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

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.0

Document Change History

Date	Release	Changed by	Change Description
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 Change Documentation

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2014-10-31	4.2.1	AUTOSAR Release Management	• Initial Release

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction and functional overview

In the AUTOSAR Layered Software Architecture, 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) 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. 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 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 or the SPI driver.

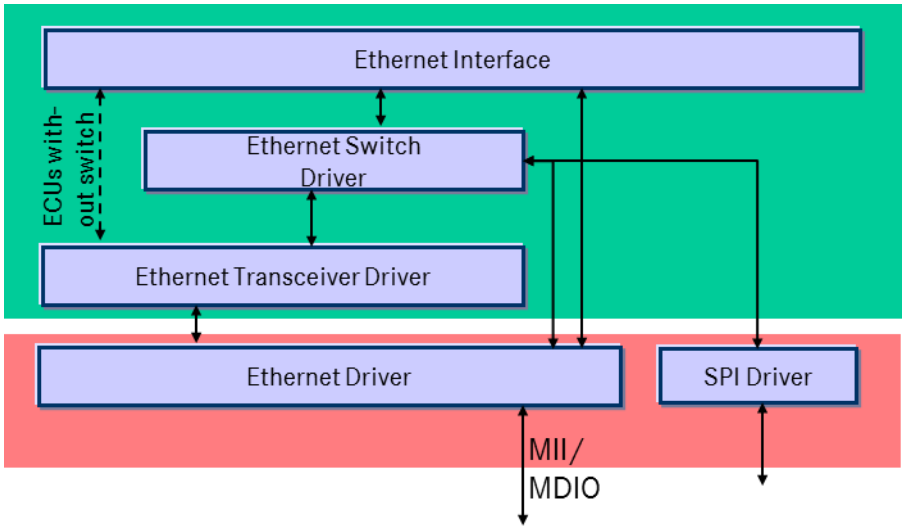


Figure 1 Ethernet Switch Driver in layer architecture

2 Acronyms and abbreviations

Abbreviation / Acronym:	*Description:*
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
internal ports	Internal ports are ports of an automotive Ethernet switch used for local communication (host ECU or cascaded switch)
external ports	External ports are ports of an automotive Ethernet switch used to communicate over an Ethernet physical connection with other ECUs (e.g. 100BASE-TX, 100BASE-T1).
host ecu	A host ECU is an ECU that controls one or more automotive Ethernet switches (e.g. switch on / off the Ethernet switch and its ports, read and write the Ethernet switch configuration). For this purpose the host ecu is connected to the Ethernet switch over a common control interface (e.g. SPI, MDIO). The host ECU also takes part in the network communication. For this purpose the host ECU is connected by a data interface (e.g. MII) to a specific Ethernet switch port (host port).It transmits and receives Ethernet frames.
host port	A host port is a port of an automotive Ethernet switch where the data interface (e.g. MII) of the host ECU is connected to. The host port could be either an internal port or an external port. The host port has a special role from the perspective of the software. (see link accumulation and port groups)
up link port	An up link port is a port of an automotive Ethernet switch, which is connected to another automotive Ethernet switch (cascaded switch). An up link port could be either an internal port or an external port. One up link port is connected to another up link port. The up link port has a special role from the perspective of the software.
port groups	Ethernet switch ports of an Ethernet switch are grouped to so called port groups. Port groups are only relevant for the host ECU. Port groups are derived from the model per VLAN and per PNC. The host port is participating in all port groups. A port group could be a mix of internal and external ports.
Link state accumulation	The link state of a certain switch port group is accumulated by embracing the link state of each port that is part of the port group. The rule how to embracing the link state is specified in the EthIf.
master switch	This is a switch where the host port is located.
slave switch	This is a switch which is connected to a master switch by up link ports.
cascaded switch	a cascaded switch is a Ethernet switch that exists of at least two Ethernet switches: a master switch and a slave switch. The master switch and the slave switch are connected by up link ports.

3 Related documentation

[1] AUTOSAR Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf
[2] AUTOSAR General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf
[3] AUTOSAR General Specification for Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf
[4] AUTOSAR Requirements on Ethernet Support in AUTOSAR
AUTOSAR_SRS_Ethernet.pdf
[5] AUTOSAR Specification of Ethernet Interface
AUTOSAR_SWS_EthernetInterface.pdf
[6] AUTOSAR Specification of Transceiver Driver
AUTOSAR_SWS_TransceiverDriver.pdf
[7] AUTOSAR Specification of Ethernet Driver
AUTOSAR_SWS_EthernetDriver.pdf
[8] AUTOSAR Specification of Global Time Synchronization over Ethernet
AUTOSAR_SWS_TimeSyncOverEthernet.pdf

3.1 Related standards and norms

[9] IEEE 802.1Q, http://standards.ieee.org/getieee802/download/802.1Q-2011.pdf
[10] IEEE 802.3, http://standards.ieee.org/about/get/802/802.3.html
[11] IEEE 802.1, http://standards.ieee.org/about/get/802/802.1.html

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software (SWS_BSWGeneral) [3] which is also valid for Ethernet Switch Driver.

Thus, the specifications SWS_BSWGeneral [3], SRS_Ethernet [4] 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.
- Testing Functionality: Mirroring of frames and the configuration of mirror ports is not supported.
- Software MAC Learning: The kind of MAC address learning is configurable, i.e. it can be either Disabled, Hardware, or Software. While the first two options are implemented in the switch hardware, the third option requires a software functionality. This functionality is not part of this specification.

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.

5.1 File structure

5.1.1 Header file structure

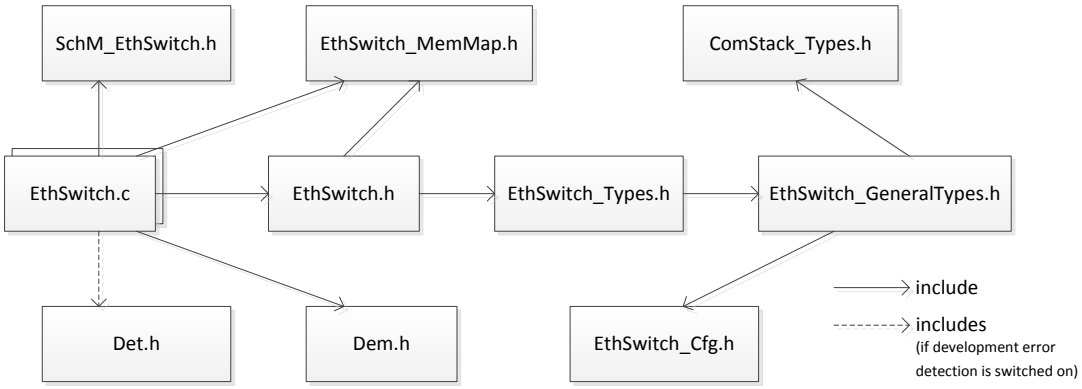


Figure 2 Ethernet Switch Driver file structure

6 Requirements traceability

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_00007, SWS_EthSwt_00008, SWS_EthSwt_00011, SWS_EthSwt_00012
SRS_BSW_00118	-	SWS_EthSwt_00046
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
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_00084, SWS_EthSwt_00090, SWS_EthSwt_00095, SWS_EthSwt_00109, SWS_EthSwt_00119, SWS_EthSwt_00124, SWS_EthSwt_00129, SWS_EthSwt_00177, SWS_EthSwt_00186, SWS_EthSwt_00191, SWS_EthSwt_00195, SWS_EthSwt_00202, SWS_EthSwt_00205, 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_00246, SWS_EthSwt_00249, SWS_EthSwt_00253, SWS_EthSwt_00257, SWS_EthSwt_00261, SWS_EthSwt_00264, SWS_EthSwt_00268, SWS_EthSwt_00273, SWS_EthSwt_00277, 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_00355, SWS_EthSwt_00359, SWS_EthSwt_00362, SWS_EthSwt_00370, SWS_EthSwt_00379
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_EthSwt_00131
SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific	SWS_EthSwt_00128, SWS_EthSwt_00164

	development error codes via the API	
SRS_BSW_00385	List possible error notifications	SWS_EthSwt_00001, SWS_EthSwt_00113, SWS_EthSwt_00137, SWS_EthSwt_00138, SWS_EthSwt_00139, SWS_EthSwt_00141, SWS_EthSwt_00142, SWS_EthSwt_00143, SWS_EthSwt_00144, SWS_EthSwt_00145, SWS_EthSwt_00146, SWS_EthSwt_00147, SWS_EthSwt_00148, SWS_EthSwt_00149, SWS_EthSwt_00150, SWS_EthSwt_00151, SWS_EthSwt_00152
SRS_BSW_00386	The BSW shall specify the configuration for detecting an error	SWS_EthSwt_00016, SWS_EthSwt_00164
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	noname, SWS_EthSwt_00013, SWS_EthSwt_00020, SWS_EthSwt_00027, SWS_EthSwt_00033, SWS_EthSwt_00039, SWS_EthSwt_00053, SWS_EthSwt_00062, SWS_EthSwt_00081, SWS_EthSwt_00088, SWS_EthSwt_00093, SWS_EthSwt_00107, SWS_EthSwt_00112, SWS_EthSwt_00116, SWS_EthSwt_00174, SWS_EthSwt_00184, SWS_EthSwt_00189, SWS_EthSwt_00200, SWS_EthSwt_00208, SWS_EthSwt_00213, SWS_EthSwt_00218, SWS_EthSwt_00223, SWS_EthSwt_00247, SWS_EthSwt_00250, SWS_EthSwt_00254, SWS_EthSwt_00258, SWS_EthSwt_00262, SWS_EthSwt_00265, SWS_EthSwt_00269, SWS_EthSwt_00274, SWS_EthSwt_00278, SWS_EthSwt_00285, SWS_EthSwt_00289, SWS_EthSwt_00294, SWS_EthSwt_00300, SWS_EthSwt_00306, SWS_EthSwt_00310, SWS_EthSwt_00314, SWS_EthSwt_00319, SWS_EthSwt_00324, SWS_EthSwt_00329, SWS_EthSwt_00335, SWS_EthSwt_00341, SWS_EthSwt_00347, SWS_EthSwt_00353, SWS_EthSwt_00357, SWS_EthSwt_00360, SWS_EthSwt_00367
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_EthSwt_00014, SWS_EthSwt_00021, SWS_EthSwt_00028, SWS_EthSwt_00034, SWS_EthSwt_00040, SWS_EthSwt_00047, SWS_EthSwt_00054, SWS_EthSwt_00082, SWS_EthSwt_00089, SWS_EthSwt_00094, SWS_EthSwt_00108, SWS_EthSwt_00120, SWS_EthSwt_00153, SWS_EthSwt_00154, SWS_EthSwt_00155, SWS_EthSwt_00156, SWS_EthSwt_00157, SWS_EthSwt_00158, SWS_EthSwt_00175, SWS_EthSwt_00180, SWS_EthSwt_00185, SWS_EthSwt_00190, SWS_EthSwt_00201, SWS_EthSwt_00209, SWS_EthSwt_00214, SWS_EthSwt_00219, SWS_EthSwt_00224, SWS_EthSwt_00237, SWS_EthSwt_00251, SWS_EthSwt_00255, SWS_EthSwt_00259, SWS_EthSwt_00266, SWS_EthSwt_00286, SWS_EthSwt_00290, SWS_EthSwt_00295, SWS_EthSwt_00298, SWS_EthSwt_00301, SWS_EthSwt_00304, SWS_EthSwt_00307, SWS_EthSwt_00311,

