIdx )`	
Service ID [hex]	0x13	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request to persistently store the MAC/Port table was accepted E_NOT_OK: Request to persistently store the MAC/Port table was not accepted
Description	Trigger the storage/reset of the configuration of the learned MAC/Port tables of a switch in a persistent manner and will be used by e.g. CDD.	
Available via	EthSwt.h	

⌋(*SRS_ETH_00087*, *SRS_ETH_00122*)

[SWS_EthSwt_00087] ⌈The function EthSwt_StoreConfiguration shall request to store the configuration of the learned MAC/Port tables of a switch in a persistent manner. This can be done in two ways: 1.) Reading out the parameters and storing them in the NV-RAM of the host CPU using the NV-RAM manager. 2.) Advising the switch to store the configuration data in its local NV-RAM.

In both alternatives <EthSwtPersistentConfigurationResultCallback> shall be invoked if EthSwtPersistentConfigurationResultCallback is configured. In case of storage to switch local NV-RAM, JobResult shall be set to NVM_REQ_OK to indicate success or to NVM_REQ_NOT_OK to indicate failure.⌋*(SRS_ETH_00087, SRS_ETH_00122)*

[SWS_EthSwt_00090] ⌈The function EthSwt_StoreConfiguration shall be pre compile time configurable On/Off by the configuration parameter: EthSwtStoreConfigurationApi.⌋*(SRS_BSW_00171)*

8.3.22 EthSwt_ResetConfiguration

[SWS_EthSwt_00091] ⌈

Service Name	EthSwt_ResetConfiguration	

Syntax	`Std_ReturnType EthSwt_ResetConfiguration (`
	`  uint8 SwitchIdx`
	`)`	

Service ID [hex]	0x14	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request to persistently reset the MAC/Port table was accepted E_NOT_OK: Request to persistently reset the MAC/Port table was not accepted
Description	The function shall request to reset and store the configuration of the learned MAC/Port tables of a Ethernet switch in a persistent manner. This could be used by e.g. a CDD.	
Available via	EthSwt.h	

⌋*(SRS_ETH_00087, SRS_ETH_00122)*

[SWS_EthSwt_00092] ⌈The function EthSwt_ResetConfiguration shall request to reset the configuration of the learned MAC/Port tables of a switch in a persistent manner. This can be done in two ways: 1.) Overwriting the learned parameters in the NV-RAM of the host CPU with preconfigured default values. 2.) Advising the switch to reset the learned configuration data in its local NV-RAM.

In both alternatives <EthSwtPersistentConfigurationResultCallback> shall be invoked if EthSwtPersistentConfigurationResultCallback is configured. In case of storage to switch local NV-RAM, JobResult shall be set to NVM_REQ_OK to indicate

success or to NVM_REQ_NOT_OK to indicate failure.⌋(*SRS_ETH_00122*, *SRS_ETH_00087*)

[SWS_EthSwt_00095] ⌈The function EthSwt_ResetConfiguration shall be pre compile time configurable On/Off by the configuration parameter: EthSwtResetConfigurationApi.⌋(*SRS_BSW_00171*)

8.3.23 EthSwt_SetMacLearningMode

[SWS_EthSwt_00182] ⌈

Service Name	EthSwt_SetMacLearningMode	
Syntax	`Std_ReturnType EthSwt_SetMacLearningMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthSwt_MacLearningType MacLearningMode )`	
Service ID [hex]	0x15	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
	MacLearningMode	Defines whether MAC addresses shall be learned and if they shall be learned in software or hardware.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: configuration could be persistently reset
Description	Sets the MAC learning mode in one of the tree modes: 1.) HW learning enabled, 2.) Hardware learning disabled, 3.) Software learning enabled. Note: This feature is hardware dependent, i.e. the switch hardware needs to support the different learning modes.	
Available via	EthSwt.h	

⌋(*SRS_ETH_00087*, *SRS_ETH_00122*)

[SWS_EthSwt_00183] ⌈The function EthSwt_SetMacLearningMode shall set the MAC learning mode according to EthSwt_MacLearningType.⌋(*SRS_ETH_00122*, *SRS_ETH_00087*)

Note: This feature is hardware dependent, i.e. the switch hardware needs to support the different modes.

[SWS_EthSwt_00186] ⌈The function EthSwt_SetMacLearningMode shall be pre compile time configurable On/Off by the configuration parameter: EthSwtSetMacLearningModeApi.⌋(*SRS_BSW_00171*)

8.3.24 EthSwt_GetMacLearningMode

[SWS_EthSwt_00187] ⌈

Service Name	EthSwt_GetMacLearningMode	
Syntax	`Std_ReturnType EthSwt_GetMacLearningMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthSwt_MacLearningType* MacLearningMode )`	
Service ID [hex]	0x16	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	MacLearningMode	Defines whether MAC addresses shall be learned and if they shall be learned in software or hardware.
Return value	Std_ReturnType	E_OK: success E_NOT_OK: configuration could be persistently reset
Description	Returns the MAC learning mode, i.e. 1.) HW learning enabled, 2.) Hardware learning disabled, 3.) Software learning enabled. Note: This feature is hardware dependent, i.e. the switch hardware needs to support the different learning modes	
Available via	EthSwt.h	

⌋*(SRS_ETH_00087)*

[SWS_EthSwt_00188] ⌈The function `EthSwt_GetMacLearningMode` shall return the MAC learning mode according to `EthSwt_MacLearningType`.⌋*(SRS_ETH_00087)*

Note: This feature is hardware dependent, i.e. the switch hardware needs to support the different learning modes.

[SWS_EthSwt_00191] ⌈The function `EthSwt_GetMacLearningMode` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetMacLearningModeApi`.⌋*(SRS_BSW_00171)*

8.3.25 EthSwt_NvmSingleBlockCallback

[SWS_EthSwt_00125] ⌈

Service Name	EthSwt_NvmSingleBlockCallback
Syntax	`Std_ReturnType EthSwt_NvmSingleBlockCallback ( NvM_BlockRequestType BlockRequest, NvM_RequestResultType JobResult )`
Service ID [hex]	0x17

▽

△

Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	BlockRequest	The request type (read, write, ... etc.) of the previous processed block job
	JobResult	Covers the job result of the previous processed single block job.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: Callback function has not been processed successfully
Description	Function will be called by the NVRAMManager after the switch configuration has been stored or resetted.	
Available via	EthSwt_NvM.h	

⌋(*SRS_ETH_00087*, *SRS_ETH_00122*)

[SWS_EthSwt_00126] ⌈The function `EthSwt_NvmSingleBlockCallback` shall be called by the NVRAMManager [13] after the switch configuration has been stored or reset in the the NV RAM.⌋(*SRS_ETH_00122*, *SRS_ETH_00087*)

[SWS_EthSwt_00196] ⌈The function `EthSwt_NvmSingleBlockCallback` shall call the function <user>_PersistentConfigurationResult to provide the `JobResult` to the caller of `EthSwt_StoreConfiguration` or `EthSwt_ResetConfiguration`.⌋ (*SRS_ETH_00122*, *SRS_ETH_00087*)

[SWS_EthSwt_00127] ⌈The function `EthSwt_NvmSingleBlockCallback` shall always return `E_OK` according to `SWS_NvM_00368`.⌋(*SRS_ETH_00122*, *SRS_ETH_00087*)

[SWS_EthSwt_00128] ⌈The function `EthSwt_NvmSingleBlockCallback` shall raise a development error if the `JobResult` equals `NVM_REQ_NOT_OK`, i.e. the write request has been finished unsuccessfully.⌋(*SRS_BSW_00369*, *SRS_ETH_00458*)

Note: Please note that a production error at this point is not necessary because the NvM will raise also a production error if the write to NV RAM was not successful.

[SWS_EthSwt_00129] ⌈The function `EthSwt_NvmSingleBlockCallback` shall be pre compile time configurable On/Off by the existence of the container `EthSwtNvm`.⌋ (*SRS_BSW_00171*)

8.3.26 EthSwt_GetVersionInfo

[SWS_EthSwt_00058] ⌈

Service Name	EthSwt_GetVersionInfo

▽

△

Syntax	`void EthSwt_GetVersionInfo ( Std_VersionInfoType* VersionInfoPtr )`	
Service ID [hex]	0x18	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	VersionInfoPtr	Pointer to where to store the version information of this module.
Return value	None	
Description	Returns the version information of this module.	
Available via	EthSwt.h	

⌋(*SRS_BSW_00171*)

[SWS_EthSwt_00124] ⌈The function `EthSwt_GetVersionInfo` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtVersionInfoApi`.⌋(*SRS_BSW_00171*)

8.3.27 EthSwt_EthRxProcessFrame

[SWS_EthSwt_91004] ⌈

Service Name	EthSwt_EthRxProcessFrame	
Syntax	`Std_ReturnType EthSwt_EthRxProcessFrame ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, uint8** DataPtr, uint16* LengthPtr, boolean* IsMgmtFrameOnlyPtr )`	
Service ID [hex]	0x23	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	DataPtr	IN: Pointer to the position of the EtherType of a common Ethernet frame OUT: Pointer to the position of the EtherType in the management frame
	LengthPtr	IN: Pointer to the length of the frame received OUT: Pointer to the length decreased by the management information length.

▽

△

Parameters (out)	IsMgmtFrameOnlyPtr	Information about the kind of frame FALSE: Frame is not only for management purpose, but also for normal communication. TRUE: Frame is only for management purpose and must not be processed in common receive process
Return value	Std_ReturnType	E_OK: Frame successfully processed E_NOT_OK: Frame processing failed
Description	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in EthSwt_EthRxFinishedIndication().	
Available via	EthSwt_Eth.h	

⌋(*SRS_Eth_00125*)

[SWS_EthSwt_00249] ⌈The function `EthSwt_EthRxProcessFrame` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi` .⌋(*SRS_BSW_00171*)

8.3.28 EthSwt_EthRxFinishedIndication

[SWS_EthSwt_91005] ⌈

Service Name	EthSwt_EthRxFinishedIndication	
Syntax	`Std_ReturnType EthSwt_EthRxFinishedIndication ( uint8 CtrlIdx, Eth_BufIdxType BufIdx )`	
Service ID [hex]	0x24	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Frame successfully processed E_NOT_OK: Frame processing failed
Description	Indication for a finished receive process for a specific Ethernet frame, which results in providing the management information retrieved during EthSwt_EthRxProcessFrame().	
Available via	EthSwt_Eth.h	

⌋(*SRS_Eth_00125*)

[SWS_EthSwt_00253] ⌈The function `EthSwt_EthRxFinishedIndication` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi` .⌋(*SRS_BSW_00171*)

8.3.29 EthSwt_EthTxPrepareFrame

[SWS_EthSwt_91006] ⌈

Service Name	EthSwt_EthTxPrepareFrame	
Syntax	`Std_ReturnType EthSwt_EthTxPrepareFrame ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, uint8** DataPtr, uint16* LengthPtr )`	
Service ID [hex]	0x25	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	DataPtr	IN: Pointer to the position of the EtherType of a common Ethernet frame OUT: Pointer to the position of the EtherType in the management frame
	LengthPtr	IN: Pointer to the length of the buffer without management information OUT: Pointer to the modified length needed for buffer and management information
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Frame successfully prepared E_NOT_OK: Frame preparation failed
Description	Prepares the Ethernet frame for common Ethernet communication (frame shall be handled by switch according to the common address resolution behavior) and stores the information for processing of EthSwt_EthTxFinishedIndication().	
Available via	EthSwt_Eth.h	

⌋*(SRS_Eth_00125)*

[SWS_EthSwt_00257] ⌈The function `EthSwt_EthTxPrepareFrame` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi` .⌋*(SRS_BSW_00171)*

8.3.30 EthSwt_EthTxAdaptBufferLength

[SWS_EthSwt_91007] ⌈

Service Name	EthSwt_EthTxAdaptBufferLength
Syntax	`void EthSwt_EthTxAdaptBufferLength ( uint16* LengthPtr )`
Service ID [hex]	0x26
Sync/Async	Synchronous

▽

△

Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	LengthPtr	IN: Pointer to the length of the buffer without management information. OUT: Pointer to the modified length needed for buffer and management information.
Parameters (out)	None	
Return value	None	
Description	Modifies the buffer length to be able to insert management information.	
Available via	EthSwt_Eth.h	

⌋(*SRS_Eth_00125*)

[SWS_EthSwt_00261] ⌈The function `EthSwt_EthTxAdaptBufferLength` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi` .⌋(*SRS_BSW_00171*)

8.3.31 EthSwt_SetMgmtInfo

[SWS_EthSwt_91008] ⌈

Service Name	EthSwt_SetMgmtInfo	
Syntax	`Std_ReturnType EthSwt_SetMgmtInfo ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, const EthSwt_MgmtInfoType* MgmtInfoPtr )`	
Service ID [hex]	0x27	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
	MgmtInfoPtr	Pointer to the management information
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Management infos successfully set E_NOT_OK: Setting of management infos failed
Description	Extends the Ethernet frame prepared previously by EthSwt_EthTxPrepareFrame() with the management information to achieve transmission only on specific ports.	
Available via	EthSwt.h	

⌋(*SRS_Eth_00125*)

[SWS_EthSwt_00264] ⌈The function `EthSwt_SetMgmtInfo` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi` .⌋(*SRS_BSW_00171*)

8.3.32 EthSwt_EthTxProcessFrame

[SWS_EthSwt_91009] ⌈

Service Name	EthSwt_EthTxProcessFrame	
Syntax	`Std_ReturnType EthSwt_EthTxProcessFrame ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, uint8** DataPtr, uint16* LengthPtr )`	
Service ID [hex]	0x28	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	DataPtr	IN: Pointer to the position of the EtherType of a common Ethernet frame OUT: Pointer to the position of the EtherType in the management frame
	LengthPtr	IN: Pointer to the length of the received frame OUT: Pointer to the length decreased by the management information length
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Frame successfully processed E_NOT_OK: Frame processing failed
Description	Function inserts management information into the Ethernet frame.	
Available via	EthSwt_Eth.h	

⌋*(SRS_Eth_00125)*

[SWS_EthSwt_00268] ⌈The function `EthSwt_EthTxProcessFrame` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi` .⌋*(SRS_BSW_00171)*

8.3.33 EthSwt_EthTxFinishedIndication

[SWS_EthSwt_91010] ⌈

Service Name	EthSwt_EthTxFinishedIndication
Syntax	`Std_ReturnType EthSwt_EthTxFinishedIndication ( uint8 CtrlIdx, Eth_BufIdxType BufIdx )`
Service ID [hex]	0x29
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Frame successfully processed E_NOT_OK: Frame processing failed
Description	Indication for a finished transmit process for a specific Ethernet frame.	
Available via	EthSwt_Eth.h	

⌋(*SRS_Eth_00125*)

[SWS_EthSwt_00273] ⌈The function `EthSwt_EthTxFinishedIndication` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi` .⌋(*SRS_BSW_00171*)

8.3.34 EthSwt_PortEnableTimeStamp

[SWS_EthSwt_91028] ⌈

Service Name	EthSwt_PortEnableTimeStamp	
Syntax	`Std_ReturnType EthSwt_PortEnableTimeStamp ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, EthSwt_MgmtInfoType* MgmtInfoPtr )`	
Service ID [hex]	0x40	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
	MgmtInfoPtr	Management information including SwitchIdx and SwitchPortIdx
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Time stamping on egress successfully enabled E_NOT_OK: Enabling of time stamping on egress has been failed
Description	Activates egress time stamping on a dedicated message object on a dedicated port of a Switch if EthSwtPortTimeStampSupport is set to TRUE for this port. The selective activation of dedicated message objects for time stamping reduces the number of notification calls only to the required calls. Some HW does store once the egress time stamp marker and some HW needs it always before transmission. There will be no disabled functionality, due to the fact, that the message type is always "time stamped" by network design.	
Available via	EthSwt.h	

⌋(*SRS_Eth_00125*)

[SWS_EthSwt_00379] ⌈The function `EthSwt_PortEnableTimeStamp` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtGlobalTimeSupportApi` .⌋(*SRS_BSW_00171*)

8.3.35 EthSwt_VerifyConfig

[SWS_EthSwt_91012] ⌈

Service Name	EthSwt_VerifyConfig	

Syntax	`Std_ReturnType EthSwt_VerifyConfig (`
	`  uint8 SwitchIdx,`
	`  boolean* Result`
	`)`	

Service ID [hex]	0x31	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout)	None	
Parameters (out)	Result	Result of verification, TRUE: configureation verified ok, FALSE: configuraton values found corrupted
Return value	Std_ReturnType	E_OK: Configuration verificaton succeeded, E_NOT_OK: Configuration verification not succeeded.
Description	Verifies the Switch Configuration depending on the HW-Architecture, HW-capability and the intended accuracy of this verification.	
Available via	EthSwt.h	

⌋*(SRS_ETH_00126)*

[SWS_EthSwt_00287] ⌈The function `EthSwt_VerifyConfig` shall be compile time configurable On/Off by the configuration parameter: `EthSwtVerifyConfigApi`.⌋ *(SRS_BSW_00171)*

8.3.36 EthSwt_SetForwardingMode

[SWS_EthSwt_91013] ⌈

Service Name	EthSwt_SetForwardingMode	

Syntax	`Std_ReturnType EthSwt_SetForwardingMode (`
	`  uint8 SwitchIdx,`
	`  boolean mode`
	`)`	

Service ID [hex]	0x32	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	mode	True Forewarding enabled, False Forwarding disabled
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: stopping of frame forwarding succeeded, E_NOT_OK: stopping of frame forwarding not succeeded.