		SWS_EthSwt_00315, SWS_EthSwt_00320, SWS_EthSwt_00325, SWS_EthSwt_00330, SWS_EthSwt_00333, SWS_EthSwt_00336, SWS_EthSwt_00339, SWS_EthSwt_00342, SWS_EthSwt_00345, SWS_EthSwt_00348, SWS_EthSwt_00351, SWS_EthSwt_00354, SWS_EthSwt_00358, SWS_EthSwt_00361, SWS_EthSwt_00368
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_EthSwt_00115
SRS_BSW_00456	- A Header file shall be defined in order to harmonize BSW Modules	SWS_EthSwt_00003
SRS_BSW_323	-	SWS_EthSwt_00009, SWS_EthSwt_00014, SWS_EthSwt_00021, SWS_EthSwt_00028, SWS_EthSwt_00034, SWS_EthSwt_00040, SWS_EthSwt_00047, SWS_EthSwt_00054, SWS_EthSwt_00064, SWS_EthSwt_00082, SWS_EthSwt_00089, SWS_EthSwt_00094, SWS_EthSwt_00108, SWS_EthSwt_00153, SWS_EthSwt_00154, SWS_EthSwt_00155, SWS_EthSwt_00156, SWS_EthSwt_00157, SWS_EthSwt_00158, SWS_EthSwt_00166, SWS_EthSwt_00167, SWS_EthSwt_00168, SWS_EthSwt_00169, SWS_EthSwt_00170, SWS_EthSwt_00171, SWS_EthSwt_00175, SWS_EthSwt_00176, SWS_EthSwt_00180, SWS_EthSwt_00185, SWS_EthSwt_00190, SWS_EthSwt_00201, SWS_EthSwt_00209, SWS_EthSwt_00214, SWS_EthSwt_00219, SWS_EthSwt_00224, SWS_EthSwt_00237, SWS_EthSwt_00238, SWS_EthSwt_00239, SWS_EthSwt_00251, SWS_EthSwt_00252, SWS_EthSwt_00255, SWS_EthSwt_00256, SWS_EthSwt_00259, SWS_EthSwt_00260, SWS_EthSwt_00263, SWS_EthSwt_00266, SWS_EthSwt_00267, SWS_EthSwt_00270, SWS_EthSwt_00271, SWS_EthSwt_00272, SWS_EthSwt_00275, SWS_EthSwt_00276, SWS_EthSwt_00279, SWS_EthSwt_00280, SWS_EthSwt_00283, SWS_EthSwt_00284, SWS_EthSwt_00286, SWS_EthSwt_00290, SWS_EthSwt_00295, SWS_EthSwt_00296, SWS_EthSwt_00298, SWS_EthSwt_00301, SWS_EthSwt_00302, SWS_EthSwt_00304, SWS_EthSwt_00307, SWS_EthSwt_00311, SWS_EthSwt_00315, SWS_EthSwt_00316, SWS_EthSwt_00320, SWS_EthSwt_00321, SWS_EthSwt_00325, SWS_EthSwt_00326, SWS_EthSwt_00330, SWS_EthSwt_00331, SWS_EthSwt_00333, SWS_EthSwt_00336, SWS_EthSwt_00337, SWS_EthSwt_00339, SWS_EthSwt_00342, SWS_EthSwt_00343, SWS_EthSwt_00345, SWS_EthSwt_00348, SWS_EthSwt_00349, SWS_EthSwt_00351,

		SWS_EthSwt_00354, SWS_EthSwt_00358, SWS_EthSwt_00361, SWS_EthSwt_00363, SWS_EthSwt_00364, SWS_EthSwt_00365, SWS_EthSwt_00366, SWS_EthSwt_00368, SWS_EthSwt_00369, SWS_EthSwt_00371, SWS_EthSwt_00381, SWS_EthSwt_00382
SRS_BSW_369	-	SWS_EthSwt_00009, SWS_EthSwt_00014, SWS_EthSwt_00021, SWS_EthSwt_00028, SWS_EthSwt_00034, SWS_EthSwt_00040, SWS_EthSwt_00047, SWS_EthSwt_00054, SWS_EthSwt_00064, SWS_EthSwt_00082, SWS_EthSwt_00089, SWS_EthSwt_00094, SWS_EthSwt_00108, SWS_EthSwt_00153, SWS_EthSwt_00154, SWS_EthSwt_00155, SWS_EthSwt_00156, SWS_EthSwt_00157, SWS_EthSwt_00158, SWS_EthSwt_00166, SWS_EthSwt_00167, SWS_EthSwt_00168, SWS_EthSwt_00169, SWS_EthSwt_00170, SWS_EthSwt_00171, SWS_EthSwt_00175, SWS_EthSwt_00176, SWS_EthSwt_00180, SWS_EthSwt_00185, SWS_EthSwt_00190, SWS_EthSwt_00201, SWS_EthSwt_00209, SWS_EthSwt_00214, SWS_EthSwt_00219, SWS_EthSwt_00224, SWS_EthSwt_00237, SWS_EthSwt_00238, SWS_EthSwt_00239, SWS_EthSwt_00251, SWS_EthSwt_00252, SWS_EthSwt_00255, SWS_EthSwt_00256, SWS_EthSwt_00259, SWS_EthSwt_00260, SWS_EthSwt_00263, SWS_EthSwt_00266, SWS_EthSwt_00267, SWS_EthSwt_00270, SWS_EthSwt_00271, SWS_EthSwt_00272, SWS_EthSwt_00275, SWS_EthSwt_00276, SWS_EthSwt_00279, SWS_EthSwt_00280, SWS_EthSwt_00283, SWS_EthSwt_00284, SWS_EthSwt_00286, SWS_EthSwt_00290, SWS_EthSwt_00295, SWS_EthSwt_00296, SWS_EthSwt_00298, SWS_EthSwt_00301, SWS_EthSwt_00302, SWS_EthSwt_00304, SWS_EthSwt_00307, SWS_EthSwt_00311, SWS_EthSwt_00315, SWS_EthSwt_00316, SWS_EthSwt_00320, SWS_EthSwt_00321, SWS_EthSwt_00325, SWS_EthSwt_00326, SWS_EthSwt_00330, SWS_EthSwt_00331, SWS_EthSwt_00333, SWS_EthSwt_00336, SWS_EthSwt_00337, SWS_EthSwt_00339, SWS_EthSwt_00342, SWS_EthSwt_00343, SWS_EthSwt_00345, SWS_EthSwt_00348, SWS_EthSwt_00349, SWS_EthSwt_00351, SWS_EthSwt_00354, SWS_EthSwt_00358, SWS_EthSwt_00361, SWS_EthSwt_00363, SWS_EthSwt_00364, SWS_EthSwt_00365, SWS_EthSwt_00366, SWS_EthSwt_00368, SWS_EthSwt_00369, SWS_EthSwt_00371, SWS_EthSwt_00381, SWS_EthSwt_00382
SRS_ETH_00086	-	SWS_EthSwt_00006, SWS_EthSwt_00010, SWS_EthSwt_00018, SWS_EthSwt_00025, SWS_EthSwt_00031, SWS_EthSwt_00037, SWS_EthSwt_00044, SWS_EthSwt_00051, SWS_EthSwt_00058, SWS_EthSwt_00060,

		SWS_EthSwt_00079, SWS_EthSwt_00086, SWS_EthSwt_00091, SWS_EthSwt_00111, SWS_EthSwt_00114, SWS_EthSwt_00117, SWS_EthSwt_00123, SWS_EthSwt_00125, SWS_EthSwt_00165, SWS_EthSwt_00172, SWS_EthSwt_00182, SWS_EthSwt_00187, SWS_EthSwt_00193, SWS_EthSwt_00203, SWS_EthSwt_00206, SWS_EthSwt_00211, SWS_EthSwt_00216, SWS_EthSwt_00221, SWS_EthSwt_00227
SRS_Eth_00086	-	SWS_EthSwt_91012
SRS_ETH_00087	-	SWS_EthSwt_00032, SWS_EthSwt_00060, SWS_EthSwt_00061, SWS_EthSwt_00086, SWS_EthSwt_00087, SWS_EthSwt_00091, SWS_EthSwt_00092, SWS_EthSwt_00111, SWS_EthSwt_00117, SWS_EthSwt_00118, SWS_EthSwt_00125, SWS_EthSwt_00126, SWS_EthSwt_00127, SWS_EthSwt_00136, SWS_EthSwt_00158, 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_00228, SWS_EthSwt_00235
SRS_ETH_00114	-	SWS_EthSwt_00134, SWS_EthSwt_00173
SRS_Eth_00114	Ethernet Switch Filtering and Policing	SWS_EthSwt_00233
SRS_ETH_00118	-	SWS_EthSwt_00019, SWS_EthSwt_00023, SWS_EthSwt_00026, SWS_EthSwt_00038, SWS_EthSwt_00045, SWS_EthSwt_00052, SWS_EthSwt_00153, SWS_EthSwt_00154, SWS_EthSwt_00155, SWS_EthSwt_00156, SWS_EthSwt_00157, SWS_EthSwt_00164, SWS_EthSwt_00217, SWS_EthSwt_00222, SWS_EthSwt_00298, SWS_EthSwt_00304, SWS_EthSwt_00333, SWS_EthSwt_00339, SWS_EthSwt_00345, SWS_EthSwt_00351
SRS_ETH_00119	-	SWS_EthSwt_00038, SWS_EthSwt_00080, SWS_EthSwt_00118, SWS_EthSwt_00154, SWS_EthSwt_00204
SRS_ETH_00120	-	SWS_EthSwt_00217, SWS_EthSwt_00222
SRS_Eth_00120	Hardware access via MII and/or SPI	SWS_EthSwt_00207, SWS_EthSwt_00212
SRS_ETH_00121	-	SWS_EthSwt_00132, SWS_EthSwt_00133, SWS_EthSwt_00134, SWS_EthSwt_00135, SWS_EthSwt_00173, SWS_EthSwt_00178, SWS_EthSwt_00179, SWS_EthSwt_00234
SRS_ETH_00122	-	SWS_EthSwt_00087, SWS_EthSwt_00092, SWS_EthSwt_00126, SWS_EthSwt_00127, SWS_EthSwt_00136, SWS_EthSwt_00183, SWS_EthSwt_00194, SWS_EthSwt_00196
SRS_Eth_00122	Persistent storage of	SWS_EthSwt_00192

	configurations	
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_00352, SWS_EthSwt_00356
SRS_ETH_00125	-	SWS_EthSwt_00240, SWS_EthSwt_00241, SWS_EthSwt_00242, SWS_EthSwt_00243, SWS_EthSwt_00244, SWS_EthSwt_00245, SWS_EthSwt_00282, SWS_EthSwt_00378
SRS_Eth_00125	The Ethernet Switch Driver shall support switch frame management	SWS_EthSwt_91002, SWS_EthSwt_91003, SWS_EthSwt_91004, SWS_EthSwt_91005, SWS_EthSwt_91006, SWS_EthSwt_91007, SWS_EthSwt_91008, SWS_EthSwt_91009, SWS_EthSwt_91010, SWS_EthSwt_91011, SWS_EthSwt_91028
SRS_ETH_00126	-	SWS_EthSwt_00181
SRS_Eth_00126	Independent reset of host ECU and switch hardware	SWS_EthSwt_00292
SRS_ETH_00128	-	SWS_EthSwt_00106, SWS_EthSwt_00199, SWS_EthSwt_00372, SWS_EthSwt_00373
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
SRS_ETH_00458	-	SWS_EthSwt_00128

7 Functional specification

7.1 Ethernet BSW stack

As part of the AUTOSAR Layered Software Architecture according to Figure 3, the Ethernet BSW modules also form a layered software stack.
Figure 3depicts 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 [9]). Thus, the configuration as shown in

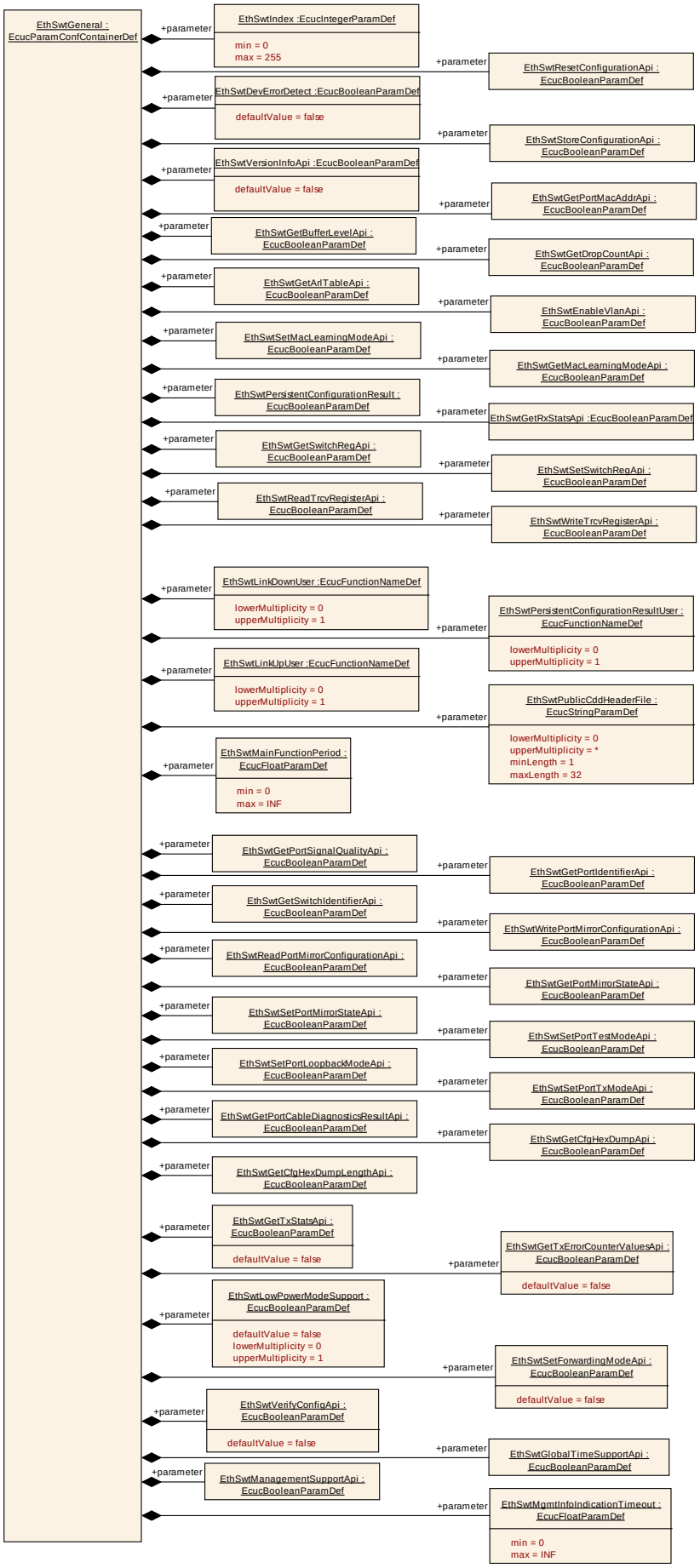

EthSwtGeneral : EcucParamConfContainerDef
+parameter
EthSwtIndex :EcucIntegerParamDef
min = 0
max = 255
EthSwtResetConfigurationApi : EcucBooleanParamDef
EthSwtDevErrorDetect :EcucBooleanParamDef
defaultValue = false
EthSwtStoreConfigurationApi : EcucBooleanParamDef
EthSwtVersionInfoApi :EcucBooleanParamDef
defaultValue = false
EthSwtGetPortMacAddrApi : EcucBooleanParamDef
EthSwtGetBufferLevelApi : EcucBooleanParamDef
EthSwtGetDropCountApi : EcucBooleanParamDef
EthSwtGetArlTableApi : EcucBooleanParamDef
EthSwtEnableVlanApi : EcucBooleanParamDef
EthSwtSetMacLearningModeApi : EcucBooleanParamDef
EthSwtGetMacLearningModeApi : EcucBooleanParamDef
EthSwtPersistentConfigurationResult : EcucBooleanParamDef
EthSwtGetRxStatsApi :EcucBooleanParamDef
EthSwtGetSwitchRegApi : EcucBooleanParamDef
EthSwtSetSwitchRegApi : EcucBooleanParamDef
EthSwtReadTrcvRegisterApi : EcucBooleanParamDef
EthSwtWriteTrcvRegisterApi : EcucBooleanParamDef
EthSwtLinkDownUser :EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
EthSwtPersistentConfigurationResultUser : EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
EthSwtLinkUpUser :EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
EthSwtPublicCddHeaderFile : EcucStringParamDef
lowerMultiplicity = 0
upperMultiplicity = *
minLength = 1
maxLength = 32
EthSwtMainFunctionPeriod : EcucFloatParamDef
min = 0
max = INF
EthSwtGetPortSignalQualityApi : EcucBooleanParamDef
EthSwtGetPortIdentifierApi : EcucBooleanParamDef
EthSwtGetSwitchIdentifierApi : EcucBooleanParamDef
EthSwtWritePortMirrorConfigurationApi : EcucBooleanParamDef
EthSwtReadPortMirrorConfigurationApi : EcucBooleanParamDef
EthSwtGetPortMirrorStateApi : EcucBooleanParamDef
EthSwtSetPortMirrorStateApi : EcucBooleanParamDef
EthSwtSetPortTestModeApi : EcucBooleanParamDef
EthSwtSetPortLoopbackModeApi : EcucBooleanParamDef
EthSwtSetPortTxModeApi : EcucBooleanParamDef
EthSwtGetPortCableDiagnosticsResultApi : EcucBooleanParamDef
EthSwtGetCfgHexDumpApi : EcucBooleanParamDef
EthSwtGetCfgHexDumpLengthApi : EcucBooleanParamDef
EthSwtGetTxStatsApi : EcucBooleanParamDef
defaultValue = false
EthSwtGetTxErrorCounterValuesApi : EcucBooleanParamDef
defaultValue = false
EthSwtLowPowerModeSupport : EcucBooleanParamDef
defaultValue = false
lowerMultiplicity = 0
upperMultiplicity = 1
EthSwtSetForwardingModeApi : EcucBooleanParamDef
defaultValue = false
EthSwtVerifyConfigApi : EcucBooleanParamDef
defaultValue = false
EthSwtGlobalTimeSupportApi : EcucBooleanParamDef
EthSwtManagementSupportApi : EcucBooleanParamDef
EthSwtMgmtInfoIndicationTimeout : EcucFloatParamDef
min = 0
max = INF

Figure 12 has to be written to the switch device.

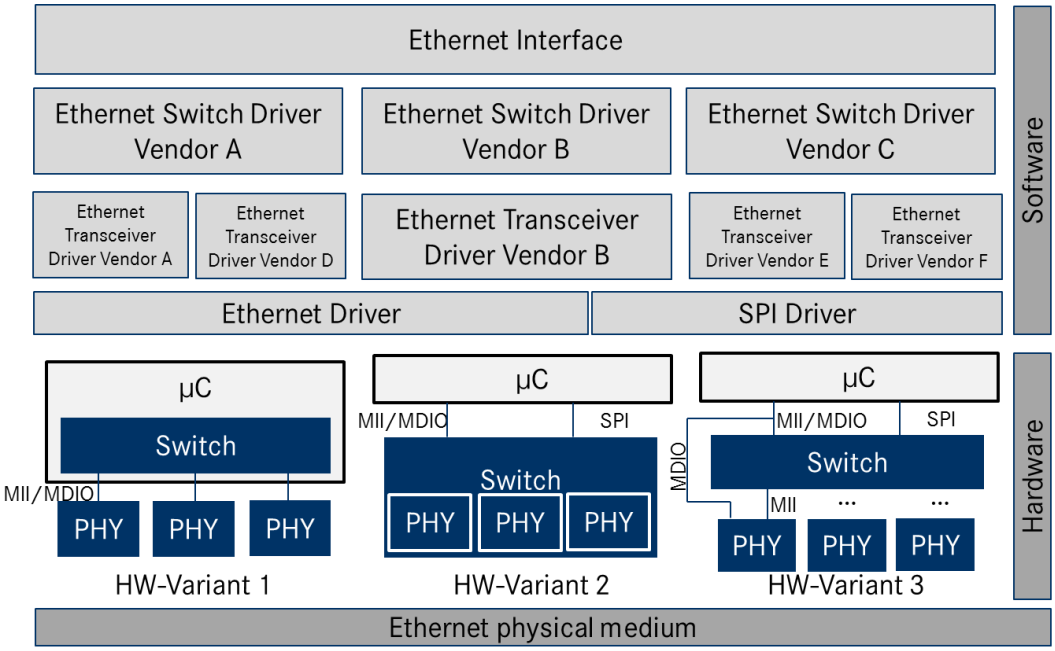


Figure 3 Basic Structure of the Ethernet BSW stack. (Note: The different hardware variants are alternative setups)