▽

△

Description	Configures switch to start or stop forwarding for all ports. This API call may be used during switch configuration verification.
Available via	EthSwt.h

⌋*(SRS_ETH_00126)*

[SWS_EthSwt_00291] ⌈The function `EthSwt_SetForwardingMode` shall be compile time configurable On/Off by the configuration parameter: `EthSwtSetForwardingModeApi`.⌋*(SRS_BSW_00171)*

8.3.37 EthSwt_GetPortSignalQuality

[SWS_EthSwt_91014] ⌈

Service Name	EthSwt_GetPortSignalQuality	
Syntax	`Std_ReturnType EthSwt_GetPortSignalQuality ( uint8 SwitchIdx, uint8 PortIdx, uint32* SignalQualityPtr )`	
Service ID [hex]	0x33	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	SignalQualityPtr	Pointer to the memory where the signal quality shall be stored.
Return value	Std_ReturnType	E_OK: signal quality could be read. E_NOT_OK: signal quality could not be read (i.e. no Ethernet transceiver is available for this Ethernet switch port)
Description	The function retrieves the signal quality of the link of the indexed Ethernet switch port. If no transceiver is referenced the signal quality shall be set to 0xFFFFFFFF.	
Available via	EthSwt.h	

⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00293] ⌈The function `EthSwt_GetPortSignalQuality` shall obtain the signal quality by calling the function `EthTrcv_GetPhySignalQuality` of the referenced Ethernet Transceiver Driver. If the current signal quality is not available, the signal quality shall be set to `0xFFFFFFFF`.⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00297] ⌈The function `EthSwt_GetPortSignalQuality` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetPortSignalQualityApi`.⌋*(SRS_BSW_00171)*

8.3.38 EthSwt_GetPortIdentifier

[SWS_EthSwt_91015] ⌈

Service Name	EthSwt_GetPortIdentifier	

Syntax	`Std_ReturnType EthSwt_GetPortIdentifier (`
	`  uint8 SwitchIdx,`
	`  uint8 PortIdx,`
	`  uint32* OrgUniqueIdPtr,`
	`  uint8* ModelNrPtr,`
	`  uint8* RevisionNrPtr`
	`)`	

Service ID [hex]	0x34	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	OrgUniqueIdPtr	Pointer to the memory where the Organizationally Unique Identifier (OUI) shall be stored.
	ModelNrPtr	Pointer to the memory where the Manufacturer's Model Number shall be stored.
	RevisionNrPtr	Pointer to the memory where the Revision Number shall be stored.
Return value	Std_ReturnType	E_OK: organizationally unique identifier of the Ethernet transceiver could be read. E_NOT_OK: organizationally unique identifier of the Ethernet transceiver could not be obtained (i.e. OUI is not available).
Description	This function retrieves the OUI (24 bit) of the indexed Ethernet switch port.	
Available via	EthSwt.h	

⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00299] ⌈The function `EthSwt_GetPortIdentifier` shall return the value of the organizationally unique identifier (OUI 24 bit) of the indexed Ethernet switch port that is connected to the indexed Ethernet switch. It shall set the 8 most significant bits of the OUI to 0xFFxxxxxx. If the Ethernet switch port references an Ethernet transceiver, the function shall obtain the OUI by calling the function `EthTrcv_GetPhyIdentifier` and set the 8 most significant bits of the OUI to 0x00xxxxxx.⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00394] ⌈If neither the Ethernet switch port nor the Ethernet Transceiver Driver can provide an OUI the function `EthSwt_GetPortIdentifier` shall return `E_NOT_OK`.⌋*()*

[SWS_EthSwt_00303] ⌈The function `EthSwt_GetPortIdentifier` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetPortIdentifierApi`.⌋*(SRS_BSW_00171)*

8.3.39 EthSwt_GetSwitchIdentifier

[SWS_EthSwt_91016] ⌈

Service Name	EthSwt_GetSwitchIdentifier	
Syntax	`Std_ReturnType EthSwt_GetSwitchIdentifier ( uint8 SwitchIdx, uint32* OrgUniqueIdPtr )`	
Service ID [hex]	0x35	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout)	None	
Parameters (out)	OrgUniqueIdPtr	Pointer to the memory where the Organizationally Unique Identifier shall be stored.
Return value	Std_ReturnType	E_OK: organizationally unique identifier of the Ethernet switch could be read. E_NOT_OK: organizationally unique identifier of the Ethernet switch could not be read (i.e. no OUI is available for this Ethernet switch)
Description	Obtain the Organizationally Unique Identifier that is given by the IEEE of the indexed Ethernet switch. This function shall provide the OUI of Ethernet switch. The OUI has a size of 24 bit. If a ethernet switch can provide the OUI the 8 most significant bits of the OUI shall be set to 0x00xxxxxx. If a Ethernet switch can not provide the OUI the 8 most significant bits of the OUI shall be set to 0xFFxxxxxx.	
Available via	EthSwt.h	

⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00305] ⌈The function `EthSwt_GetSwitchIdentifier` shall return the value of the organizationally unique identifier of the indexed Ethernet switch.⌋ *(SRS_Eth_00123)*

[SWS_EthSwt_00308] ⌈The function `EthSwt_GetSwitchIdentifier` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetSwitchIdentifierApi`.⌋*(SRS_BSW_00171)*

8.3.40 EthSwt_WritePortMirrorConfiguration

[SWS_EthSwt_91018] ⌈

Service Name	EthSwt_WritePortMirrorConfiguration
Syntax	`Std_ReturnType EthSwt_WritePortMirrorConfiguration ( uint8 MirroredSwitchIdx, const EthSwt_PortMirrorCfgType* PortMirrorConfigurationPtr )`
Service ID [hex]	0x36
Sync/Async	Synchronous

▽

△

Reentrancy	Non Reentrant	
Parameters (in)	MirroredSwitchIdx	Index of the switch within the context of the Ethernet Switch Driver, where the Ethernet switch port is located, that has to be mirrored
	PortMirrorConfiguration Ptr	Pointer of the port configuration, which shall be stored in a shadow buffer in the Ethernet switch driver
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: the port mirror configuration for the indexed Ethernet switch port was written. E_NOT_OK: the port mirror configuration for the indexed Ethernet switch port was not written. (i.e. indexed ethernet switch is not available) ETHSWT_PORT_MIRRORING_CONFIGURATION_NOT_SUPPORTED: port mirroring configuration is not supported by Ethernet switch driver or by the Ethernet switch hardware
Description	Store the given port mirror configuration in a shadow buffer in the Ethernet switch driver for the given MirroredSwitchIdx.	
Available via	EthSwt.h	

⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00309] ⌈The function `EthSwt_WritePortMirrorConfiguration` shall store the port mirror configuration of the given `MirroredSwitchIdx` in a shadow buffer. The `MirroredSwitchIdx` shall be used to identify the port mirror configuration within the Ethernet switch driver.⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00312] ⌈The function `EthSwt_WritePortMirrorConfiguration` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtWritePortMirrorConfigurationApi`.⌋*(SRS_BSW_00171)*

[SWS_EthSwt_00424] ⌈The function shall return with `ETHSWT_PORT_MIRRORING_CONFIGURATION_NOT_SUPPORTED`, if the port mirroring configuration is not supported by the Ethernet switch driver or by the Ethernet switch hardware, e.g.:

- the configured mirrored traffic direction (see [SWS_EthSwt_91017] "TrafficDirectionIngressBitMask" and "TrafficDirectionEgressBitMask") for ingress and egress traffic of the same port is not supported
- mirrored ports and capture ports, respectively, are not available within the Ethernet switch driver

⌋*(SRS_Eth_00123)*

8.3.41 EthSwt_ReadPortMirrorConfiguration

[SWS_EthSwt_91019] ⌈

Service Name	EthSwt_ReadPortMirrorConfiguration	
Syntax	`Std_ReturnType EthSwt_ReadPortMirrorConfiguration ( uint8 MirroredSwitchIdx, EthSwt_PortMirrorCfgType* PortMirrorConfigurationPtr )`	
Service ID [hex]	0x37	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	MirroredSwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver, where the Ethernet switch ports are located, that have to be mirrored
Parameters (inout)	None	
Parameters (out)	PortMirrorConfiguration Ptr	Pointer to the memory where the port configuration shall be stored.
Return value	Std_ReturnType	E_OK: the port mirror configuration for the indexed Ethernet switch port was red successfully. E_NOT_OK: the port mirror configuration for the indexed Ethernet switch was not red successfully. (i.e. indexed Ethernet switch is not available)
Description	Obtain the port mirror configuration of the given Ethernet switch.	
Available via	EthSwt.h	

⌋(*SRS_Eth_00123*)

[SWS_EthSwt_00313] ⌈The function `EthSwt_ReadPortMirrorConfiguration` shall return the port mirror configuration identified by the given MirroredSwitchIdx. If no port mirror configuration is found for the MirroredSwitchIdx, the function shall return `E_NOT_OK`.⌋(*SRS_Eth_00123*)

[SWS_EthSwt_00317] ⌈The function `EthSwt_ReadPortMirrorConfiguration` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtReadPortMirrorConfigurationApi`.⌋(*SRS_BSW_00171*)

8.3.42 EthSwt_DeletePortMirrorConfiguration

[SWS_EthSwt_91034] ⌈

Service Name	EthSwt_DeletePortMirrorConfiguration	
Syntax	`Std_ReturnType EthSwt_DeletePortMirrorConfiguration ( uint8 MirroredSwitchIdx )`	
Service ID [hex]	0x4a	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MirroredSwitchIdx. Non reentrant for the same SwitchIdx.	
Parameters (in)	MirroredSwitchIdx	Index of the switch within the context of the Ethernet Switch Driver.
Parameters (inout)	None	

▽

△

Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Port mirror configuration was deleted successfully E_NOT_OK: Port mirror configuration was not deleted successfully. (e.g. the port mirroring is enabled)
Description	Delete the stored port mirror configuration of the given MirroredSwitchIdx. If no port mirror configuration was found for the given MirroredSwitchIdx, the return value shall be E_OK.	
Available via	EthSwt.h	

⌋()

[SWS_EthSwt_00425] ⌈The function `EthSwt_DeletePortMirrorConfiguration` shall mark the stored port mirror configuration in the shadow buffer of the given MirroredSwitchIdx as "to be deleted".⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00426] ⌈If a port mirroring for the given MirroredSwitchIdx is enabled, the request to delete the configuration shall be rejected by returning `E_NOT_OK`. Only those port configurations are allowed to be deleted, where the port mirroring of the given MirroredSwitchIdx is disabled.⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00427] ⌈The function `EthSwt_DeletePortMirrorConfiguration` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtDeletePortMirrorConfigurationApi`.⌋*(SRS_BSW_00171)*

8.3.43 EthSwt_GetPortMirrorState

[SWS_EthSwt_91021] ⌈

Service Name	EthSwt_GetPortMirrorState	
Syntax	`Std_ReturnType EthSwt_GetPortMirrorState ( uint8 SwitchIdx, uint8 PortIdx, EthSwt_PortMirrorStateType* PortMirrorStatePtr )`	
Service ID [hex]	0x38	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	PortMirrorStatePtr	Pointer to the memory where the port mirroring state (either PORT_MIRRORING_ENABLED or PORT_MIRRORING_ DISABLED)of the given Ethernet switch port shall be stored.
Return value	Std_ReturnType	E_OK: the port mirroring state for the indexed Ethernet switch port returned successfully. E_NOT_OK: the port mirror configuration for the indexed Ethernet switch returned not successfully. (i.e. indexed ethernet switch port is not available)
Description	Obtain the current status of the port mirroring for the indexed Ethernet switch port	

▽

△

Available via	EthSwt.h

⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00318] ⌈The function `EthSwt_GetPortMirrorState` shall return the port mirroring state of the indexed ethernet switch port.⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00322] ⌈The function `EthSwt_GetPortMirrorState` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetPortMirrorStateApi`.⌋*(SRS_BSW_00171)*

8.3.44 EthSwt_SetPortMirrorState

[SWS_EthSwt_91022] ⌈

Service Name	EthSwt_SetPortMirrorState	
Syntax	`Std_ReturnType EthSwt_SetPortMirrorState ( uint8 MirroredSwitchIdx, EthSwt_PortMirrorStateType PortMirrorState )`	
Service ID [hex]	0x39	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	MirroredSwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver, where the port mirroring configuration is located that has to be enabled and disabled, repectively.
	PortMirrorState	Contain the requested port mirroring state either PORT_MIRRORING_ENABLED or PORT_MIRRORING_DISABLED
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: the requested port mirroring state for the indexed Ethernet switch port was set successfully. E_NOT_OK: the requested port mirroring state for the indexed Ethernet switch was not set successfully. (i.e. indexed Ethernet switch is not available, no port mirrior configuration is available)
Description	Request to set the given port mirroring state of the port mirror configuration for the given Ethernet switch.	
Available via	EthSwt.h	

⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00323] ⌈The function `EthSwt_SetPortMirrorState` shall request the given port mirroring state for the port mirroring configuration of the indexed Ethernet switch, and store the requested port mirror state in a shadow buffer.⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00327] ⌈The function `EthSwt_SetPortMirrorState` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtSetPortMirrorStateApi`.⌋*(SRS_BSW_00171)*

8.3.45 EthSwt_SetPortTestMode

[SWS_EthSwt_91029] ⌈

Service Name	EthSwt_SetPortTestMode	
Syntax	`Std_ReturnType EthSwt_SetPortTestMode ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_PhyTestModeType Mode )`	
Service ID [hex]	0x3a	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
	Mode	Test mode to be activated
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: the port test mode for the indexed Ethernet switch port was set successfully. E_NOT_OK: the port test mode for the indexed Ethernet switch was not set successfully. (i.e. indexed Ethernet switch port is not available)
Description	Activates a given test mode of the indexed Ethernet switch port.	
Available via	EthSwt.h	

⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00328] ⌈The function `EthSwt_SetPortTestMode` shall forward the call with the given test mode by calling the function `EthTrcv_SetPhyTestMode` of the referenced Ethernet Transceiver Driver.⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00332] ⌈The function `EthSwt_SetPortTestMode` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtSetPortTestModeApi`.⌋*(SRS_BSW_00171)*

8.3.46 EthSwt_SetPortLoopbackMode

[SWS_EthSwt_91023] ⌈

Service Name	EthSwt_SetPortLoopbackMode
Syntax	`Std_ReturnType EthSwt_SetPortLoopbackMode ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_PhyLoopbackModeType Mode )`
Service ID [hex]	0x3b
Sync/Async	Synchronous