7.1.1 Indexing scheme

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

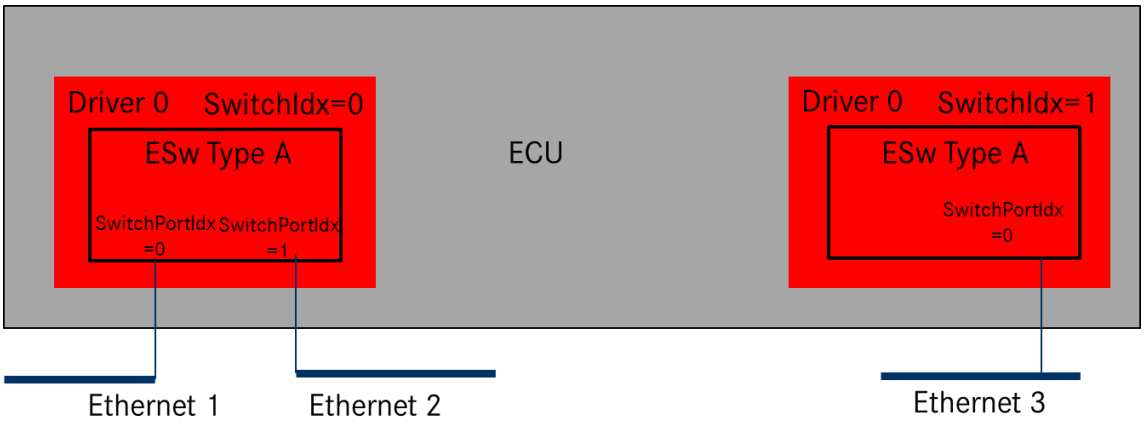


Figure 4 Ethernet Switch Driver indexing scheme

[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)

[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 (EthIf) 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.

7.1.2 Functional Description

7.1.2.1 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 5. 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.

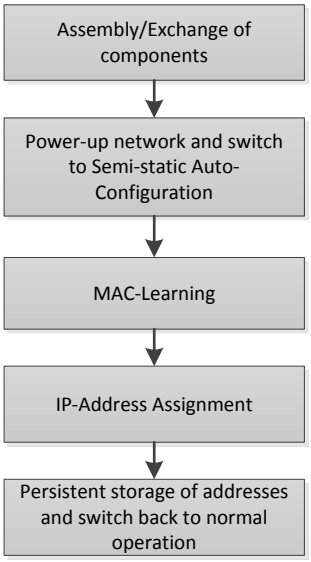


Figure 5 Learning Process

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 6). 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 3), 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.

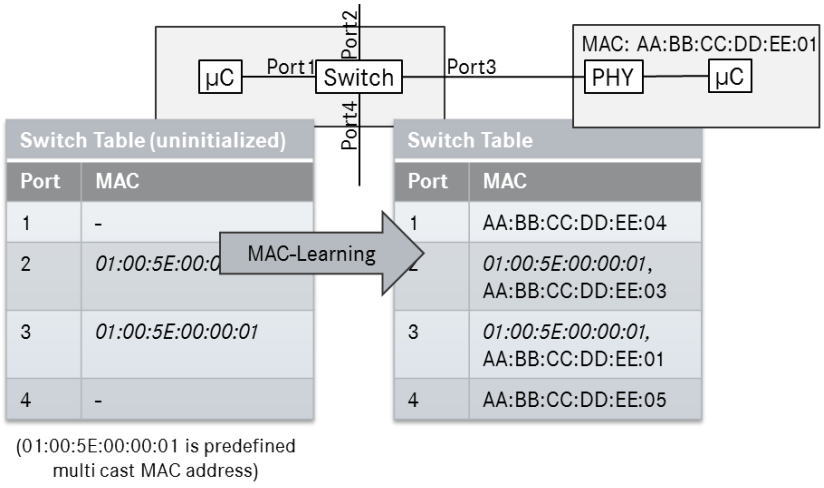


Figure 6 MAC-learning within the switch

IP-Address Assignment: In this phase, ECUs without a predefined IP-address will start to acquire an IP-address via DHCP (cf. Figure 7). 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.

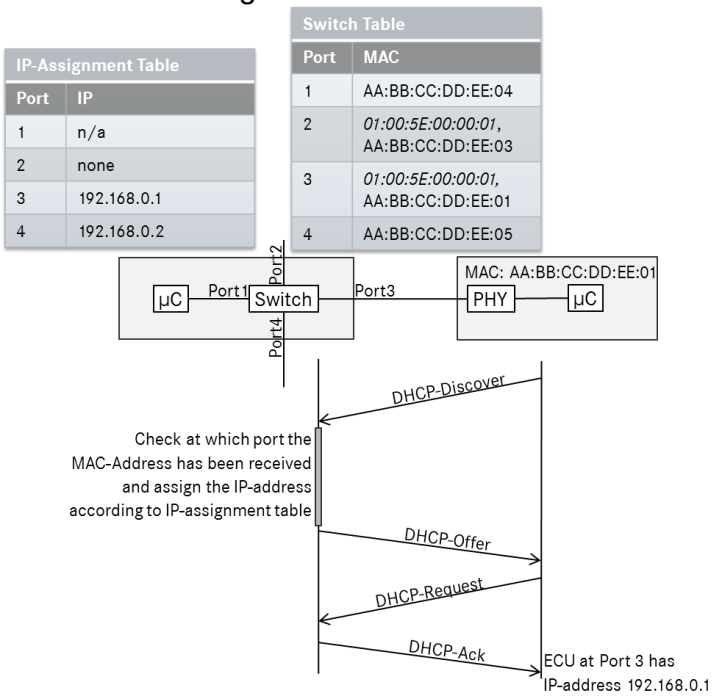


Figure 7 IP-address assignment via DHCP

[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 GetPortMacAddr().](SRS_ETH_00087)

7.1.2.2 Configuration of Egress Port Structure

As shown in Figure 8, 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 8 are meant as an example. Other structures with different FIFO numbers are possible.

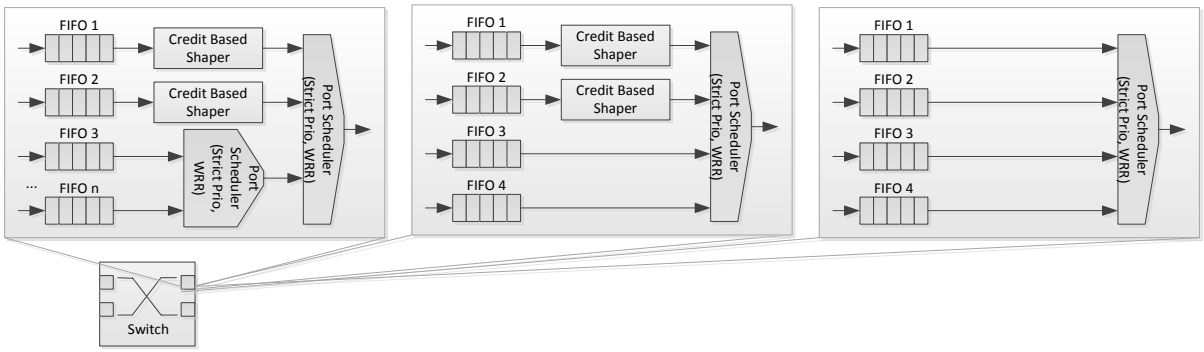


Figure 8 Ethernet Egress Port Structure

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.

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.

1. 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

2. 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.

3. 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.

7.1.2.3 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 EthSwtPort:EthSwtPortVlanMembership where the EthSwtPortVlanDefaultPriority and EthSwtPortVlanForwardingType are configured.

7.1.2.4 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.

7.1.2.5 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)

7.1.2.6 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

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 ECUC_EthSwt_00057 to ECUC_EthSwt_00059.
⌋ (SRS_ETH_00121)

7.1.2.7 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 ECUC_EthSwt_00023).
⌋ (SRS_ETH_00121)

7.1.2.8 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.

7.1.2.9 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)

7.1.2.10 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 [7] .

[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)

[SWS_EthSwt_00282]⌈ If the management information for reception will not be available within the time specified in the configuration parameter `EthSwtMgmtInfoIndicationTimeout` , the pointer `MgmtInfoPtr` shall be set to NULL before calling the function `EthIf_SwitchMgmtInfoIndication()`.⌋ (SRS_ETH_00125)

7.1.2.11 Global Time support

For more details regarding time measurement with Switches, please refer to [8].

[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_00244]⌈ If EthSwt_EnableTimeStamp is called for a SwitchIdx, the switch driver shall enable the time-stamping for all his ports except the ports where EthSwtPortRole is set to ETHSWT_UP_LINK_PORT and if EthSwtPortTimeStampSupport is set to TRUE for this port.⌋(SRS_ETH_00125)

[SWS_EthSwt_00378]⌈ If EthSwt_PortEnableTimeStamp is called for a PortIdx, the switch driver shall enable the time-stamping for all this port if EthSwtPortRole is not set to ETHSWT_UP_LINK_PORT and 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.

7.1.2.12 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:

1. Stop frame forwarding by calling EthSwt_SetForwardingMode(FALSE).
2. Verify the switch configuration
3. In case the switch configuration is valid then frame forwarding shall be enabled by calling EthSwt_SetForwardingMode(TRUE).(if disabled in step 1)
4. 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.

7.1.2.13 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_GetCfgHexDump
- EthSwt_GetCfgHexDumpLength

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

7.1.2.14 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 EthSwtPort 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 "General Specification of Basic Software Modules" 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] Development Error Types⌈

Type or error	***Related error code***	***Value [hex]***
Invalid switch index	ETHSWT_E_INV_SWITCH_IDX	0x01
EthSwt module was not initialized	ETHSWT_E_NOT_INITIALIZED	0x02
Invalid pointer in parameter list	ETHSWT_E_INV_POINTER	0x03
Invalid API which is not available by another module	ETHSWT_E_INV_API	0x05
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

⌋ (SRS_BSW_00385)

7.2.2 Runtime Errors

There are no runtime errors.

7.2.3 Transient Faults

There are no transient errors.

7.2.4 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	If during initialization the switch cannot be configured and a ETHSWT_E_ACCESS error is reported by the API call. Before the initialization of the switch hardware is executed this condition can be reseted.
	Pass	If no ETHSWT_E_ACCESS is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

7.2.5 Extended Production Errors

[SWS_EthSwt_00137][

Error Name:	ETHSWT_E_BUFFEROVERRUN	
Short Description:	Dropped packet due to buffer overrun in switch	
Long Description:	Dropped packet due to buffer overrun in switch	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00138][

Error Name:	ETHSWT_E_CRC	
Short Description:	Dropped packet due to CRC error detected in switch	
Long Description:	Dropped packet due to CRC error detected in switch	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00139][

Error Name:	ETHSWT_E_DROPCOUNT
Short Description:	Dropped packet due to other reason than buffer overrun or CRC error
Long Description:	Dropped packet due to other reason than buffer overrun or CRC error
Recommended DTC:	N/A

Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00141][

Error Name:	ETHSWT_E_UNDERSIZEPCKT	
Short Description:	An undersized packet occurred	
Long Description:	An error due to the occurrence undersized packets which were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed. (see IETF RFC 1757)	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00142][

Error Name:	ETHSWT_E_OVERSIZEPCKT	
Short Description:	An undersized packet occurred	
Long Description:	An error due to the occurrence oversized packets which are longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed. (see IETF RFC 1757)	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00143][

Error Name:	ETHSWT_E_ALIGNMENT	
Short Description:	Alignment error of an Ethernet frame	
Long Description:	Alignment errors occur if packets are received and are not an integral number of octets in length and do not pass the CRC.	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	

Monitor Frequency	N/A
MIL illumination:	N/A

] (SRS_BSW_00385)

[SWS_EthSwt_00144][

Error Name:	ETHSWT_E_SQETEST	
Short Description:	SQE test error	
Long Description:	SQE test error according to IETF RFC1643 dot3StatsSQETestErrors	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00145][

Error Name:	ETHSWT_E_INDISCARD	
Short Description:	Discard of inbound packets	
Long Description:	This error occurs if inbound packets were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space. (see IETF RFC 2233 ifInDiscards)	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00146][

Error Name:	ETHSWT_E_INERROR	
Short Description:	Discard of inbound packets	
Long Description:	This error occurs if the total number of erroneous inbound packets is greater than zero	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00147][

Error Name:	ETHSWT_E_OUTDISCARD

Short Description:	Discard of inbound packets	
Long Description:	This error occurs if outbound packets were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. (see IETF RFC 2233 ifOutDiscards)	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00148][

Error Name:	ETHSWT_E_OUTERROR	
Short Description:	Discard of inbound packets	
Long Description:	This error occurs if the total number of erroneous outbound packets is greater than zero	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00149][

Error Name:	ETHSWT_E_SINGLECOLLISION	
Short Description:	Number of packets with a single collision	
Long Description:	Single collision frames: A count of successfully transmitted frames on a particular interface for which transmission is inhibited by exactly one collision. (see IETF RFC1643 dot3StatsSingleCollisionFrames)	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00150][

Error Name:	ETHSWT_E_MULTIPLECOLLISION
Short Description:	Number of packets with multiple collisions
Long Description:	Multiple collision frames: A count of successfully transmitted frames on a particular interface for which transmission is

	inhibited by more than one collision. (see IETF RFC1643 dot3StatsMultipleCollisionFrames)	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00151][

Error Name:	ETHSWT_E_DEFFEREDTRANSMISSION	
Short Description:	Number of packets which are deffered	
Long Description:	Number of deferred transmission: A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy. (see IETF RFC1643 dot3StatsDeferredTransmissions)	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

[SWS_EthSwt_00152][

Error Name:	ETHSWT_E_LATECOLLISION	
Short Description:	Number of packets with a late collision	
Long Description:	Number of late collisions: The number of times that a collision is detected on a particular interface later than 512 bit-times into the transmission of a packet. (see IETF RFC1643 dot3StatsLateCollisions)	
Recommended DTC:	N/A	
Detection Criteria:	Fail	If main function detects that the corresponding counter value is greater than zero, this error will be reported
	Pass	If no such error is reported.
Secondary Parameters:	N/A	
Time Required:	N/A	
Monitor Frequency	N/A	
MIL illumination:	N/A	

] (SRS_BSW_00385)

8 API specification

8.1 Imported types

This chapter lists all types included from the following files:

[SWS_EthSwt_00002] ⌈

Module	*Imported Type*
Dem	Dem_EventIdType
	Dem_EventStatusType
Eth_GeneralTypes	EthTrcv_BaudRateType
	EthTrcv_CableDiagResultType
	EthTrcv_DuplexModeType
	EthTrcv_LinkStateType
	EthTrcv_ModeType
	EthTrcv_PhyLoopbackModeType
	EthTrcv_PhyTestModeType
	EthTrcv_PhyTxModeType
	Eth_BufIdxType
	Eth_CounterType
	Eth_DataType
	Eth_MacVlanType
	Eth_RxStatsType
	Eth_TimeStampType
	Eth_TxErrorCounterValuesType
	Eth_TxStatsType
NvM	NvM_BlockIdType
	NvM_RequestResultType
Spi	Spi_AsyncModeType
	Spi_ChannelType
	Spi_DataBufferType
	Spi_NumberOfDataType
	Spi_SequenceType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋ ()

8.2 Type definitions

[SWS_EthSwt_00003] ⌈
EthSwt.h shall include Eth_GeneralTypes.h for include of general Ethernet stack type declarations. ⌋(SRS_BSW_00456)

[SWS_EthSwt_00004] ⌈
The types specified in SWS_EthernetSwitchDriver shall be declared in Eth_GeneralTypes.h⌋()

8.2.1 EthSwt_StateType

[SWS_EthSwt_00123] ⌈

Name:	EthSwt_StateType		
Type:	Enumeration		
Range:	ETHSWT_STATE_UNINIT	0x00	Driver is not yet configured
	ETHSWT_STATE_INIT	0x01	Driver is configured
	ETHSWT_STATE_ACTIVE	0x02	Driver is active
Description:	Status supervision used for Development Error Detection. The state shall be available for debugging.		

⌋ (SRS_ETH_00086)

8.2.2 EthSwt_ConfigType

[SWS_EthSwt_00165] ⌈

Name:	EthSwt_ConfigType		
Type:	Structure		
Element:	void	implementation specific	--
Description:	Implementation specific structure of the post build configuration.		

⌋ (SRS_ETH_00086)

8.2.3 EthSwt_MacLearningType

[SWS_EthSwt_00227] ⌈

Name:	EthSwt_MacLearningType		
Type:	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		

⌋ (SRS_ETH_00086)

8.2.4 EthSwt_MgmtInfoType

[SWS_EthSwt_91002] ⌈

Name:	EthSwt_MgmtInfoType		
Type:	Structure		
Element:	uint8	SwitchIdx	Switch index
	uint8	SwitchPortIdx	Port index of the switch
Description:	Type for holding the management information received/transmitted on Switches (ports).		

⌋ (SRS_Eth_00125)

8.2.5 EthSwt_PortMirrorCfgType

[SWS_EthSwt_91017] ⌈

Name:	EthSwt_PortMirrorCfgType		
Type:	Structure		
Element:	uint8	SetSelection	specifies the type selection 0x00== free configuration, 0x01 - 0x0FF == set number
	uint8	TrafficDirection	specifies whether the direction is Ingress (0) or Egress (1)
	uint8[6]	srcMacAddrFilter	Specifies the source MAC address [0..255,0..255,0..255,0..255,0..255,0..255] that should be mirrored
	uint8[6]	dstMacAddrFilter	Specifies the source MAC address [0..255,0..255,0..255,0..255,0..255,0..255] that should be mirrored
	uint16	VlanIdFilter	Specifies the VLAN address 0..65535 that should be mirrored
	uint8	MirroringPacketDivider	Divider if only a subset of received frames should be mirrored. E.g. MirroringPacketDivider = 2 means every second frames is mirrored
	uint8	MirroringMode	specifies the mode how the mirrored traffic should be tagged : 0x00 == No VLAN retagging; 0x01 == VLAN retagging; 0x03 == VLAN Double tagging
	uint16	MirroringTimeout	specifies a time constant in seconds after the mirroring configuration should be resumed (e.g. ensuring that mirroring is not endlessly active).
Description:	In case of port mirroring it shall be possible to set up port mirroring configurations. It shall be possible to store different filter packages in the Ethernet switch. The parameter SetSelection set the index for the filter package (0x01 .. 0xFF). Filter packages are accessible by the index. In case SetSelection is set to 0x00, the package use the a free configuration.		

⌋ ()

8.2.6 EthSwt_PortMirrorStateType

[SWS_EthSwt_91020] ⌈

Name:	EthSwt_PortMirrorStateType		
Type:	Enumeration		
Range:	PORT_MIRROR_DISABLED	0x00	port mirroring disabled
	PORT_MIRROR_ENABLED	0x01	port mirroring enabled
Description:	It shall be possible to set and get the state (enable / disable) of the ehternet switch port.		

⌋ ()

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	

⌋ (SRS_ETH_00086)

[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 the component from ETHSWT_STATE_UNINIT to ETHSWT_STATE_INIT. ⌋ (SRS_BSW_00101)

[SWS_EthSwt_00009] ⌈
If development error detection is enabled: the function EthSwt_Init shall check the parameter CfgPtr for being valid, i.e. not Null pointer. If the check fails, the function shall raise the development error ETHSWT_E_INV_POINTER and return E_NOT_OK. In case of variant pre-compile, NULL_PTR is allowed.⌋(SRS_BSW_323, SRS_BSW_369)

8.3.2 EthSwt_SwitchInit

[SWS_EthSwt_00010] ⌈

Service name:	EthSwt_SwitchInit	

Syntax:	`Std_ReturnType EthSwt_SwitchInit(`
	`  uint8 SwitchIdx`
	`)`	

Service ID[hex]:	0x02	
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: success E_NOT_OK: switch could not be initialized
Description:	Initializes the indexed swtich with a given configuration for the switch index	

⌋ (SRS_ETH_00086)

[SWS_EthSwt_00011][
EthSwt_SwitchInit shall:

- Configure all configuration parameters (e.g. port structure, VLAN configuration, ...) at all ports of the switch and the switch itself.
- Perform a soft reset, i.e. resetting the switch via register setting not via a reset pin. This is hardware dependent and might not be supported by all switch devices.
- After resetting the switch and EthSwtLowPowerModeSupport set to TRUE, the Ethernet switch shall enter an inactive or low power mode. If EthSwtLowPowerModeSupport is not defined or set to FALSE the Ethernet switch shall enter an active state
  ] (SRS_BSW_00101)

[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 "ETHTRCV_MODE_DOWN" during initialization.] ()

[SWS_EthSwt_00012][
EthSwt_SwitchInit shall change the state of the component from ETHSWT_STATE_INIT to ETHSWT_STATE_ACTIVE.] (SRS_BSW_00101)

[SWS_EthSwt_00013] [
If development error detection is enabled: EthSwt_SwitchInit shall check that the service EthSwt_Init was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00014] [
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_SwitchInit shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00016][
The function EthSwt_SwitchInit shall check the access to the Ethernet controller, 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 production error ETHSWT_E_ACCESS and return E_NOT_OK.](SRS_BSW_00386)