▽

△

Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
	Mode	Loop-back mode to be activated
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: the port mirroring loop-back back mode for the indexed Ethernet switch port was activated successfully. E_NOT_OK: the port mirroring loop-back back mode for the indexed Ethernet switch port was not activated successfully. (i.e. indexed Ethernet switch port is not available)
Description	Activates a given test loop-back mode of the indexed Ethernet switch port.	
Available via	EthSwt.h	

⌋(*SRS_Eth_00123*)

[SWS_EthSwt_00334] ⌈The function `EthSwt_SetPortLoopbackMode` shall forward the call with the given loop-back mode by calling the function `EthTrcv_SetPhyLoopbackMode` of the referenced Ethernet Transceiver Driver.⌋(*SRS_Eth_00123*)

[SWS_EthSwt_00338] ⌈The function `EthSwt_SetPortLoopbackMode` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtSetPortLoopbackModeApi`.⌋(*SRS_BSW_00171*)

8.3.47 EthSwt_SetPortTxMode

[SWS_EthSwt_91024] ⌈

Service Name	EthSwt_SetPortTxMode	
Syntax	`Std_ReturnType EthSwt_SetPortTxMode ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_PhyTxModeType Mode )`	
Service ID [hex]	0x3c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
	Mode	Transmission mode to be activated
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: the port Tx mode for the indexed Ethernet switch port was activated successfully. E_NOT_OK: the port Tx mode for the indexed Ethernet switch port was not activated successfully. (i.e. indexed Ethernet switch port is not available)

▽

△

Description	Activates a given transmission mode of the indexed Ethernet switch port.
Available via	EthSwt.h

⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00340] ⌈The function `EthSwt_SetPortTxMode` shall forward the call with the given transmission mode by calling the function `EthTrcv_SetPhyTxMode` of the referenced Ethernet Transceiver Driver.⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00344] ⌈The function `EthSwt_SetPortTxMode` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtSetPortTxModeApi`.⌋*(SRS_BSW_00171)*

8.3.48 EthSwt_RunPortCableDiagnostic

[SWS_EthSwt_91011] ⌈

Service Name	EthSwt_RunPortCableDiagnostic	
Syntax	`Std_ReturnType EthSwt_RunPortCableDiagnostic ( uint8 SwitchIdxIdx, uint8 PortIdx )`	
Service ID [hex]	0x45	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different SwitchIdx and PortIdx. Non reentrant for the same SwitchIdx and Port Idx.	
Parameters (in)	SwitchIdxIdx	Index of the switch within the context of the Ethernet Switch Driver.
	PortIdx	Index of the port at the addressed switch.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The trigger to run the cable diagnostic has been accepted E_NOT_OK: The trigger to run the cable diagnostic has not been accepted
Description	Trigger the cable diagnostics of the given Ethernet Switch port (PortIdx) by calling EthTrcv_RunCableDiagnostic of the referenced Ethernet transceiver.	
Available via	EthSwt.h	

⌋*()*

[SWS_EthSwt_00429] ⌈The function `EthSwt_RunPortCableDiagnostic` shall forward the call by calling `EthTrcv_RunCableDiagnostic` of the referenced Ethernet Transceiver Driver.⌋*()*

8.3.49 EthSwt_GetPortCableDiagnosticsResult

[SWS_EthSwt_91025] ⌈

Service Name	EthSwt_GetPortCableDiagnosticsResult	
Syntax	`Std_ReturnType EthSwt_GetPortCableDiagnosticsResult ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_CableDiagResultType* ResultPtr )`	
Service ID [hex]	0x3f	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	ResultPtr	Pointer to the location where the cable diagnostics result shall be stored
Return value	Std_ReturnType	E_OK:the port cable diagnostic result for the indexed Ethernet switch port was obtained successfully. E_NOT_OK: the port cable diagnostic result for the indexed Ethernet switch port was not obtained successfully. (i.e. indexed Ethernet switch port is not available)
Description	Retrieves the cable diagnostics result of the indexed Ethernet switch port respectively the referenced Ethernet Transceiver Driver.	
Available via	EthSwt.h	

⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00346] ⌈The function `EthSwt_GetPortCableDiagnosticsResult` shall obtain the cable diagnostics result by calling the function `EthTrcv_GetCableDiagnosticsResult` of the referenced Ethernet Transceiver Driver.⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00350] ⌈The function `EthSwt_GetPortCableDiagnosticsResult` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetPortCableDiagnosticsResultApi`.⌋*(SRS_BSW_00171)*

8.3.50 EthSwt_GetCfgDataRaw

[SWS_EthSwt_91030] ⌈

Service Name	EthSwt_GetCfgDataRaw
Syntax	`Std_ReturnType EthSwt_GetCfgDataRaw ( uint8 SwitchIdx, uint32 Offset, uint16 Length, uint8* BufferPtr )`

▽

△

Service ID [hex]	0x41	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver
	Offset	Offset of the Ethernet switch memory from where the reading starts
	Length	Length of data in bytes that shall be copied
Parameters (inout)	None	
Parameters (out)	BufferPtr	Pointer to the location where the data shall be copied
Return value	Std_ReturnType	E_OK: the data read was triggered successfully E_NOT_OK: the data read was not triggered successfully (i.e. indexed Ethernet switch is not available)
Description	Retrieves the data in memory of the indexed Ethernet switch in variable length	
Available via	EthSwt.h	

⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00403] ⌈The function `EthSwt_GetCfgDataRaw` shall only be available if parameter EthSwtGetCfgRaw is set to TRUE.⌋*(SRS_BSW_00171)*

[SWS_EthSwt_00404] ⌈When calling the function `EthSwt_GetCfgDataRaw`, the function shall check the access to the Ethernet switch driver. If the check fails, the function shall raise the extended production error `ETHSWT_E_ACCESS` and return `E_NOT_OK`, otherwise pass the extended production error `ETHSWT_E_ACCESS` and return `E_OK`.⌋*()*

8.3.51 EthSwt_GetCfgDataInfo

[SWS_EthSwt_91031] ⌈

Service Name	EthSwt_GetCfgDataInfo	
Syntax	`Std_ReturnType EthSwt_GetCfgDataInfo ( uint8 SwitchIdx, uint32* DataSizePtr, uint32* DataAdressPtr )`	
Service ID [hex]	0x42	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver
Parameters (inout)	None	
Parameters (out)	DataSizePtr	Pointer to the location where the total size of the configuration data shall be copied
	DataAdressPtr	Pointer to the location where the start address of the configuration registers shall be copied

▽

△

Return value	Std_ReturnType	E_OK: the data was obtained successfully E_NOT_OK: the data was not obtained successfully. (i.e. indexed Ethernet switch is not available)
Description	Retrieves the total size of data and the memory start address of the indexed Ethernet Switch.	
Available via	EthSwt.h	

⌋*(SRS_Eth_00123)*

[SWS_EthSwt_00405] ⌈The function `EthSwt_GetCfgDataInfo` shall only be available if parameter EthSwtGetCfgRaw is set to TRUE.⌋*(SRS_BSW_00171)*

[SWS_EthSwt_00406] ⌈When calling the function `EthSwt_GetCfgDataInfo`, the function shall check the access to the Ethernet switch driver. If the check fails, the function shall raise the extended production error `ETHSWT_E_ACCESS` and return `E_NOT_OK`, otherwise pass the extended production error `ETHSWT_E_ACCESS` and return `E_OK`.⌋*()*

8.3.52 EthSwt_PortLinkStateRequest

[SWS_EthSwt_91123] ⌈

Service Name	EthSwt_PortLinkStateRequest	
Syntax	`Std_ReturnType EthSwt_PortLinkStateRequest ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_LinkStateType PortLinkState )`	
Service ID [hex]	0x49	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different SwitchIdx and PortIdx. Non reentrant for the same SwitchIdx and Port Idx.	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver.
	PortIdx	Index of the port at the addressed switch.
	PortLinkState	The Ethernet link state of a physical Ethernet connection.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request has been accepted and if the function call is in state ETHSWT_STATE_PORTINIT_COMPLETED or ETHSWT_STATE_ACTIVE E_NOT_OK: Request has not been accepted. (e.g. the indexed Ethernet switch port does not reference an EthTrcv)
Description	Request a link state by calling EthTrcv_TransceiverLinkStateRequest with the TrcvIdx of the Ethernet transceiver which is referenced by the Ethernet Switch port (PortIdx).	
Available via	EthSwt.h	

⌋*()*

[SWS_EthSwt_00415] ⌈The function `EthSwt_PortLinkStateRequest` shall request the given link state for the indexed Ethernet switch port of the switch by calling the `EthTrcv_TransceiverLinkStateRequest` with the given `EthTrcv_LinkStateType`. If the EthSwtPort does not reference an EthTrcv, then the function shall return `E_NOT_OK`.⌋*()*

8.3.53 EthSwt_GetMaxFIFOBufferFillLevel

[SWS_EthSwt_91050] ⌈

Service Name	EthSwt_GetMaxFIFOBufferFillLevel	
Syntax	`Std_ReturnType EthSwt_GetMaxFIFOBufferFillLevel ( uint8 SwitchIdx, uint8 SwitchPortIdx, uint8 SwitchPortEgressFifoIdx, uint32* SwitchPortEgressFifoBufferLevelPtr )`	
Service ID [hex]	0x48	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different SwitchIdx and PortIdx. Non reentrant for the same SwitchIdx and Port Idx.	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver.
	SwitchPortIdx	Index of the Ethernet switch egress port at the addressed Ethernet switch.
	SwitchPortEgressFifoIdx	Index of the egress FIFO of the addressed Ethernet switch port
Parameters (inout)	None	
Parameters (out)	SwitchPortEgressFifo BufferLevelPtr	Pointer to a memory location, where the maximum amount of allocated FIFO buffer (in bytes) since the last read out shall be stored
Return value	Std_ReturnType	E_OK: success E_NOT_OK: The maximal FIFO buffer level could not be obtained
Description	The function retrieves the maximum amount of allocated FIFO buffer of the indexed Ethernet switch egress port. If the Ethernet switch hardware does not support Ethernet switch port based maximal FIFO buffer level, the content of SwitchPortEgressFifoBufferLevelPtr shall be set to 0xFFFFFFFF. This API may be called by e.g. a CDD.	
Available via	EthSwt.h	

⌋*()*

[SWS_EthSwt_00430] ⌈The function `EthSwt_GetMaxFIFOBufferFillLevel` shall read out the maximum amount of allocated FIFO buffer since the last read out.⌋ *(SRS_ETH_00119)*

[SWS_EthSwt_00431] ⌈When the maximum amount of allocated FIFO buffer is read out, the value shall be reset to `0x00000000` explicitly, if it is not done by the hardware.⌋ *(SRS_ETH_00119)*

[SWS_EthSwt_00432] ⌈The function `EthSwt_GetMaxFIFOBufferFillLevel` shall be pre compile time configurable On/Off by the configuration parameter: `EthSwtGetMaxFIFOBufferFillLevelApi`.⌋*(SRS_BSW_00171)*

8.3.54 EthSwt_GetRxMgmtObject

[SWS_EthSwt_91038] ⌈

Service Name	EthSwt_GetRxMgmtObject	
Syntax	`Std_ReturnType EthSwt_GetRxMgmtObject ( uint8 CtrlIdx, Eth_DataType* DataPtr, EthSwt_MgmtObjectType** MgmtObjectPtr )`	
Service ID [hex]	0x47	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	CtrlIdx	Index of an Ethernet Interface controller
	DataPtr	Ethernet data pointer
Parameters (inout)	None	
Parameters (out)	MgmtObjectPtr	Pointer to the management object.
Return value	Std_ReturnType	E_OK: success E_NOT_OK: management object could not be obtained
Description	Obtains the MgmtObject of the (in this context) unique DataPtr.	
Available via	EthSwt.h	

⌋()

8.3.55 EthSwt_GetTxMgmtObject

[SWS_EthSwt_91039] ⌈

Service Name	EthSwt_GetTxMgmtObject	
Syntax	`Std_ReturnType EthSwt_GetTxMgmtObject ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, EthSwt_MgmtObjectType** MgmtObjectPtr )`	
Service ID [hex]	0x44	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	CtrlIdx	Index of an Ethernet Interface controller
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	None	
Parameters (out)	MgmtObjectPtr	Pointer to the management object.
Return value	Std_ReturnType	E_OK: success E_NOT_OK: management object could not be obtained
Description	Obtains the MgmtObject of the (in this context) unique BufIdx.	
Available via	EthSwt.h	

⌋()

8.4 Callback notifications

8.4.1 EthSwtPersistentConfigurationResultCallback

[SWS_EthSwt_00193] ⌈

Service Name	<EthSwtPersistentConfigurationResultCallback>	

Syntax	`void <EthSwtPersistentConfigurationResultCallback> (`
	`  NvM_RequestResultType JobResult`
	`)`	

Service ID [hex]	0x1b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	JobResult	Covers the job result of the previous processed single block job.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Job end notification of EthSwt_StoreConfiguration or EthSwt_ResetConfiguration	
Available via	EthSwtExternals.h	

⌋*(SRS_ETH_00122, SRS_ETH_00087)*

[SWS_EthSwt_00194] ⌈The callback function <EthSwtPersistentConfigurationResultCallback> shall be called by the `EthSwt_NvmSingleBlockCallback` to inform the caller of `EthSwt_StoreConfiguration` or `EthSwt_ResetConfiguration` about the state of the past calls.⌋*(SRS_ETH_00122, SRS_ETH_00087)*

8.5 Scheduled functions

8.5.1 EthSwt_MainFunction

[SWS_EthSwt_00114] ⌈

Service Name	EthSwt_MainFunction

Syntax	`void EthSwt_MainFunction (`
	`  void`
	`)`

Service ID [hex]	0x1c
Description	Service to support asynchronous behavior of API calls
Available via	EthSwt_SchM.h

⌋*(SRS_BSW_00433)*

[SWS_EthSwt_00115] ⌈The `EthSwt_MainFunction` support asynchronous behavior of API calls. This function is directly called by Basic Software Scheduler.⌋*(SRS_BSW_00433)*

8.5.2 EthSwt_BackgroundTask

[SWS_EthSwt_91104] ⌈

Service Name	EthSwt_BackgroundTask
Syntax	`void EthSwt_BackgroundTask ( void )`
Service ID [hex]	0x46
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	The background task should be scheduled as often as possible when no other task runs. It may be used for switch and port initialization in case the EthSwt_Init function needs too much time.
Available via	EthSwt.h

⌋()

8.6 Expected interfaces

In this chapter all external interfaces required from other modules are listed.

8.6.1 Mandatory Interfaces

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

No mandatory Interfaces defined.

8.6.2 Optional Interfaces

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

[SWS_EthSwt_00098] ⌈

API Function	Header File	Description
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)
Det_ReportError	Det.h	Service to report development errors.
Eth_ReadMii	Eth.h	Reads a transceiver register
Eth_WriteMii	Eth.h	Configures a transceiver register or triggers a function offered by the receiver
EthTrcv_GetBaudRate	EthTrcv.h	Obtains the baud rate of the indexed transceiver
EthTrcv_GetDuplexMode	EthTrcv.h	Obtains the duplex mode of the indexed transceiver
EthTrcv_GetLinkState	EthTrcv.h	Obtains the link state of the indexed transceiver
EthTrcv_GetTransceiverMode	EthTrcv.h	Obtains the state of the indexed transceiver
EthTrcv_SetTransceiverMode	EthTrcv.h	Enables / disables the indexed transceiver
EthTrcv_StartAutoNegotiation	EthTrcv.h	Restarts the negotiation of the transmission parameters used by the indexed transceiver
NvM_GetErrorStatus	NvM.h	Service to read the block dependent error/status information.
NvM_ReadBlock	NvM.h	Service to copy the data of the NV block to its corresponding RAM block.
NvM_WriteBlock	NvM.h	Service to copy the data of the RAM block to its corresponding NV block.
Spi_AsyncTransmit	Spi.h	Service to transmit data on the SPI bus.
Spi_Cancel	Spi.h	Service cancels the specified on-going sequence transmission.
Spi_ReadIB	Spi.h	Service for reading synchronously one or more data from an IB SPI Handler/Driver Channel specified by parameter.
Spi_SetAsyncMode	Spi.h	Service to set the asynchronous mechanism mode for SPI busses handled asynchronously.
Spi_SetupEB	Spi.h	Service to setup the buffers and the length of data for the EB SPI Handler/Driver Channel specified.
Spi_SyncTransmit	Spi.h	Service to transmit data on the SPI bus
Spi_WriteIB	Spi.h	Service for writing one or more data to an IB SPI Handler/Driver Channel specified by parameter.

⌋(*SRS_Eth_00122*, *SRS_ETH_00118*, *SRS_ETH_00119*, *SRS_ETH_00120*, *SRS_ETH_00087*, *SRS_ETH_00125*, *SRS_BSW_00375*)

[SWS_EthSwt_00192] ⌈The NvM APIs will only be used if the respective block is not configured for `NvM_ReadAll` and `NvM_WriteAll`.⌋(*SRS_Eth_00122*)

8.6.3 Configurable interfaces

In this chapter all interfaces are listed where the target function could be configured.

The names of these kind of interfaces are not fixed because they are configurable.

8.6.3.1 <EthSwtLinkDownCallout>

[SWS_EthSwt_00117] ⌈

Service Name	<EthSwtLinkDownCallout>	

Syntax	`void <EthSwtLinkDownCallout> (`
	`  uint8 SwitchIdx,`
	`  uint8 PortIdx`
	`)`	

Service ID [hex]	0x19	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Is called, if a link which is configured goes down.	
Available via	EthSwt_Externals.h	

⌋(*SRS_ETH_00119*, *SRS_ETH_00087*)

[SWS_EthSwt_00118] ⌈The function `<EthSwtLinkDownCallout>` shall be called if a link, which is configured, goes down (link loss). The function provides the Switch index and the Port index, such that the port which went down can be identified.⌋(*SRS_ETH_00119*, *SRS_ETH_00087*)

8.6.3.2 <EthSwtLinkUpCallout>

[SWS_EthSwt_00203] ⌈

Service Name	<EthSwtLinkUpCallout>	

Syntax	`void <EthSwtLinkUpCallout> (`
	`  uint8 SwitchIdx,`
	`  uint8 PortIdx`
	`)`	

Service ID [hex]	0x1a	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Is called, if a link which is configured goes up	
Available via	EthSwt_Externals.h	

⌋(*SRS_ETH_00119*, *SRS_ETH_00087*)

[SWS_EthSwt_00204] ⌈The function `<EthSwtLinkUpCallout>` shall be called if a link, which is configured, goes up. The function provides the Switch index and the Port index, such that the port which went up can be identified.⌋(*SRS_ETH_00119*, *SRS_ETH_00087*)

Note: If the hardware cannot signal a link up with an interrupt, the status of the link has to be determined in polling mode by checking the state of the link.

8.6.3.3 <GetCfgDataRawDone>

[SWS_EthSwt_91032] ⌈

Service Name	<GetCfgDataRawDone>	

Syntax	`void <GetCfgDataRawDone> (`
	`  uint8 SwitchIdx`
	`)`	

Service ID [hex]	0x43	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SwitchIdx	Index of the Ethernet switch where the Configuration is read.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The call of the function EthSwt_GetCfgDataRaw() triggers a asynchrony read of a certain memory section of the Ethernet switch driver. If the read is done, the configured callout function <GetCfgDataRawDone> shall be called]	
Available via	EthSwt_Externals.h	

⌋(*SRS_Eth_00123*)

8.7 Service Interfaces

No direct access is necessary from the application layer.

9 Sequence diagrams

The following sequence diagram shows the interaction between the DHCP-Server in the TCP/IP-module and the Ethernet Switch Driver:

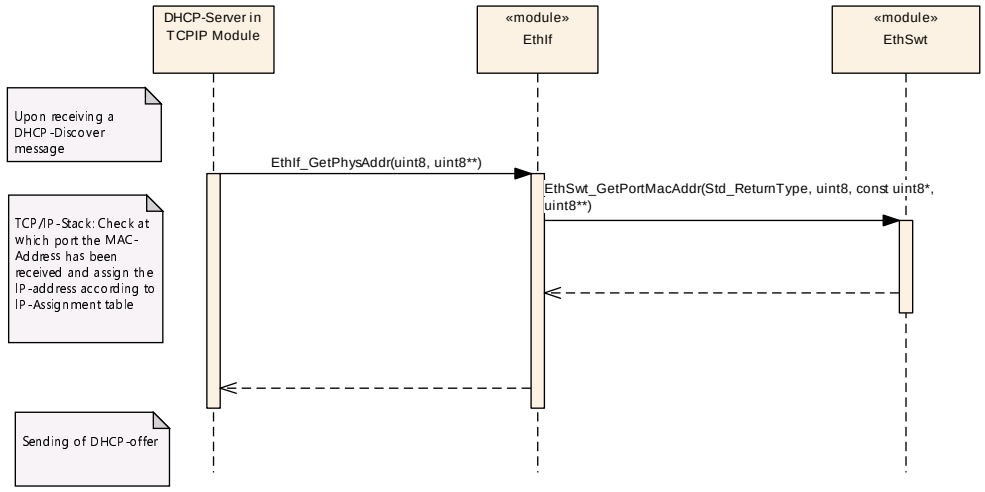

Figure 9.1

The following sequence diagram shows the interaction between the EthIf, EthSwt and the EthTrcv for API calls to the EthIf:

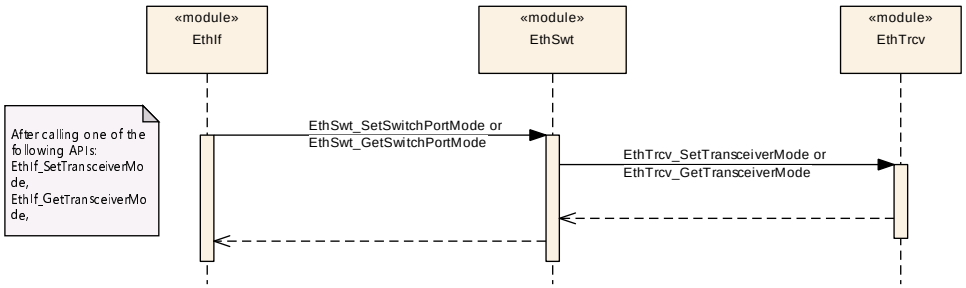

Figure 9.2

The following sequence diagram shows the interaction between the EthIf, EthSwt, and the EthTrcv for API calls which are initiated by the EthIf:

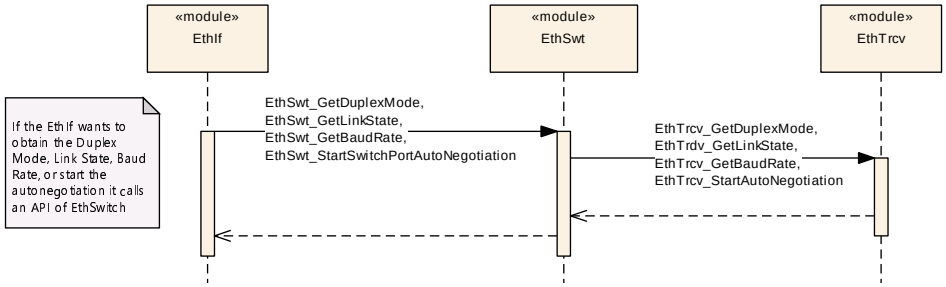

Figure 9.3

9.1 Switch Management support

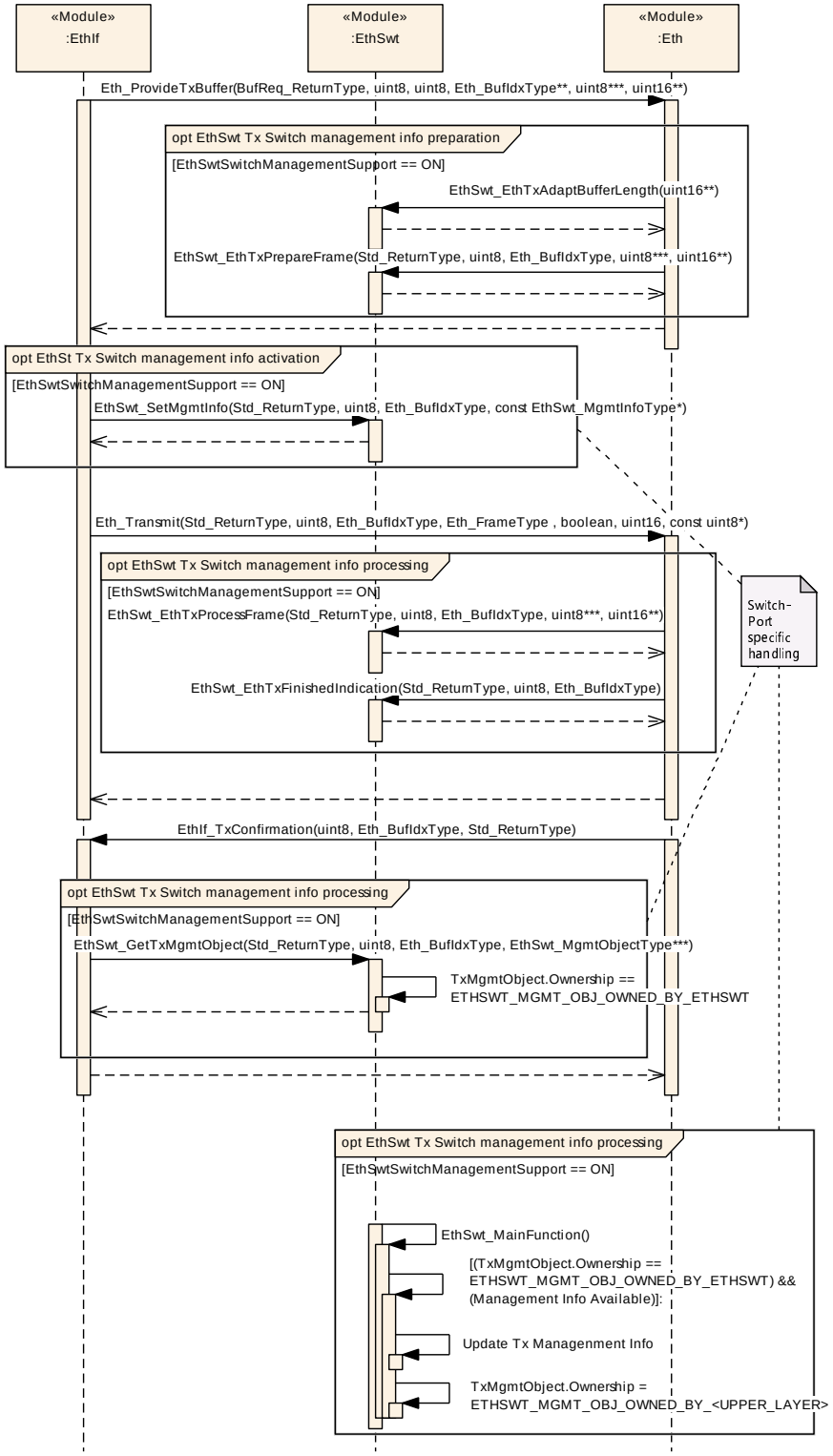

Figure 9.4: Switch Management support for transmission

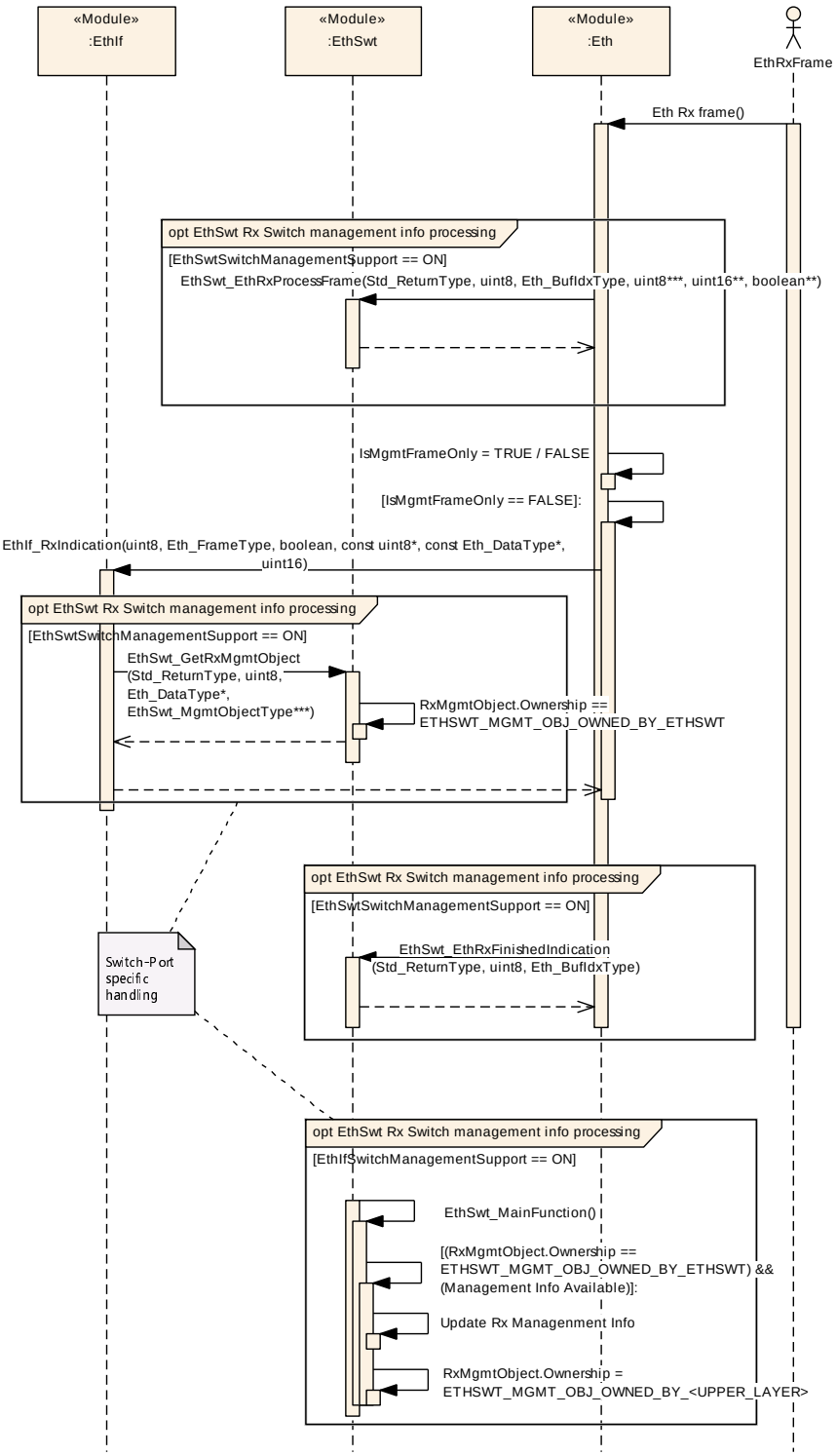

Figure 9.5: Management support for reception

10 Configuration specification

section 10.2 specifies the structure (containers) and the parameters of the module Eth Swt.

10.1 Containers and configuration parameters

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

[SWS_EthSwt_00414] ⌈The Ethernet Switch Driver module shall reject configurations with partition mappings which are not supported by the implementation.⌋*()*

10.1.1 EthSwt

Module SWS Item	ECUC_EthSwt_00046	
Module Name	EthSwt	
Module Description	Configuration of the EthSwt (Ethernet Switch Driver) module.	
Post-Build Variant Support	true	
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE	
Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthSwtConfig	1..*	Configuration of one Ethernet Switch.
EthSwtGeneral	1	General configuration of Ethernet Switch Driver module.