8.3.3 EthSwt_SetSwitchPortMode

[SWS_EthSwt_00018] [

Service name:	EthSwt_SetSwitchPortMode
Syntax:	`Std_ReturnType EthSwt_SetSwitchPortMode( uint8 SwitchIdx,`

	uint8 SwitchPortIdx, EthTrcv_ModeType PortMode)	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous /Asynchronous	
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	ETHTRCV_MODE_DOWN: disable the addressed port at the switch ETHTRCV_MODE_ACTIVE: enable the addressed port at the switch
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 PortMode
Description:	Enables/disables the indexed switch port	

] (SRS_ETH_00086)

[SWS_EthSwt_00019] [
The function EthSwt_SetSwitchPortMode shall put the indexed port of the switch in the specified mode by calling the function EthTrcv_SetTransceiverMode of the Ethernet Transceiver Driver.] (SRS_ETH_00118)

[SWS_EthSwt_00020] [
If development error detection is enabled: the function EthSwt_SetSwitchPortMode shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00021] [
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_SetSwitchPortMode shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00166] [
If development error detection is enabled and the parameter SwitchPortIdx is not valid,EthSwt_SetSwitchPortMode shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00022] [
The function EthSwt_SetSwitchPortMode shall be pre compile time configurable On/Off by the configuration parameter: EthTrcvSetTransceiverModeApi.] (SRS_BSW_00171)

[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 return E_NOT_OK and if development error tracing is activated by EthSwtDevErrorDetect the ETHSWT_E_INV_API shall be raised.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369, SRS_ETH_00118)

[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)

8.3.4 EthSwt_GetSwitchPortMode

[SWS_EthSwt_00025] ⌈

Service name:	EthSwt_GetSwitchPortMode	
Syntax:	`Std_ReturnType EthSwt_GetSwitchPortMode( uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_ModeType* SwitchModePtr )`	
Service ID[hex]:	0x04	
Sync/Async:	Synchronous /Asynchronous	
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	ETHTRCV_MODE_DOWN: the port of the switch is disabled ETHTRCV_MODE_ACTIVE: the port of the switch is enabled
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: The mode of the indexed switch port could not be obtained.
Description:	Obtains the mode of the indexed switch port	

⌋ (SRS_ETH_00086)

[SWS_EthSwt_00026] ⌈
The function EthSwt_GetSwitchPortMode
shall read the mode of the indexed port of the switch by calling the corresponding function EthTrcv_GetTransceiverMode of the Ethernet Transceiver Driver. ⌋ (SRS_ETH_00118)

[SWS_EthSwt_00027] ⌈
If development error detection is enabled: the function EthSwt_GetSwitchPortMode shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK. ⌋ (SRS_BSW_00406)

[SWS_EthSwt_00028] ⌈
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_GetSwitchPortMode shall raise the development error

ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00167] ⌈
If development error detection is enabled and the parameter SwitchPortIdx is not valid,EthSwt_GetSwitchPortMode shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK. ⌋ (SRS_BSW_323, SRS_BSW_369)

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

[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 return E_NOT_OK and if development error tracing is activated by EthSwtDevErrorDetect the ETHSWT_E_INV_API shall be raised.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369, SRS_ETH_00118)

8.3.5 EthSwt_StartSwitchPortAutoNegotiation

[SWS_EthSwt_00031] ⌈

Service name:	EthSwt_StartSwitchPortAutoNegotiation	
Syntax:	`Std_ReturnType EthSwt_StartSwitchPortAutoNegotiation( uint8 SwitchIdx, uint8 SwitchPortIdx )`	
Service ID[hex]:	0x05	
Sync/Async:	Asynchronous /Asynchronous	
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.
Description:	Starts the auto-negotiation of the indexed switch port	

⌋ (SRS_ETH_00086)

[SWS_EthSwt_00032] ⌈

The function EthSwt_StartSwitchPortAutoNegotiation shall restart the automatic negotiation of the transmission parameters used by calling the API EthTrcv_StartAutoNegotiation by the indexed transceiver.] (SRS_ETH_00087)

[SWS_EthSwt_00033] [
If development error detection is enabled: the function EthSwt_StartSwitchPortAutoNegotiation shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00034] [
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_StartSwitchPortAutoNegotiation shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00168] [
If development error detection is enabled and the parameter SwitchPortIdx is not valid,EthSwt_StartSwitchPortAutoNegotiation shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)

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

[SWS_EthSwt_00158] ⌈
The function EthSwt_StartSwitchPortAutonegotiation shall check whether the EthTrcv_StartAutoNegotiation() 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 return E_NOT_OK and if development error tracing is activated by EthSwtDevErrorDetect the ETHSWT_E_INV_API shall be raised.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369, SRS_ETH_00087)

8.3.6 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 /Asynchronous	
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	ETHSWT_LINK_STATE_DOWN: Switch port is disconnected ETHSWT_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
Description:	Obtains the link state of the indexed switch port	

⌋ (SRS_ETH_00086)

[SWS_EthSwt_00038] ⌈
The function EthSwt_GetLinkState shall read the current link state of the indexed switch port by calling the corresponding function EthTrcv_GetLinkState of the Ethernet Transceiver Driver. ⌋ (SRS_ETH_00118, SRS_ETH_00119)

[SWS_EthSwt_00039] ⌈
If development error detection is enabled: the function EthSwt_GetLinkState shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK. ⌋ (SRS_BSW_00406)

[SWS_EthSwt_00040] ⌈
If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_GetLinkState shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK. ⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00169] ⌈

If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_GetLinkState shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)

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

[SWS_EthSwt_00154] [
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 return E_NOT_OK and if development error tracing is activated by EthSwtDevErrorDetect the ETHSWT_E_INV_API shall be raised.]
(SRS_ETH_00118, SRS_ETH_00119, SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

8.3.7 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 /Asynchronous	
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
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: Baud rate of the indexed switch port could not be obtained
Description:	Obtains the baud rate of the indexed switch port	

] (SRS_ETH_00086)

[SWS_EthSwt_00045] [
The function EthSwt_GetBaudRate shall read the current baud rate of the indexed switch port by calling the corresponding function EthTrcv_GetBaudRate of the Ethernet Transceiver Driver.] (SRS_ETH_00118)

[SWS_EthSwt_00046] [

If development error detection is enabled: the function EthSwt_GetBaudRate shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00118)

[SWS_EthSwt_00047] [
If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_GetBaudRate shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00170] [
If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_GetBaudRate shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)

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

[SWS_EthSwt_00153] [
The function EthSwt_GetBaudRate shall check whether the EthTrcv_GetBaudRate() 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 return E_NOT_OK . If development error tracing is activated by EthSwtDevErrorDetect, EthSwt_GetBaudRate shall raise the development error ETHSWT_E_INV_API .] (SRS_ETH_00118, SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

8.3.8 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 /Asynchronous	
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
Description:	Obtains the duplex mode of the indexed switch port	

] (SRS_ETH_00086)

[SWS_EthSwt_00052] [
The function EthSwt_GetDuplexMode shall read the current duplex mode of the indexed switch port by calling the function EthTrcv_ GetDuplexMode of the Ethernet Transceiver Driver.] (SRS_ETH_00118)

[SWS_EthSwt_00053] [
If development error detection is enabled: the function EthSwt_GetDuplexMode shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00054] [
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_GetDuplexMode shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00171] [
If development error detection is enabled and the parameter SwitchPortIdx is not valid,EthSwt_GetDuplexMode shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)

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

[SWS_EthSwt_00155] [
The function EthSwt_GetDuplexMode shall check whether the EthTrcv_GetDuplexMode() 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 return E_NOT_OK and if development error tracing is activated by EthSwtDevErrorDetect the ETHSWT_E_INV_API shall be raised.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369, SRS_ETH_00118)

8.3.9 EthSwt_GetPortMacAddr

[SWS_EthSwt_00060] [

Service name:	EthSwt_GetPortMacAddr

Syntax:	`Std_ReturnType EthSwt_GetPortMacAddr(`
	`  const uint8* MacAddrPtr,`
	`  const uint8* SwitchIdxPtr,`
	`  uint8* PortIdxPtr`

	)	
Service ID[hex]:	0x09	
Sync/Async:	Synchronous /Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	MacAddrPtr	MAC-address for which a switch port is searched over which the node with this MAC-address can be reached.
	SwitchIdxPtr	Pointer to the switch index
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.	

] (SRS_ETH_00086, SRS_ETH_00087)

[SWS_EthSwt_00061] [
The function EthSwt_GetPortMacAddr shall return the switch and port index over which the given MAC-address is reachable. 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_00062] [
If development error detection is enabled: the function EthSwt_GetPortMacAddr shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00064]
[If development error detection is enabled and the parameter MacAddrPtr is a NULL pointer, EthSwt_GetPortMacAddr shall raise the development error ETHSWT_E_INV_POINTER and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)

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

8.3.10 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 /Asynchronous	
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 numberOfElements 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.	

] (SRS_ETH_00086, 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 arlTableListPointer 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_00112] [
If development error detection is enabled: the function EthSwt_GetArlTable shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00237][If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_GetArlTable API shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

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

8.3.11 EthSwt_GetBufferLevel

[SWS_EthSwt_00079] ⌈

Service name:	EthSwt_GetBufferLevel	

Syntax:	`Std_ReturnType EthSwt_GetBufferLevel(`
	`  uint8 SwitchIdx,`
	`  uint32* SwitchBufferLevelPtr`
	`)`	

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous /Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout):	None	
Parameters (out):	SwitchBufferLevelPtr	The interpretation of this value is switch dependent
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: buffer level could not be obtained
Description:	Reads the buffer level of the corresponding switch. Whether this buffer level is one value for the entire switch (shared memory) or one value for each port at a switch is technology dependent. This API will be called, e.g. by a CDD	

⌋ (SRS_ETH_00086)

[SWS_EthSwt_00080] ⌈
The function EthSwt_GetBufferLevel shall read the buffer level of the currently used buffer of the switch. ⌋ (SRS_ETH_00119)

[SWS_EthSwt_00081] ⌈
If development error detection is enabled: the function EthSwt_GetBufferLevel shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK. ⌋ (SRS_BSW_00406)

[SWS_EthSwt_00082] ⌈
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_GetBufferLevel shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK. ⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00084] ⌈
The function EthSwt_GetBufferLevel shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetBufferLevelApi. ⌋ (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.	