10.1.2 EthSwtConfig

SWS Item	[ECUC_EthSwt_00001]		
Container Name	EthSwtConfig		
Parent Container	EthSwt		
Description	Configuration of one Ethernet Switch.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Name	EthSwtArlTableEntryTimeout [ECUC_EthSwt_00127]		
Parent Container	EthSwtConfig		
Description	If present, this parameter specifies the timeout in seconds for removing unused entries from the ARL table of the Ethernet switch. Otherwise, entries are not removed automatically.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[1 .. 65535]		
Default Value	300		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtClockSynchronizationSupport [ECUC_EthSwt_00128]		
Parent Container	EthSwtConfig		
Description	This parameter defines, if a Ethernet switch shall enable clock synchronization with another Ethernet switch to which it is connected via uplink port. If this parameter is set to TRUE the clock synchronization between connected Ethernet switches is activated and the clocks of the Ethernet switches are synchronized. If this parameter is set to FALSE the clock synchronization between connected Ethernet switches is deactivated. This parameter shall only be set to TRUE if the Ethernet switch hardware supports clock synchronization.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	EthSwtDropDoubleTagged [ECUC_EthSwt_00073]		
Parent Container	EthSwtConfig		
Description	This parameter defines if a switch shall drop double tagged (Q in Q) frames. If this parameter is set to TRUE double tagged frames are dropped at all ports. If this parameter is set to FALSE, then double tagged frames are forwarded. If double tagging is used as a feature, this parameter must be set to FALSE. This parameter shall only be set to TRUE when Switch-HW supports the filtering of double tagged frames as filtering by SW is NOT possible!		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	EthSwtIdx [ECUC_EthSwt_00004]		
Parent Container	EthSwtConfig		
Description	Specifies the instance ID of the configured Ethernet Switch.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	EthSwtConfigEcucPartitionRef [ECUC_EthSwt_00130]
Parent Container	EthSwtConfig
Description	Maps the configuration of one single Ethernet switch to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the Ethernet switch driver is mapped to.
Multiplicity	0..1
Type	Reference to EcucPartition
Post-Build Variant Multiplicity	true
Post-Build Variant Value	true

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	EthSwtManagementEthCtrlRef [ECUC_EthSwt_00110]		
Parent Container	EthSwtConfig		
Description	Reference to the Ethernet controller connected to the management port where the management frames will be transmitted/received.		
Multiplicity	0..1		
Type	Symbolic name reference to EthCtrlConfig		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtManagementPortRef [ECUC_EthSwt_00111]		
Parent Container	EthSwtConfig		
Description	Reference to the port where the management CPU is connected to.		
Multiplicity	0..1		
Type	Reference to EthSwtPort		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthSwtDemEvent ParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
EthSwtNvm	0..1	Configuration of one Ethernet Switch Nvm usage in case the module requires non volatile memory in the Ecu to store switch configuration.
EthSwtPort	1..*	Configuration of one Ethernet Switch Port.
EthSwtSpi	0..1	Configuration of one Ethernet Switch SPI access (if SPI is used).

10.1.3 EthSwtDemEventParameterRefs

SWS Item	[ECUC_EthSwt_00016]		
Container Name	EthSwtDemEventParameterRefs		
Parent Container	EthSwtConfig		
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	ETHSWT_E_ACCESS [ECUC_EthSwt_00006]		
Parent Container	EthSwtDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Ethernet Switch Access Failure" has occurred.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

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

Name	ETHSWT_E_SYNCPORT2PHY [ECUC_EthSwt_00125]		
Parent Container	EthSwtDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "Ethernet switch port and the referenced Ethernet transceiver are in contradicting modes" has occurred.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

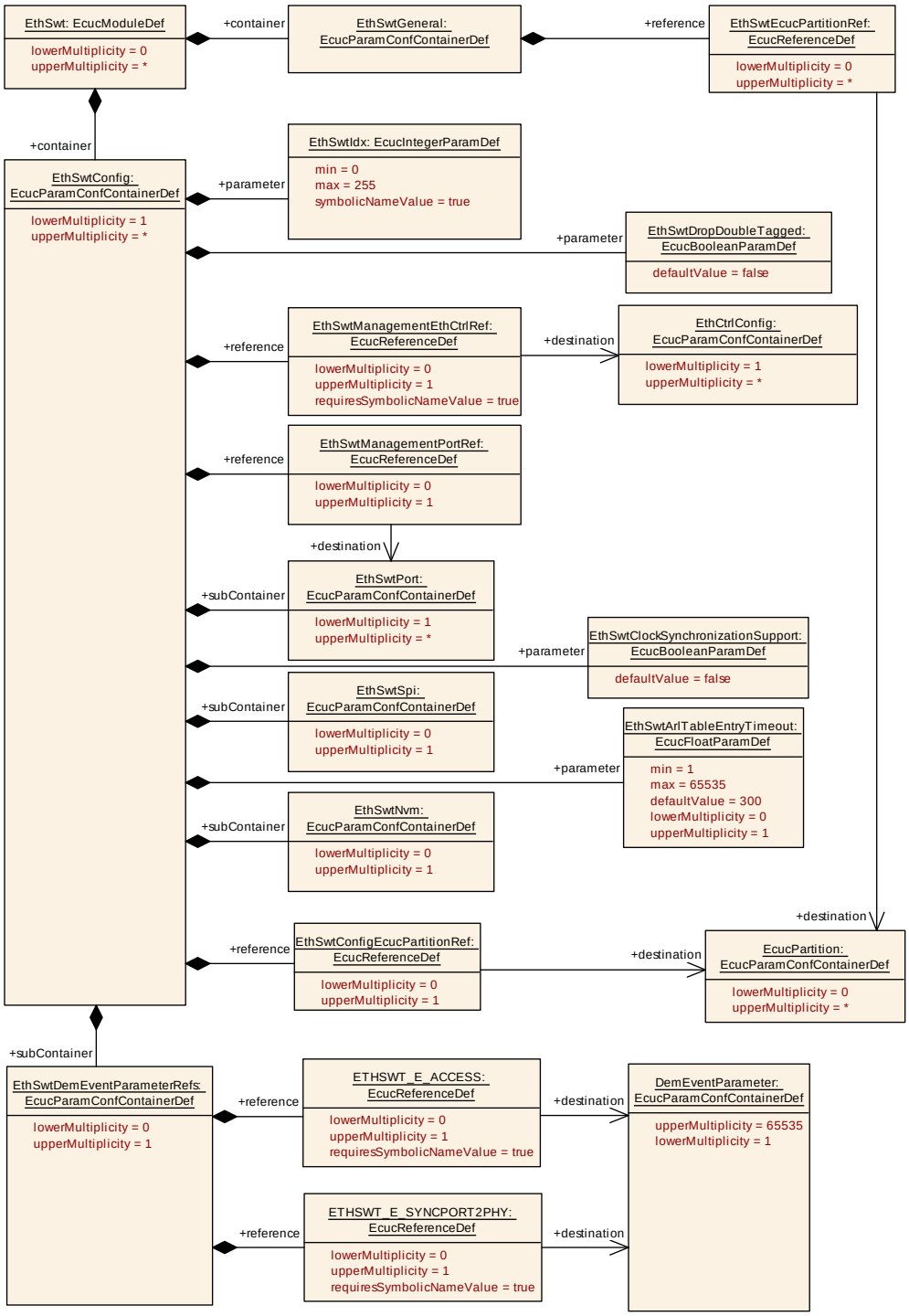

Figure 10.1: EthSwt

10.1.4 EthSwtGeneral

SWS Item	[ECUC_EthSwt_00003]
Container Name	EthSwtGeneral
Parent Container	EthSwt

Description	General configuration of Ethernet Switch Driver module.
Configuration Parameters	

Name	EthSwtCheckWakeupApi [ECUC_EthSwt_00136]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_CheckWakeup API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtDeletePortMirrorConfigurationApi [ECUC_EthSwt_00133]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_DeletePortMirrorConfiguration API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtDevErrorDetect [ECUC_EthSwt_00002]		
Parent Container	EthSwtGeneral		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtEnableCableDiagnosticApi [ECUC_EthSwt_00135]		
Parent Container	EthSwtGeneral		
Description	Enable/disable the APIs for cable diagnostic: EthSwt_RunPortCableDiagnostic, EthSwt_GetPortCableDiagnosticsResult		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtEnableVlanApi [ECUC_EthSwt_00055]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_EnableVLAN API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetArlTableApi [ECUC_EthSwt_00052]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetArlTable API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetBaudRateApi [ECUC_EthSwt_00121]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetBaudRate API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetCfgDataRawDone [ECUC_EthSwt_00124]		
Parent Container	EthSwtGeneral		
Description	Defines the function name for <GetCfgDataRawDone>		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local dependency: The function GetCfgDataRawDone shall only be configured if parameter EthSwtGetCfgRaw is set to TRUE.		

Name	EthSwtGetCfgRaw [ECUC_EthSwt_00123]		
Parent Container	EthSwtGeneral		
Description	Disable /Enable support of reading raw data from switch memory		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetCounterValuesApi [ECUC_EthSwt_00053]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetCounterValues API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetDuplexModeApi [ECUC_EthSwt_00122]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetDuplexMode API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetLinkStateApi [ECUC_EthSwt_00120]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetLinkState API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetMacLearningModeApi [ECUC_EthSwt_00061]
Parent Container	EthSwtGeneral
Description	Enables / Disables EthSwt_GetMacLearningMode API.
Multiplicity	1
Type	EcucBooleanParamDef
Default Value	

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

Name	EthSwtGetMaxFIFOBufferFillLevelApi [ECUC_EthSwt_00131]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetMaxFIFOBufferFillLevel API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetPortCableDiagnosticsResultApi [ECUC_EthSwt_00092]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetPortCableDiagnosticsResult API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetPortIdentifierApi [ECUC_EthSwt_00083]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetPortIdentifier API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Scope / Dependency	scope: local

Name	EthSwtGetPortMacAddrApi [ECUC_EthSwt_00051]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetPortMacAddr API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetPortMirrorStateApi [ECUC_EthSwt_00087]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetPortMirrorState API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetPortSignalQualityApi [ECUC_EthSwt_00082]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetPortSignalQuality API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetRxStatsApi [ECUC_EthSwt_00065]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetRxStats API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetSwitchIdentifierApi [ECUC_EthSwt_00084]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetSwitchIdentifier API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetSwitchPortModeApi [ECUC_EthSwt_00118]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetSwitchPortMode API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetSwitchPortWakeupReasonApi [ECUC_EthSwt_00137]
Parent Container	EthSwtGeneral
Description	Enables / Disables EthSwt_GetSwitchPortWakeupReason API.
Multiplicity	1
Type	EcucBooleanParamDef
Default Value	

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

Name	EthSwtGetSwitchRegApi [ECUC_EthSwt_00066]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_GetSwitchReg API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetTxErrorCounterValuesApi [ECUC_EthSwt_00100]		
Parent Container	EthSwtGeneral		
Description	Enables/Disables Eth_GetTxErrorCounterValues API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtGetTxStatsApi [ECUC_EthSwt_00099]		
Parent Container	EthSwtGeneral		
Description	Enables/Disables Eth_GetTxStats API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Scope / Dependency	scope: local

Name	EthSwtGlobalTimeSupportApi [ECUC_EthSwt_00107]		
Parent Container	EthSwtGeneral		
Description	Enables/Disables the Global Time APIs used amongst others by Global Time Synchronization over Ethernet.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtIndex [ECUC_EthSwt_00033]		
Parent Container	EthSwtGeneral		
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			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtLinkDownCallout [ECUC_EthSwt_00115]		
Parent Container	EthSwtGeneral		
Description	Defines the function name for the <EthSwtLinkDownCallout> callout.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

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

Name	EthSwtLinkUpCallout [ECUC_EthSwt_00116]		
Parent Container	EthSwtGeneral		
Description	Defines the function name for the <EthSwtLinkUpCallout> callout.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtLowPowerModeSupport [ECUC_EthSwt_00102]		
Parent Container	EthSwtGeneral		
Description	Disable / Enable support of low power mode.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtMainFunctionPeriod [ECUC_EthSwt_00071]		
Parent Container	EthSwtGeneral		
Description	The cycle time of the periodic main function of EthSwt. Defined in seconds .		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

Name	EthSwtManagementSupportApi [ECUC_EthSwt_00108]		
Parent Container	EthSwtGeneral		
Description	Enables/Disables the Switch management APIs to support a Switch-port specific communication attribute access.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtPersistentConfigurationResult [ECUC_EthSwt_00062]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables the callback API <User>_PersistentConfigurationResult.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtPersistentConfigurationResultCallback [ECUC_EthSwt_00063]		
Parent Container	EthSwtGeneral		
Description	Defines the function name for <EthSwtPersistentConfigurationResultCallback>.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtPublicCddHeaderFile [ECUC_EthSwt_00064]		
Parent Container	EthSwtGeneral		
Description	Defines header files for callback functions which shall be included in case of CDDs.		
Multiplicity	0..*		
Type	EcucStringParamDef		
Default Value			
Length	1–32		
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtReadPortMirrorConfigurationApi [ECUC_EthSwt_00086]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_ReadPortMirrorConfiguration API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtReadTrcvRegisterApi [ECUC_EthSwt_00069]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_ReadTrcvRegister API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtResetConfigurationApi [ECUC_EthSwt_00049]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_ResetConfiguration API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtSetForwardingModeApi [ECUC_EthSwt_00104]
Parent Container	EthSwtGeneral
Description	Enables /disables EthSwt_SetForwardingMode API.
Multiplicity	1
Type	EcucBooleanParamDef
Default Value	false

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

Name	EthSwtSetMacLearningModeApi [ECUC_EthSwt_00060]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_SetMacLearningMode API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtSetPortLoopbackModeApi [ECUC_EthSwt_00090]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_SetPortLoopbackModeApi API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtSetPortMirrorStateApi [ECUC_EthSwt_00088]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_SetPortMirrorState API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Scope / Dependency	scope: local

Name	EthSwtSetPortTestModeApi [ECUC_EthSwt_00089]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_SetPortTestMode API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtSetPortTxModeApi [ECUC_EthSwt_00091]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_SetPortTxModeApi API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtSetSwitchPortModeApi [ECUC_EthSwt_00117]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_SetSwitchPortMode API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtSetSwitchRegApi [ECUC_EthSwt_00067]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_SetSwitchReg API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtStartSwitchPortAutoNegotiationApi [ECUC_EthSwt_00119]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_StartSwitchPortAutoNegotiation API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtStoreConfigurationApi [ECUC_EthSwt_00050]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_StoreConfiguration API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtVerifyConfigApi [ECUC_EthSwt_00105]
Parent Container	EthSwtGeneral
Description	Enables /disables EthSwt_VerifyConfig API.
Multiplicity	1
Type	EcucBooleanParamDef
Default Value	false

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

Name	EthSwtVersionInfoApi [ECUC_EthSwt_00031]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables version info API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtWritePortMirrorConfigurationApi [ECUC_EthSwt_00085]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_WritePortMirrorConfiguration API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtWriteTrcvRegisterApi [ECUC_EthSwt_00070]		
Parent Container	EthSwtGeneral		
Description	Enables / Disables EthSwt_WriteTrcvRegister API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Scope / Dependency	scope: local

Name	EthSwtEcucPartitionRef [ECUC_EthSwt_00129]		
Parent Container	EthSwtGeneral		
Description	Maps the Ethernet switch driver to zero or multiple ECUC partitions to make the modules API available in this partition. The Ethernet switch driver will operate as an independent instance in each of the partitions.		
Multiplicity	0..*		
Type	Reference to EcucPartition		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

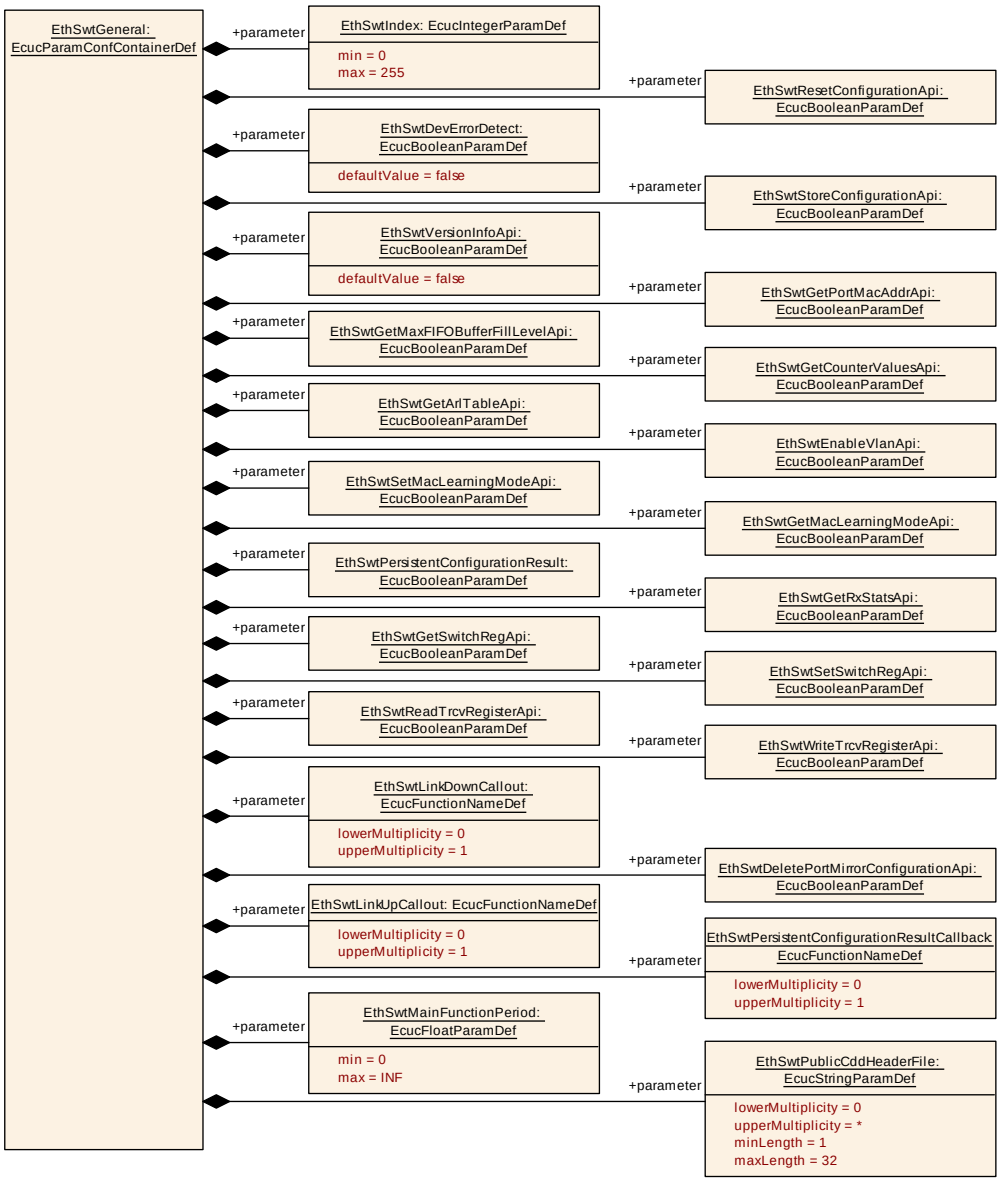

Figure 10.2: EthSwtGeneral (1/2)

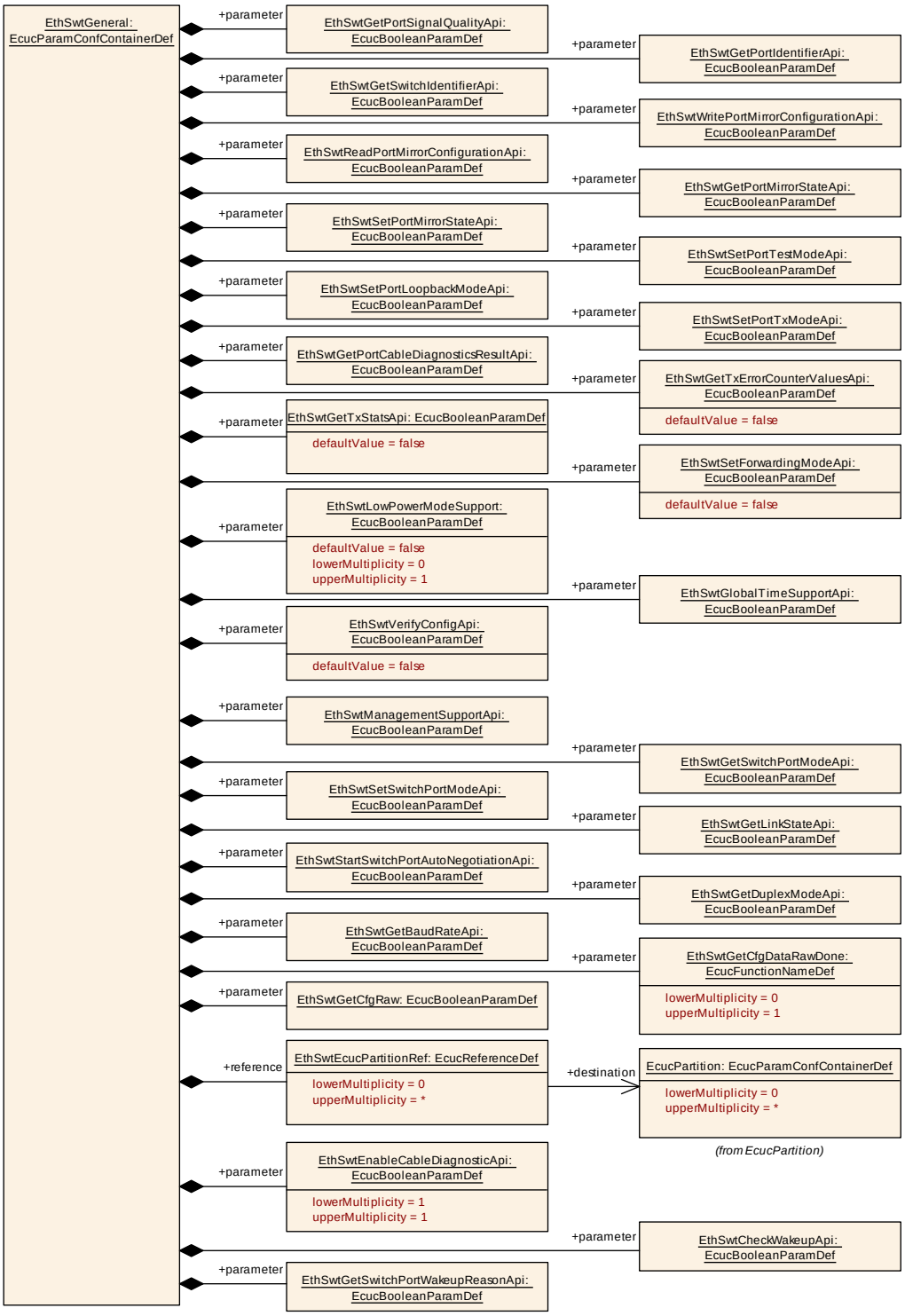

Figure 10.3: EthSwtGeneral (2/2)

10.1.5 EthSwtPort

SWS Item	[ECUC_EthSwt_00005]
Container Name	EthSwtPort

Parent Container	EthSwtConfig		
Description	Configuration of one Ethernet Switch Port.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Name	EthSwtPortIdx [ECUC_EthSwt_00013]		
Parent Container	EthSwtPort		
Description	Specifies the instance ID of the configured Ethernet Switch Port.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	EthSwtPortMacLayerSpeed [ECUC_EthSwt_00114]		
Parent Container	EthSwtPort		
Description	Defines the baud rate of the MAC layer.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	ETH_MAC_LAYER_SPEED_100M		
	ETH_MAC_LAYER_SPEED_10G		
	ETH_MAC_LAYER_SPEED_10M		
	ETH_MAC_LAYER_SPEED_1G		
	ETH_MAC_LAYER_SPEED_2500M		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	

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

Name	EthSwtPortMacLayerSubType [ECUC_EthSwt_00113]		
Parent Container	EthSwtPort		
Description	Defines the MAC layer subtype of this EthSwtPort.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	REDUCED	Reduced media-independent interface	
	REVERSED	reversed media-independent interface (to provide direct connection between two Ethernet MACs)	
	SERIAL	low-power and low pin-count serial 8b/10b-coded media-independent interface	
	STANDARD	standard media-independent interface	
	UNIVERSAL_SERIAL	Universal low-power and low pin-count serial 8b/10b-coded media-independent interface	
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	EthSwtPortMacLayerType [ECUC_EthSwt_00072]	
Parent Container	EthSwtPort	
Description	Defines the MAC layer type of this EthSwtPort.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	ETHSWT_PORT_MAC_LAYER_TYPE_XGMII	MAC layer interface (data) bandwith class 1Gbit/s (e.g. GMII, RGMII, SGMII, RvGMII, USGMII)
	ETHSWT_PORT_MAC_LAYER_TYPE_XMII	MAC layer interface (data) bandwith class 100Mbit/s (e.g. RMII, RvMII, SMII, RvMII)

	ETHSWT_PORT_MAC_LAYER_TYPE_XXGMII	MAC layer interface (data) bandwith class 10Gbit/s	
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	EthSwtPortPhysicalLayerType [ECUC_EthSwt_00054]		
Parent Container	EthSwtPort		
Description	Defines the physical layer type of this EthSwtPort.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	ETHSWT_PORT_1000BASE_T	physical layer interface 1000BASE-T (1Gbit/s, 4 pairs). Used for consumer electronic.	
	ETHSWT_PORT_1000BASE_T1	physical layer interface 1000BASE-T1 (1Gbit/s, 1 pair). Used for automotive.	
	ETHSWT_PORT_100BASE_T1	physical layer interface 100BASE-T1 (100Mbit/s, 1 pair). Used for automotive.	
	ETHSWT_PORT_100BASE_TX	physical layer interface 100BASE-TX (100Mbit/s, 2 pairs). Used for consumer electronic.	
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: If a EthSwtPort has an EthSwtPortPhysicalLayerType then EthSwtPort shall reference an EthTrcv.		

Name	EthSwtPortPredefinedMacAddresses [ECUC_EthSwt_00032]		
Parent Container	EthSwtPort		
Description	Specifies a list of 48-bit physical addresses (MAC addresses) which can be reached via this port in network byte order. Note that further addresses can be learned during runtime.		
Multiplicity	0..*		
Type	EcucStringParamDef		
Default Value			
Regular Expression	[0-9a-fA-F]{2}[[:-][0-9a-fA-F]{2}]{5}		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	EthSwtPortRole [ECUC_EthSwt_00101]		
Parent Container	EthSwtPort		
Description	Set a special role of the Ethernet switch port. It is either a host port or a up link port. If not configured it is a standard port.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	ETHSWT_HOST_PORT	The hostPort is connected to an ECU (host ecu). The host ECU controls the connected CouplingElement (e.g. Ethernet switch).	
	ETHSWT_UP_LINK_PORT	A CouplingPort can be connected to another CouplingPort of a CouplingElement located on the same ECU (CouplingElement.ecuInstance) using the CouplingPortConnection. This is used to model a cascaded switch.	
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local dependency: One Ethernet switch shall have either exactly one host port or at least one up link port. In case of having a host port also multiple up link port can exist. A master switch shall be connected by one host port with the host ecu. A slave switch shall be connected to a master switch by one up link port.

Name	EthSwtPortTimeStampSupport [ECUC_EthSwt_00112]		
Parent Container	EthSwtPort		
Description	Enables/Disables the Switch-port specific timestamping.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: EthSwtPortTimeStampSupport can only be set to TRUE, * if (EthSwtClockSynchronizationSupport is FALSE) OR * if ((EthSwtClockSynchronizationSupport is TRUE) AND (EthSwtPortRole is NOT ETHSWT_UP_LINK_PORT))		

Name	EthSwtPortTrcvRef [ECUC_EthSwt_00041]		
Parent Container	EthSwtPort		
Description	Reference to the Ethernet transceiver driver this EthSwtPort is connected with.		
Multiplicity	0..1		
Type	Symbolic name reference to EthTrcvConfig		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: If EthSwtPortPhysicalLayerType is defined, then EthSwtPortTrcvRef holds the reference to the corresponding EthTrcv.		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthSwtPortEgress	1	Configuration of one Ethernet Switch Port Egress behavior.
EthSwtPortIngress	1	Configuration of one Ethernet Switch Port ingress behavior.
EthSwtPortVlan Membership	0..4095	Description Determines the membership of this port to the virtual network, i.e. frames with this VID can be received and transmitted via this port.

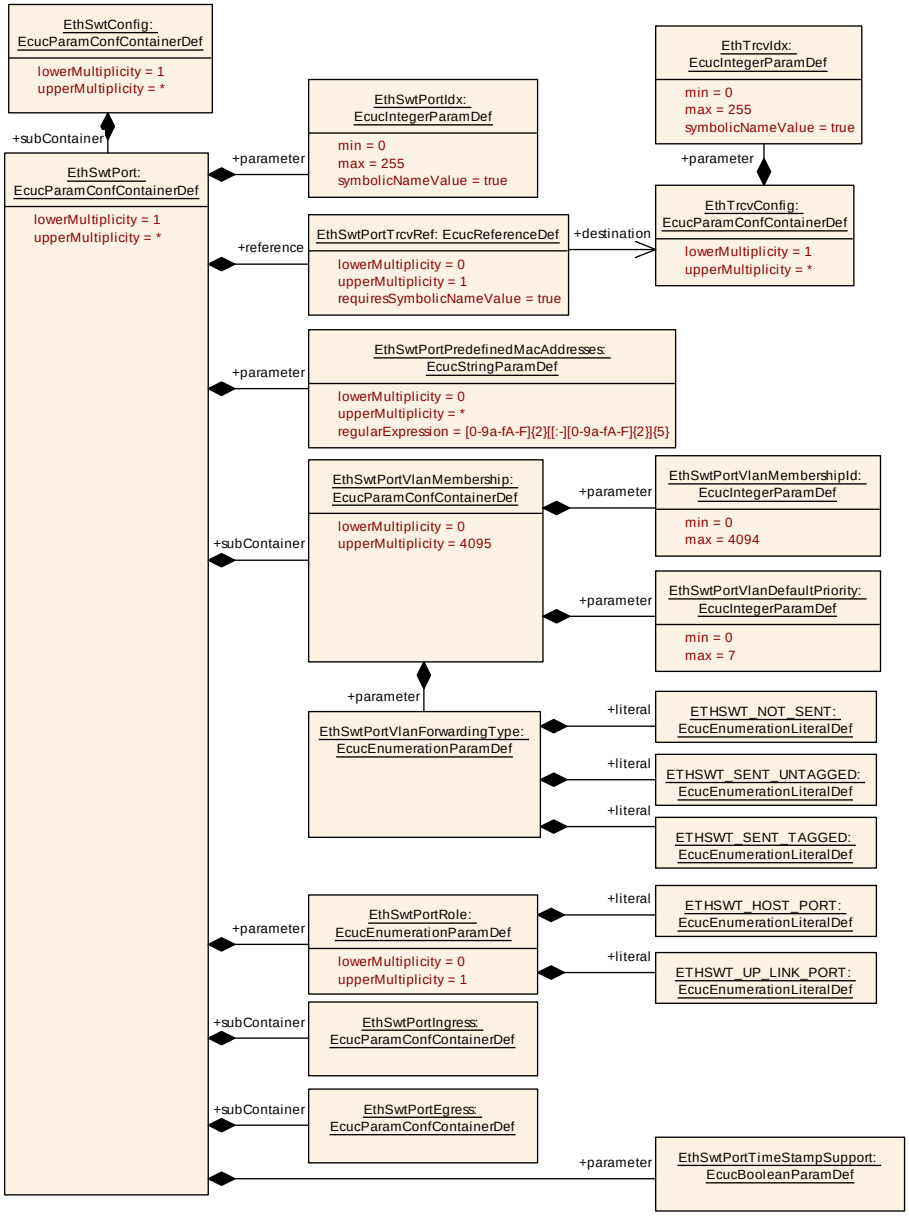

Figure 10.4: EthSwt Port (1/2)