] (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)

[SWS_EthSwt_00107] ⌈
If development error detection is enabled: the function EthSwt_ GetCounterValues shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK. ⌋ (SRS_BSW_00406)

[SWS_EthSwt_00108] ⌈
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_ GetCounterValues shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK. ⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00238]⌈ If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_ GetCounterValues shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK. ⌋ (SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00239]⌈ If development error detection is enabled and the parameter CounterPtr is a NULL pointer, EthSwt_ GetCounterValues shall raise the development error ETHSWT_E_INV_POINTER and return E_NOT_OK.⌋ (SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00109] ⌈
The function EthSwt_ GetCounterValues shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetDropCountApi. ⌋ (SRS_BSW_00171)

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 /Asynchronous	
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.	

⌋ (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_00200] ⌈
If development error detection is enabled: the function EthSwt_GetRxStats shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK. ⌋ (SRS_BSW_00406)
[SWS_EthSwt_00201] ⌈
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_GetRxStats shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK. ⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)
[SWS_EthSwt_00363]⌈ If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_ GetRxStats shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK. ⌋ (SRS_BSW_323, SRS_BSW_369)
[SWS_EthSwt_00364]⌈ If development error detection is enabled and the parameter RxStats is a NULL pointer, EthSwt_ GetRxStats shall raise the development error ETHSWT_E_INV_POINTER and return E_NOT_OK ⌋ (SRS_BSW_323, SRS_BSW_369)
[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.	

⌋ (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_00360]⌈ If development error detection is enabled: the function EthSwt_GetTxStats shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.⌋ (SRS_BSW_00406)

[SWS_EthSwt_00361]⌈ If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_GetTxStats shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00365]⌈ If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_ GetTxStats shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.⌋ (SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00366]⌈ If development error detection is enabled and the parameter TxStats is a NULL pointer, EthSwt_ GetTxStats shall raise the development error ETHSWT_E_INV_POINTER and return E_NOT_OK ⌋ (SRS_BSW_323, SRS_BSW_369)

[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.	

⌋ (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_00367]⌈ If development error detection is enabled: the function EthSwt_GetTxErrorCounterValues shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.⌋ (SRS_BSW_00406)

[SWS_EthSwt_00368]⌈ If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_GetTxErrorCounterValues shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00369]⌈ If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_ GetTxErrorCounterValues shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.⌋ (SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00371]⌈ If development error detection is enabled and the parameter TxErrorCounterValues is a NULL pointer, EthSwt_ GetTxErrorCounterValues shall raise the development error ETHSWT_E_INV_POINTER and return E_NOT_OK ⌋ (SRS_BSW_323, SRS_BSW_369)

[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 /Asynchronous	
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	

⌋ (SRS_ETH_00086)

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

[SWS_EthSwt_00208] ⌈
If development error detection is enabled: the function EthSwt_GetSwitchRegs shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK. ⌋ (SRS_BSW_00406)

[SWS_EthSwt_00209] ⌈
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_GetSwitchReg shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK. ⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[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 /Asynchronous	
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	

] (SRS_ETH_00086)

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

[SWS_EthSwt_00213] [
If development error detection is enabled: the function EthSwt_SetSwitchRegs shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00214] [
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_SetSwitchReg shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[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 /Asynchronous

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: drop counter could not be obtained
Description:	Generic API for reading the content of a transceiver register	

] (SRS_ETH_00086)

[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_00218] [
If development error detection is enabled: the function EthSwt_ReadTrcvRegister shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00219] [
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_ReadTrcvRegister shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[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 /Asynchronous	
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: drop counter could not be obtained
Description:	Generic API for writing the content of a transceiver register	

] (SRS_ETH_00086)

[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_00223] [
If development error detection is enabled: the function EthSwt_WriteTrcvRegister shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00224] [
If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_WriteTrcvRegister shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[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 /Asynchronous	
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.	

] (SRS_ETH_00086)

[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_00174] ⌈
If development error detection is enabled: the function EthSwt_EnableVlan shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK. ⌋ (SRS_BSW_00406)

[SWS_EthSwt_00175] ⌈
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_EnableVlan shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK. ⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00176] ⌈
If development error detection is enabled and the parameter SwitchPortIdx is not valid,EthSwt_EnableVlan shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK. ⌋ (SRS_BSW_323, SRS_BSW_369)

[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 /Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success

		E_NOT_OK: Configuration could not be persistently stored
Description:	Stores the configuration of the learned MAC/Port tables of a switch in a persistent manner and will be used by e.g. CDD.	

] (SRS_ETH_00086, SRS_ETH_00087)

[SWS_EthSwt_00087] ⌈
The function EthSwt_StoreConfiguration shall 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.⌋ (SRS_ETH_00087, SRS_ETH_00122)

[SWS_EthSwt_00088] ⌈
If development error detection is enabled: the function EthSwt_StoreConfiguration shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK. ⌋ (SRS_BSW_00406)

[SWS_EthSwt_00089] ⌈
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_StoreConfiguration shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK. ⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[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 /Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: configuration could be persistently reset
Description:	Resets the configuration of the learned MAC/Port tables of a switch in a persistent manner and will be used by e.g. CDD. The statically configured entries shall still remain.	

] (SRS_ETH_00086, SRS_ETH_00087)

[SWS_EthSwt_00092] ⌈
The function EthSwt_ResetConfiguration shall 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. ⌋ (SRS_ETH_00122, SRS_ETH_00087)

[SWS_EthSwt_00093] ⌈
If development error detection is enabled: the function EthSwt_ResetConfiguration shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK. ⌋ (SRS_BSW_00406)

[SWS_EthSwt_00094] ⌈
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_ResetConfiguration shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK. ⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[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 /Asynchronous	
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.	

⌋ (SRS_ETH_00086, SRS_ETH_00087)

[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_00184] ⌈
If development error detection is enabled: the function EthSwt_SetMacLearningMode shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK. ⌋ (SRS_BSW_00406)

[SWS_EthSwt_00185] ⌈
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_SetMacLearningMode shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK. ⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[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 /Asynchronous	
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	

⌋ (SRS_ETH_00086, 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_00189] ⌈
If development error detection is disabled: the function EthSwt_GetMacLearningMode shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK. ⌋ (SRS_BSW_00406)

[SWS_EthSwt_00190] ⌈
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_GetMacLearningMode shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK. ⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[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( uint8 ServiceId, NvM_RequestResultType JobResult )`	
Service ID[hex]:	0x17	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ServiceId	Unique Service ID of NVRAM manager service
	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.	

⌋ (SRS_ETH_00086, SRS_ETH_00087)

[SWS_EthSwt_00126] ⌈

The function EthSwt_NvmSingleBlockCallback shall be called by the NVRAMManager 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. 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. ⌋ (SRS_BSW_00369, SRS_ETH_00458)

[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.	

⌋ (SRS_ETH_00086)

[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_MgmtInit

[SWS_EthSwt_91003] ⌈

Service name:	EthSwt_MgmtInit

Syntax:	`void EthSwt_MgmtInit(`
	`  void`
	`)`

Service ID[hex]:	0x22
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Function initializes additional Switch Management handling.

⌋ (SRS_Eth_00125)

[SWS_EthSwt_00246]⌈ The function `EthSwt_MgmtInit()` shall be pre compile time configurable `ON`/`OFF` by the configuration parameter: `EthSwtManagementSupportApi`.⌋ (SRS_BSW_00171)

[SWS_EthSwt_00247]⌈ If default error detection is enabled: the function `EthSwt_MgmtInit()` shall check that the service `EthSwt_Init()` was previously called.
If the check fails, the function `EthSwt_MgmtInit()` shall raise the development error `ETHSWT_E_NOT_INITIALIZED`.⌋ (SRS_BSW_00406)

8.3.28 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().	

] (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)

[SWS_EthSwt_00250][If default error detection is enabled: the function `EthSwt_EthRxProcessFrame()` shall check that the service `EthSwt_Init()` was previously called.
If the check fails, the function `EthSwt_EthRxProcessFrame()` shall raise the development error `ETHSWT_E_NOT_INITIALIZED` and return E_NOT_OK..] (SRS_BSW_00406)

[SWS_EthSwt_00251][If default error detection is enabled: the function `EthSwt_EthRxProcessFrame()` shall check the parameter `CtrlIdx` for being valid. If the check fails, the function `EthSwt_EthRxProcessFrame()` shall raise the development error `ETHSWT_E_INV_CTRL_IDX` and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00252][If default error detection is enabled: the function `EthSwt_EthRxProcessFrame()` shall check the parameter `BufIdx` for being valid.
If the check fails, the function `EthSwt_EthRxProcessFrame()` raise the development error ETHSWT_E_INV_PARAM and return E_NOT_OK] (SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00284][If default error detection is enabled: the function `EthSwt_EthRxProcessFrame()` shall check the parameter `DataPtr`, `LengthPtr` and `IsMgmtFrameOnlyPtr` for being valid.
If the check fails, the function `EthSwt_EthRxProcessFrame()` shall raise the default error `ETHSWT_E_INV_POINTER` and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)

8.3.29 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().	

⌋ (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)

[SWS_EthSwt_00254]⌈ If default error detection is enabled: the function `EthSwt_EthRxFinishedIndication()` shall check that the service `EthSwt_Init()` was previously called.
If the check fails, the function `EthSwt_EthRxFinishedIndication()` shall raise the development error `ETHSWT_E_NOT_INITIALIZED` and return E_NOT_OK.⌋ (SRS_BSW_00406)

[SWS_EthSwt_00255]⌈ If default error detection is enabled: the function EthSwt_EthRxFinishedIndication() shall check the parameter CtrlIdx for being valid.
If the check fails, the function `EthSwt_EthRxFinishedIndication()` shall raise the development error `ETHSWT_E_INV_CTRL_IDX` and return E_NOT_OK.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00256]⌈ If default error detection is enabled: the function EthSwt_EthRxFinishedIndication() shall check the parameter BufIdx for being valid.
If the check fails, the function `EthSwt_EthRxFinishedIndication()` shall raise the development error ETHSWT_E_INV_PARAM and return E_NOT_OK ⌋ (SRS_BSW_323, SRS_BSW_369)

8.3.30 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().	

] (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)

[SWS_EthSwt_00258][If default error detection is enabled: the function EthSwt_EthTxPrepareFrame() shall check that the service EthSwt_Init() was previously called.
If the check fails, the function `EthSwt_EthTxPrepareFrame()` shall raise the development error `ETHSWT_E_NOT_INITIALIZED` and return E_NOT_OK] (SRS_BSW_00406)

[SWS_EthSwt_00259][If default error detection is enabled: the function EthSwt_EthTxPrepareFrame() shall check the parameter CtrlIdx for being valid.
If the check fails, the function `EthSwt_EthTxPrepareFrame()` shall raise the development error `ETHSWT_E_INV_CTRL_IDX`. and return E_NOT_OK] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00260][If default error detection is enabled: the function EthSwt_EthTxPrepareFrame() shall check the parameter BufIdx for being valid.
If the check fails, the function `EthSwt_EthTxPrepareFrame()` shall raise the development error ETHSWT_E_INV_PARAM and return E_NOT_OK] (SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00283][If default error detection is enabled: the function `EthSwt_EthTxPrepareFrame()` shall check the parameter `DataPtr` and `LengthPtr` for being valid.
If the check fails, the function `EthSwt_EthTxPrepareFrame()` shall raise the default error `ETHSWT_E_INV_POINTER`.] (SRS_BSW_323, SRS_BSW_369)

8.3.31 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.	