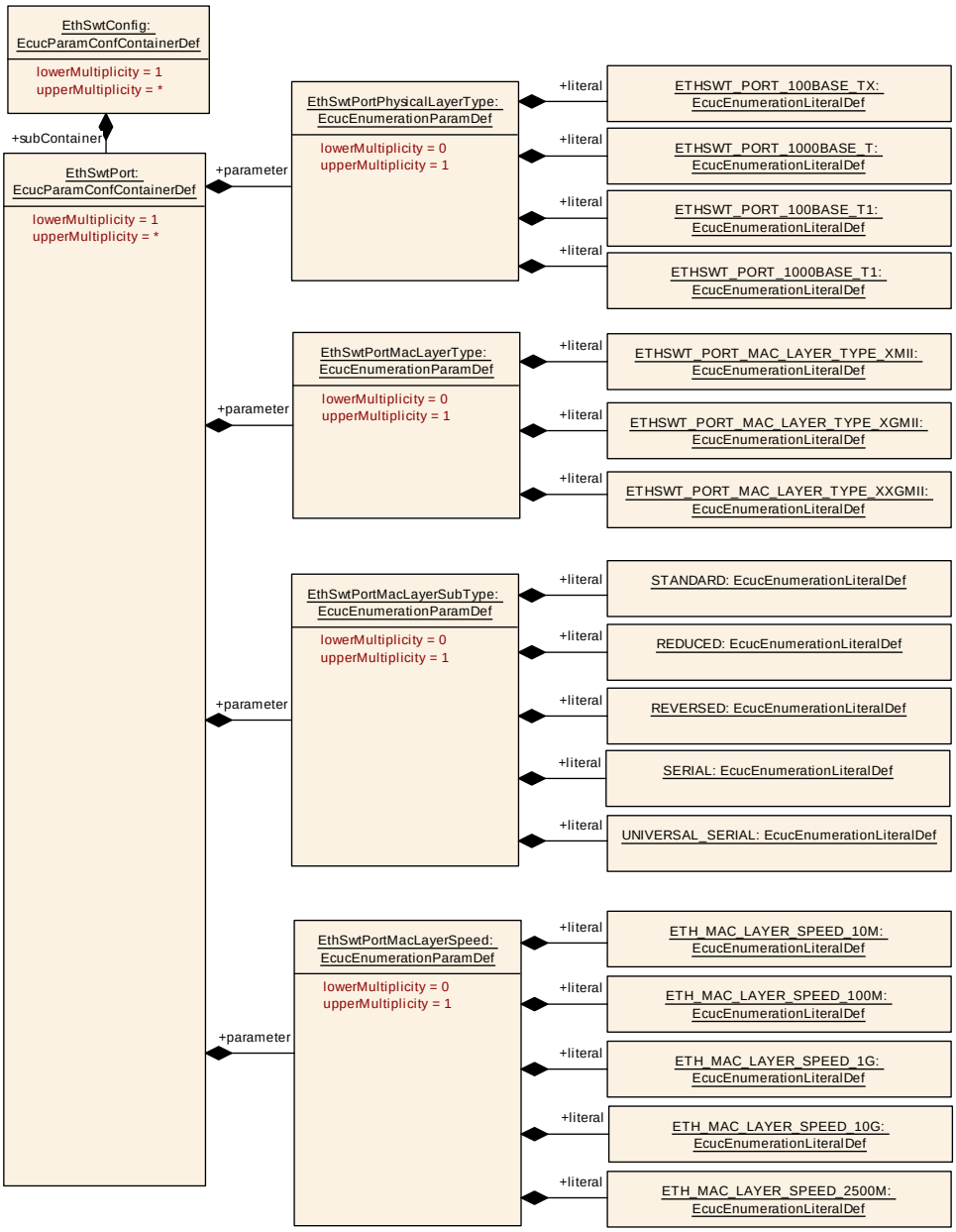

Figure 10.5: EthSwt Port (2/2)

Please note that the functional behavior of the ingress and egress port of a switch is implemented in hardware in the switch devices (see [11]). Thus, the configuration of EthSwtPort and described in the following has to be written to the switch device or is related to the switch configuration.

10.1.6 EthSwtPortIngress

SWS Item	[ECUC_EthSwt_00014]
Container Name	EthSwtPortIngress
Parent Container	EthSwtPort
Description	Configuration of one Ethernet Switch Port ingress behavior.

Configuration Parameters

Name	EthSwtPortIngressDefaultPriority [ECUC_EthSwt_00096]		
Parent Container	EthSwtPortIngress		
Description	Default priority for ingress.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default Value	0		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: If EthSwtPortIngressDefaultPriority is configured (multiplicity set to 1) then EthSwtPortIngressDefaultVlan shall be configured. If EthSwtPortIngressDefaultVlan is configured EthSwtPortIngressDropUntagged shall be set to FALSE.		

Name	EthSwtPortIngressDefaultVlan [ECUC_EthSwt_00095]		
Parent Container	EthSwtPortIngress		
Description	Default VLAN for ingress.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4094		
Default Value	1		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local dependency: If EthSwtPortIngressDefaultVlan is configured (multiplicity set to 1) then EthSwtPortIngressDefaultPriority shall be configured. If EthSwtPortIngressDefaultVlan is configured EthSwtPortIngressDropUntagged shall be set to FALSE.

Name	EthSwtPortIngressDropUntagged [ECUC_EthSwt_00097]		
Parent Container	EthSwtPortIngress		
Description	Defines the ingress behavior for untagged frames.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: If EthSwtPortIngressDropUntagged is set to TRUE then EthSwtPortIngressDefaultVlan and EthSwtPortIngressDefaultPriority parameters shall not be configured.		

Name	EthSwtPortIngressVlanModification [ECUC_EthSwt_00015]		
Parent Container	EthSwtPortIngress		
Description	If this parameter is defined all messages which arrive at this ingress port will be tagged with this VLAN Id. This tagging happen also if the arriving message already has a VLAN Id, it will be overwritten by the defined one. If this parameter is not defined no changes to the VLAN Id shall happen at this ingress port.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4095		
Default Value			
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	EthSwtPortTrafficClassAssignment [ECUC_EthSwt_00023]		
Parent Container	EthSwtPortIngress		
Description	If this parameter is defined all arriving messages at this ingress port shall be assigned this traffic class. If this parameter is not defined no general port based traffic class assignment is done.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default Value			
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthSwtPortPolicer	0..32760	Definition of Rate Policing parameters.
EthSwtPriority Regeneration	0..8	Defines a priority regeneration where the EthSwtPriorityRegenerationIngressPriority is replaced by EthSwtPriorityRegenerationRegeneratedPriority. The EthSwtPriorityRegeneration is optional in case no priority regeneration shall be performed. In case a EthSwtPriorityRegeneration is defined it shall have 8 mappings, one for each priority.
EthSwtPriorityTraffic ClassAssignment	0..8	Defines a priority based traffic class assignment. All messages with a specific priority (EthSwtPriorityTrafficClassAssignmentPriority) arriving at this ingress port or, if enabled regenerated priorities (EthSwtPriorityRegeneration), shall be assigned to a traffic class (EthSwtPriorityTrafficClassAssignmentTrafficClass).

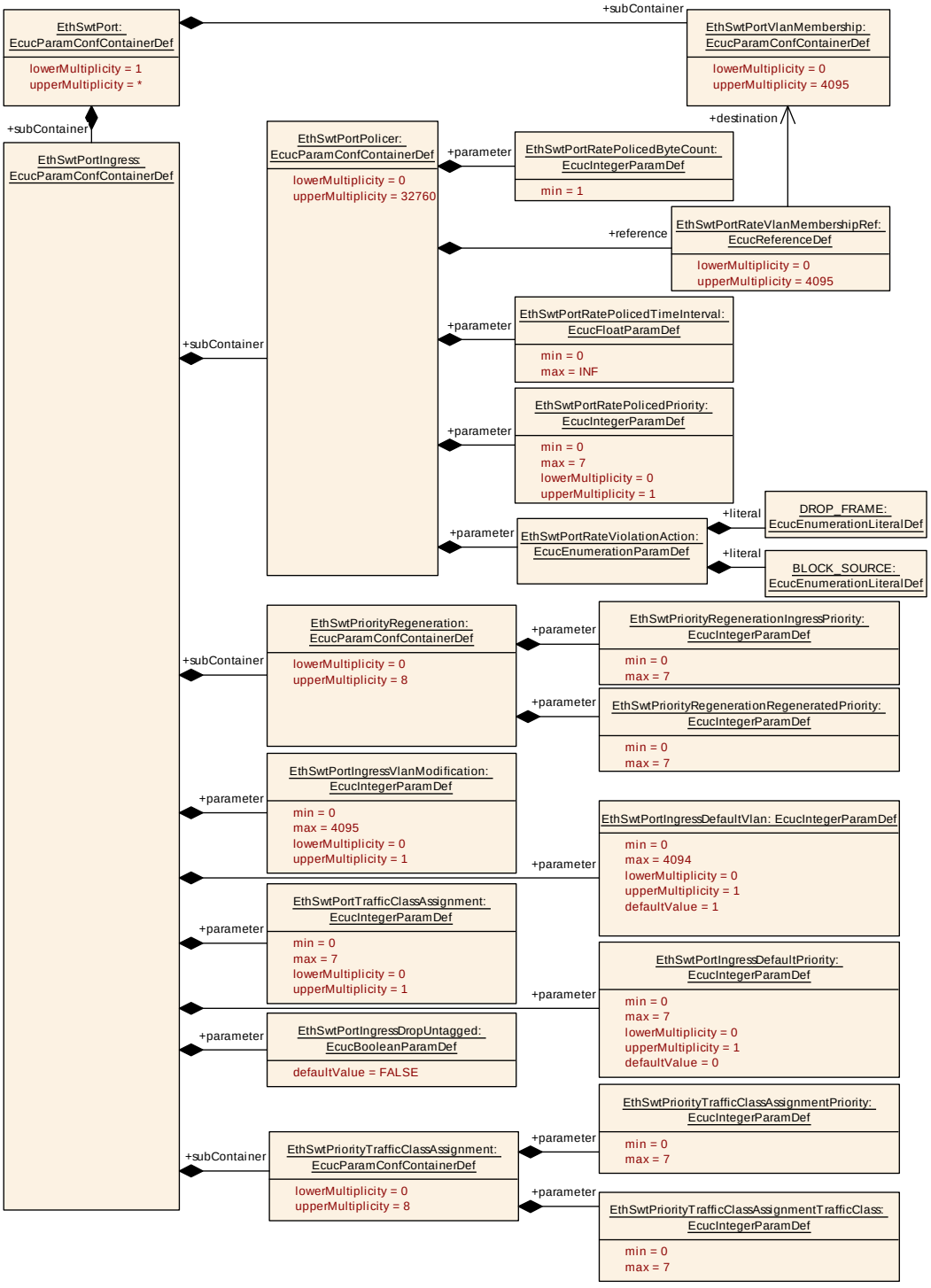

Figure 10.6: EthSwtPortIngress

10.1.7 EthSwtPortPolicer

SWS Item	[ECUC_EthSwt_00057]
Container Name	EthSwtPriorityRegeneration
Parent Container	EthSwtPortIngress

Description	Defines a priority regeneration where the EthSwtPriorityRegenerationIngressPriority is replaced by EthSwtPriorityRegenerationRegeneratedPriority. The EthSwtPriorityRegeneration is optional in case no priority regeneration shall be performed. In case a EthSwtPriorityRegeneration is defined it shall have 8 mappings, one for each priority.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Name	EthSwtPriorityRegenerationIngressPriority [ECUC_EthSwt_00058]		
Parent Container	EthSwtPriorityRegeneration		
Description	Message priority of the incoming message.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	EthSwtPriorityRegenerationRegeneratedPriority [ECUC_EthSwt_00059]		
Parent Container	EthSwtPriorityRegeneration		
Description	Message priority the incoming message will be tagged with.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

SWS Item	[ECUC_EthSwt_00074]		
Container Name	EthSwtPortPolicer		
Parent Container	EthSwtPortIngress		
Description	Definition of Rate Policing parameters.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Name	EthSwtPortRatePolicedByteCount [ECUC_EthSwt_00075]		
Parent Container	EthSwtPortPolicer		
Description	Amount of Byte Counts (excluding Header information) which can be received in a configured EthSwtPortRatePolicedTimeInterval.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 18446744073709551615		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	EthSwtPortRatePolicedPriority [ECUC_EthSwt_00077]		
Parent Container	EthSwtPortPolicer		
Description	Defines the priority which this rate policy shall be limited on. If no priority is given this rate policy is not considering priority.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default Value			
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local dependency: If no priority is configured the rate policing only applies to the configured EthSwtPortRateVlanMembershipRef.

Name	EthSwtPortRatePolicedTimeInterval [ECUC_EthSwt_00076]		
Parent Container	EthSwtPortPolicer		
Description	Time interval in seconds where a configured EthSwtPortRatePolicedByteCount can be received without a rate limitation.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	EthSwtPortRateViolationAction [ECUC_EthSwt_00078]		
Parent Container	EthSwtPortPolicer		
Description	Action to be taken when the rate policy criteria defined for this EthSwtPortPolicer are met.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BLOCK_SOURCE	All incoming traffic from the violating Source based on the MAC-Address is blocked.	
	DROP_FRAME	The received frame which led to the violation of the rate policy is dropped.	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	EthSwtPortRateVlanMembershipRef [ECUC_EthSwt_00081]		
Parent Container	EthSwtPortPolicer		
Description	References the Vlans this rate policy shall apply to. If no EthSwtPortRateVlanMembershipRef is configured the rate policing applies only on the configured EthSwtPortRatePolicedPriority.		
Multiplicity	0..4095		
Type	Reference to EthSwtPortVlanMembership		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.1.8 EthSwtPriorityTrafficClassAssignment

SWS Item	[ECUC_EthSwt_00027]		
Container Name	EthSwtPriorityTrafficClassAssignment		
Parent Container	EthSwtPortIngress		
Description	Defines a priority based traffic class assignment. All messages with a specific priority (EthSwtPriorityTrafficClassAssignmentPriority) arriving at this ingress port or, if enabled regenerated priorities (EthSwtPriorityRegeneration), shall be assigned to a traffic class (EthSwtPriorityTrafficClassAssignmentTrafficClass).		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Name	EthSwtPriorityTrafficClassAssignmentPriority [ECUC_EthSwt_00028]	
Parent Container	EthSwtPriorityTrafficClassAssignment	
Description	Message priority.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default Value		

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

Name	EthSwtPriorityTrafficClassAssignmentTrafficClass [ECUC_EthSwt_00029]		
Parent Container	EthSwtPriorityTrafficClassAssignment		
Description	Traffic Class value.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.1.9 EthSwtPortEgress

SWS Item	[ECUC_EthSwt_00007]
Container Name	EthSwtPortEgress
Parent Container	EthSwtPort
Description	Configuration of one Ethernet Switch Port Egress behavior.
Configuration Parameters	

Name	EthSwtPortEgressLastSchedulerRef [ECUC_EthSwt_00008]		
Parent Container	EthSwtPortEgress		
Description	Reference to the port scheduler which is the last in the egress port structure.		
Multiplicity	1		
Type	Reference to EthSwtPortScheduler		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthSwtPortFifo	1..*	Represents a Fifo in the egress port.
EthSwtPortScheduler	1..*	Represents a Scheduler in the egress port.
EthSwtPortShaper	0..*	Represents a Shaper in the egress port.

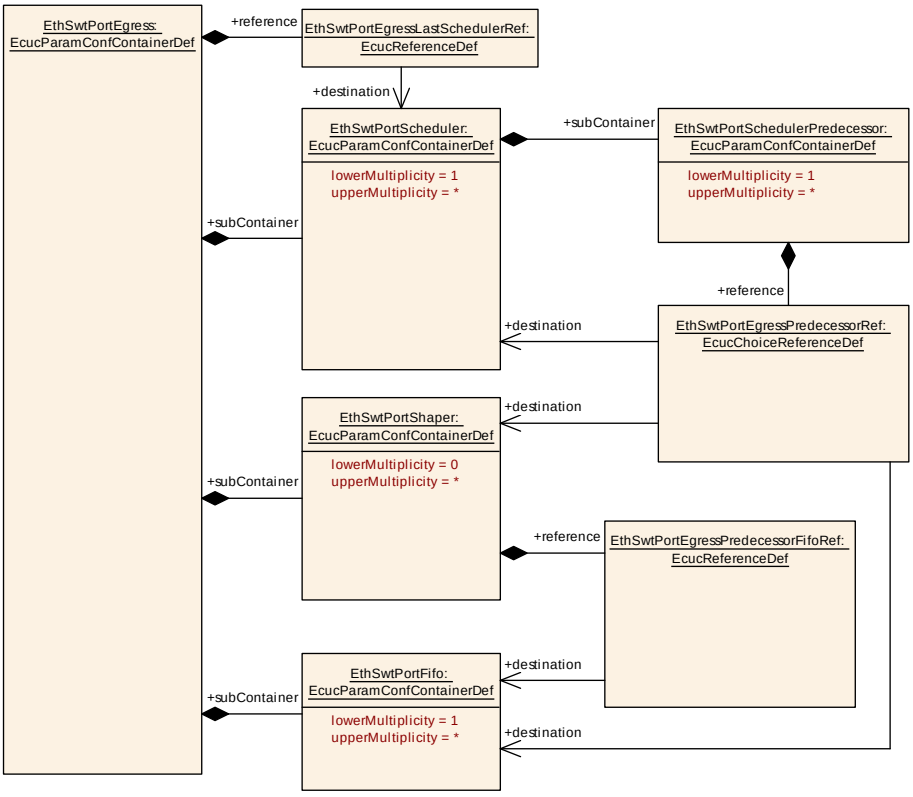

Figure 10.7: EthSwtPortEgress (1/2)

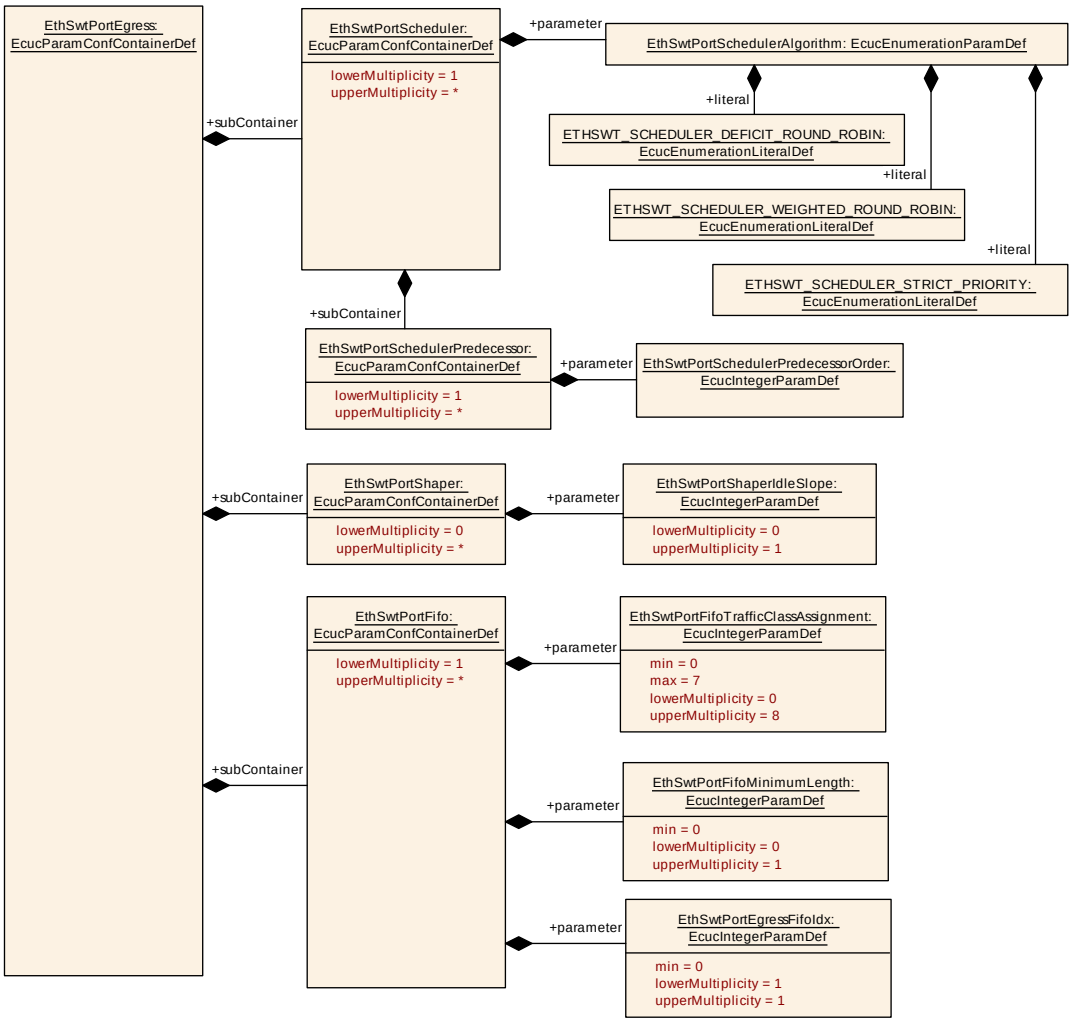

Figure 10.8: EthSwtPortEgress (2/2)

10.1.10 EthSwtPortScheduler

SWS Item	[ECUC_EthSwt_00017]		
Container Name	EthSwtPortScheduler		
Parent Container	EthSwtPortEgress		
Description	Represents a Scheduler in the egress port.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Name	EthSwtPortSchedulerAlgorithm [ECUC_EthSwt_00018]		
Parent Container	EthSwtPortScheduler		
Description	Defines the scheduler algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	ETHSWT_SCHEDULER_DEFICIT_ROUND_ROBIN	deficit round robin	
	ETHSWT_SCHEDULER_STRICT_PRIORITY	strict priority	
	ETHSWT_SCHEDULER_WEIGHTED_ROUND_ROBIN	weighted round robin	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthSwtPortScheduler Predecessor	1..*	Defines an ordered list of predecessors for this scheduler.

10.1.11 EthSwtPortSchedulerPredecessor

SWS Item	[ECUC_EthSwt_00019]		
Container Name	EthSwtPortSchedulerPredecessor		
Parent Container	EthSwtPortScheduler		
Description	Defines an ordered list of predecessors for this scheduler.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Name	EthSwtPortSchedulerPredecessorOrder [ECUC_EthSwt_00020]		
Parent Container	EthSwtPortSchedulerPredecessor		
Description	Defines the order of the scheduler predecessors. This value has to be understood as a relative value, i.e. the value shows only the relative ordering of the elements. The highest value has the highest priority and gaps are allowed (not dense based). The values need to be unique within one EthSwtPortScheduler.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	EthSwtPortEgressPredecessorRef [ECUC_EthSwt_00010]		
Parent Container	EthSwtPortSchedulerPredecessor		
Description	Choice reference to the scheduler predecessor.		
Multiplicity	1		
Type	Choice reference to [EthSwtPortFifo, EthSwtPortScheduler, EthSwtPortShaper]		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.1.12 EthSwtPortShaper

SWS Item	[ECUC_EthSwt_00021]
Container Name	EthSwtPortShaper
Parent Container	EthSwtPortEgress
Description	Represents a Shaper in the egress port.
Post-Build Variant Multiplicity	true

Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Name	EthSwtPortShaperIdleSlope [ECUC_EthSwt_00042]		
Parent Container	EthSwtPortShaper		
Description	Defines the increase of credit in bits per second for the AVB shaper.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	EthSwtPortEgressPredecessorFifoRef [ECUC_EthSwt_00009]		
Parent Container	EthSwtPortShaper		
Description	Reference to the fifo which is the predecessor for this shaper.		
Multiplicity	1		
Type	Reference to EthSwtPortFifo		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.1.13 EthSwtPortFifo

SWS Item	[ECUC_EthSwt_00011]
Container Name	EthSwtPortFifo
Parent Container	EthSwtPortEgress
Description	Represents a Fifo in the egress port.
Post-Build Variant Multiplicity	true

Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Name	EthSwtPortEgressFifoIdx [ECUC_EthSwt_00132]		
Parent Container	EthSwtPortFifo		
Description	Specifies the instance ID of the fifo of the configured Ethernet switch egress port		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	EthSwtPortFifoMinimumLength [ECUC_EthSwt_00098]		
Parent Container	EthSwtPortFifo		
Description	FIFO minimum length in Byte. This assignment is used to configure a guaranteed size of a configured FIFO.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default Value			
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	EthSwtPortFifoTrafficClassAssignment [ECUC_EthSwt_00012]		
Parent Container	EthSwtPortFifo		
Description	Defines which traffic classes are assigned to this Fifo.		
Multiplicity	0..8		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.1.14 EthSwtPortVlanMembership

SWS Item	[ECUC_EthSwt_00079]		
Container Name	EthSwtPortVlanMembership		
Parent Container	EthSwtPort		
Description	Description Determines the membership of this port to the virtual network, i.e. frames with this VID can be received and transmitted via this port.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Name	EthSwtPortVlanDefaultPriority [ECUC_EthSwt_00056]		
Parent Container	EthSwtPortVlanMembership		
Description	Determines the standard output-priority outgoing messages will be tagged with.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	EthSwtPortVlanForwardingType [ECUC_EthSwt_00026]		
Parent Container	EthSwtPortVlanMembership		
Description	Defines how the message with a specific VLAN Id shall be handled.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	ETHSWT_NOT_SENT	The message with the specific VLAN Id shall not be sent at this port.	
	ETHSWT_SENT_TAGGED	The message with the specific VLAN Id shall be sent with its VLAN Id at this port.	
	ETHSWT_SENT_UNTAGGED	The message with the specific VLAN Id shall sent untagged.	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	EthSwtPortVlanMembershipId [ECUC_EthSwt_00080]		
Parent Container	EthSwtPortVlanMembership		
Description	Determines the VID of the virtual network this port belongs to.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4094		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.1.15 EthSwtSpi

SWS Item	[ECUC_EthSwt_00030]		
Container Name	EthSwtSpi		
Parent Container	EthSwtConfig		
Description	Configuration of one Ethernet Switch SPI access (if SPI is used).		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Configuration Parameters

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthSwtSpiSequence	1..*	Container gives EthSwt driver information about one SPI sequence. One SPI sequence used by EthSwt driver is in exclusive use for it. No other driver is allowed to access this sequence. EthSwt driver may use one sequence to access n EthSwt hardware chips of the same type or n sequences are used to access one single EthSwt hardware chip. If a EthSwt hardware has no SPI interface, there is no instance of this container.

10.1.16 EthSwt Spi Sequence

SWS Item	[ECUC_EthSwt_00034]		
Container Name	EthSwtSpiSequence		
Parent Container	EthSwtSpi		
Description	Container gives EthSwt driver information about one SPI sequence. One SPI sequence used by EthSwt driver is in exclusive use for it. No other driver is allowed to access this sequence. EthSwt driver may use one sequence to access n EthSwt hardware chips of the same type or n sequences are used to access one single EthSwt hardware chip. If a EthSwt hardware has no SPI interface, there is no instance of this container.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	EthSwtSpiAccessSynchronous [ECUC_EthSwt_00036]
Parent Container	EthSwtSpiSequence
Description	This parameter is used to define whether the access to the Spi sequence is synchronous or asynchronous. true: SPI access is synchronous. false: SPI access is asynchronous.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default Value	
Post-Build Variant Multiplicity	true
Post-Build Variant Value	true

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	EthSwtSpiSequenceName [ECUC_EthSwt_00035]		
Parent Container	EthSwtSpiSequence		
Description	Reference to a Spi sequence configuration container.		
Multiplicity	0..*		
Type	Symbolic name reference to SpiSequence		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

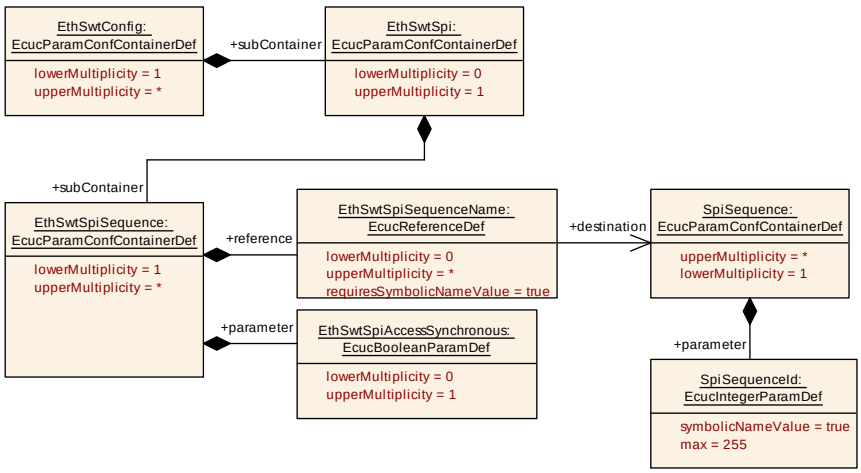

Figure 10.9: EthSwt SPI Interaction

10.1.17 EthSwtNvm

SWS Item	[ECUC_EthSwt_00043]		
Container Name	EthSwtNvm		
Parent Container	EthSwtConfig		
Description	Configuration of one Ethernet Switch Nvm usage in case the module requires non volatile memory in the Ecu to store switch configuration.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	EthSwtConfigurationNvmBlockDescriptorRef [ECUC_EthSwt_00134]		
Parent Container	EthSwtNvm		
Description	Reference to the Nvm block description in the Nvm module configuration to store e.g. the port mirror configurations		
Multiplicity	1		
Type	Symbolic name reference to NvMBlockDescriptor		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	EthSwtTableNvmBlockDescriptorRef [ECUC_EthSwt_00044]		
Parent Container	EthSwtNvm		
Description	Reference to the Nvm block description in the Nvm module configuration to store e.g. the learned ARL table		
Multiplicity	1		
Type	Symbolic name reference to NvMBlockDescriptor		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

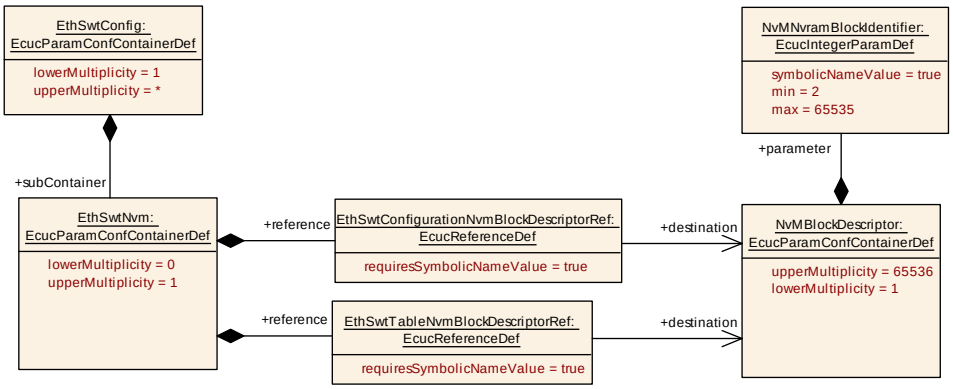

Figure 10.10: EthSwt Nvm Interaction

10.2 Constraints

[SWS_EthSwt_CONSTR_00413] ⌈The module will operate as an independent instance in each of the partitions (see `EthSwtEcucPartitionRef`), means the called API will only target the partition it is called in.⌋*()*

[SWS_EthSwt_CONSTR_00411] ⌈The ECUC partitions referenced by `EthSwtConfigEcucPartitionRef` shall be a subset of the ECUC partitions referenced by `EthSwtEcucPartitionRef`.⌋*()*

[SWS_EthSwt_CONSTR_00412] ⌈`EthSwtConfig`, `EthCtrlConfig` and `EthTrcvConfig` of one communication channel shall all reference the same ECUC partition.⌋*()*

[SWS_EthSwt_CONSTR_00438] ⌈If `EthSwtEcucPartitionRef` references one or more ECUC partitions, `EthSwtConfigEcucPartitionRef` shall have a multiplicity of one and reference one of these ECUC partitions as well.⌋*()*