] (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)

[SWS_EthSwt_00262][If default error detection is enabled: the function `EthSwt_EthTxAdaptBufferLength()` shall check that the service `EthSwt_Init()` was previously called.
If the check fails, the function `EthSwt_EthTxAdaptBufferLength()` shall raise the development error `ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK`.] (SRS_BSW_00406)

[SWS_EthSwt_00263][If default error detection is enabled: the function `EthSwt_EthTxAdaptBufferLength()` shall check the parameter `LengthPtr` for being valid.
If the check fails, the function `EthSwt_EthTxAdaptBufferLength()` shall raise the development error ETHSWT_E_INV_POINTER and return E_NOT_OK] (SRS_BSW_323, SRS_BSW_369)

8.3.32 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.	

⌋ (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)

[SWS_EthSwt_00265]⌈ If default error detection is enabled: the function `EthSwt_SetMgmtInfo()` shall check that the service `EthSwt_Init()` was previously called.
If the check fails, the function `EthSwt_SetMgmtInfo()` shall raise the development error `ETHSWT_E_NOT_INITIALIZED` and return E_NOT_OK.⌋ (SRS_BSW_00406)

[SWS_EthSwt_00266]⌈ f default error detection is enabled: the function `EthSwt_SetMgmtInfo()` shall check the parameter CtrlIdx for being valid.
If the check fails, the function `EthSwt_EthTxPrepareFrame()` shall raise the development error `ETHSWT_E_INV_CTRL_IDX` and return E_NOT_OK.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00267]⌈ If default error detection is enabled: the function `EthSwt_SetMgmtInfo()` shall check the parameter `BufIdx` for being valid.
If the check fails, the function `EthSwt_SetMgmtInfo()` shall raise the development error ETHSWT_E_INV_PARAM and return E_NOT_OK⌋ (SRS_BSW_323, SRS_BSW_369)

8.3.33 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.	

⌋ (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)

[SWS_EthSwt_00269]⌈ If default error detection is enabled: the function `EthSwt_EthTxProcessFrame()` shall check that the service `EthSwt_Init()` was previously called.
If the check fails, the function `EthSwt_EthTxProcessFrame()` shall raise the development error `ETHSWT_E_NOT_INITIALIZED` and return E_NOT_OK. ⌋ (SRS_BSW_00406)

[SWS_EthSwt_00270]⌈ If default error detection is enabled: the function `EthSwt_EthTxProcessFrame()` shall check the parameter `CtrlIdx` for being valid.
If the check fails, the function `EthSwt_EthTxProcessFrame()` shall raise the development error `ETHSWT_E_INV_CTRL_IDX` and return E_NOT_OK. ⌋ (SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00271]⌈ If default error detection is enabled: the function `EthSwt_EthTxProcessFrame()` shall check the parameter `BufIdx` for being valid.

If the check fails, the function `EthSwt_EthTxProcessFrame()` shall raise the development error `ETHIF_E_INV_PARAM` and return E_NOT_OK.⌋ (SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00272]⌈ If default error detection is enabled: the function `EthSwt_EthTxProcessFrame()` shall check the parameter `DataPtr` and `LengthPtr` for being valid.
If the check fails, the function `EthSwt_EthTxProcessFrame()` shall raise the development error `ETHSWT_E_INV_POINTER`.⌋ (SRS_BSW_323, SRS_BSW_369)

8.3.34 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.	

⌋ (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)

[SWS_EthSwt_00274]⌈ If default error detection is enabled: the function `EthSwt_EthTxFinishedIndication()` shall check that the service `EthSwt_Init()` was previously called.
If the check fails, the function `EthSwt_EthTxFinishedIndication()` shall raise the development error `ETHSWT_E_NOT_INITIALIZED` and return E_NOT_OK.⌋ (SRS_BSW_00406)

[SWS_EthSwt_00275]⌈ If default error detection is enabled: the function `EthSwt_EthTxFinishedIndication()` shall check the parameter `CtrlIdx` for being valid.
If the check fails, the function `EthSwt_EthTxFinishedIndication()` shall raise the development error `ETHSWT_E_INV_CTRL_IDX` and return E_NOT_OK.⌋ (SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00276]⌈ If default error detection is enabled: the function `EthSwt_EthTxFinishedIndication()` shall check the parameter `BufIdx` for being valid.
If the check fails, the function `EthSwt_EthTxFinishedIndication()` shall raise the development error `ETHIF_E_INV_PARAM` and return E_NOT_OK.⌋ (SRS_BSW_323, SRS_BSW_369)

8.3.35 EthSwt_EnableTimeStamping

[SWS_EthSwt_91011] ⌈

Service name:	EthSwt_EnableTimeStamping	
Syntax:	`Std_ReturnType EthSwt_EnableTimeStamping( uint8 SwitchIdx, Eth_BufIdxType BufIdx, EthSwt_MgmtInfoType* MgmtInfoPtr )`	
Service ID[hex]:	0x30	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	BufIdx	Index of the message buffer, where Application expects egress time stamping
	MgmtInfoPtr	Management information
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 all ports of a Switch but on uplink ports between cascaded switches. 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.	

⌋ (SRS_Eth_00125)

[SWS_EthSwt_00277]⌈ The function `EthSwt_EnableTimeStamping()` shall be pre compile time configurable `ON`/`OFF` by the configuration parameter: EthSwtGlobalTimeSupportApi .⌋ (SRS_BSW_00171)

[SWS_EthSwt_00278]⌈ If default error detection is enabled: the function `EthSwt_EnableTimeStamping()` shall check that the service `EthSwt_Init()` was previously called.
If the check fails, the function `EthSwt_EnableTimeStamping()` shall raise the development error `ETHIF_E_NOT_INITIALIZED` and return E_NOT_OK..⌋ (SRS_BSW_00406)

[SWS_EthSwt_00279][If default error detection is enabled: the function `EthSwt_EnableTimeStamping()` shall check the parameter `SwitchIdx` for being valid.
If the check fails, the function `EthSwt_EnableTimeStamping()` shall raise the development error `ETHIF_E_INV_SWITCH_IDX` and return E_NOT_OK..] (SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00280][If default error detection is enabled: the function `EthSwt_EnableTimeStamping()` shall check the parameter `BufIdx` for being valid.
If the check fails, the function `EthSwt_EnableTimeStamping()` shall raise the development error `ETHIF_E_INV_PARAM` and return E_NOT_OK..] (SRS_BSW_323, SRS_BSW_369)

8.3.36 EthSwt_PortEnableTimeStamp

[SWS_EthSwt_91028] [

Service name:	EthSwt_PortEnableTimeStamp	

Syntax:	`Std_ReturnType EthSwt_PortEnableTimeStamp(`
	`  uint8 SwitchIdx,`
	`  uint8 PortIdx,`
	`  Eth_BufIdxType BufIdx,`
	`  EthSwt_MgmtInfoType* MgmtInfoPtr`
	`)`	

Service ID[hex]:	0x40	
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
	BufIdx	Index of the message buffer, where Application expects egress time stamping
	MgmtInfoPtr	Management information
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. 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.	

] (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)

[SWS_EthSwt_00380][ ][If default error detection is enabled: the function EthSwt_PortEnableTimeStamp() shall check that the service EthSwt_Init() was previously called.

If the check fails, the function EthSwt_PortEnableTimeStamp() shall raise the development error ETHIF_E_NOT_INITIALIZED and return E_NOT_OK.⌋ (SRS_BSW_00406)

[SWS_EthSwt_00381]⌈ If default error detection is enabled: the function EthSwt_PortEnableTimeStamp() shall check the parameter SwitchIdx for being valid. If the check fails, the function EthSwt_PortEnableTimeStamp() shall raise the development error ETHIF_E_INV_SWITCH_IDX and return E_NOT_OK.⌋ (SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00383]⌈ If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_PortEnableTimeStamp() shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK. ⌋ ()

[SWS_EthSwt_00382]⌈ If default error detection is enabled: the function EthSwt_PortEnableTimeStamp() shall check the parameter BufIdx for being valid. If the check fails, the function EthSwt_PortEnableTimeStamp() shall raise the development error ETHIF_E_INV_PARAM and return E_NOT_OK.⌋ (SRS_BSW_323, SRS_BSW_369)

8.3.37 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.	

⌋ (SRS_Eth_00086)

[SWS_EthSwt_00285]⌈ If development error detection is enabled: the function EthSwt_VerifyConfig shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.⌋ (SRS_BSW_00406)

[SWS_EthSwt_00286]⌈ If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_VerifyConfig shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

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

8.3.38 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.	

] ()

[SWS_EthSwt_00289][If development error detection is enabled: the function EthSwt_SetForwardingMode shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00290][If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_SetForwardingMode shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

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

8.3.39 EthSwt_GetPortSignalQuality

[SWS_EthSwt_91014] [

Service name:	EthSwt_GetPortSignalQuality

Syntax:	`Std_ReturnType EthSwt_GetPortSignalQuality(`
	`  uint8 SwitchIdx,`
	`  uint8 PortIdx,`
	`  uint8* 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 in percent 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:	Obtains the current signal quality of the link of the indexed Ethernet switch port. This function shall obtain the signal quality of Ethernet transceiver referenced by the EthSwtPort by calling EthTrcv_GetPhySignalQuality(). If no transceiver is referenced the signal quality shall be set to 0xFF.	

] ()

[SWS_EthSwt_00293][The function EthSwt_GetPortSignalQuality shall return the value of the signal quality of the indexed Ethernet transceiver that is connected to the indexed port. The indexed port is connected to the indexed EthSwt. The value is provided by calling the function EthTrcv_GetPhySignalQuality of the Ethernet Transceiver Driver.] (SRS_Eth_00123)

[SWS_EthSwt_00294][If development error detection is enabled: the function EthSwt_GetPortSignalQuality shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00295][If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_GetPortSignalQuality shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00296][If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_GetPortSignalQuality shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)

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

[SWS_EthSwt_00298][The function EthSwt_GetPortSignalQuality shall check if the corresponding EthTrcv API EthTrcv_GetPhySignalQuality() of the indexed transceiver driver for the given SwitchPortIdx is available. If this is not the case, the function shall return E_NOT_OK and if development error tracing is activated by EthSwtDevErrorDetect the ETHSWT_E_INV_API shall be raised.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369, SRS_ETH_00118)

8.3.40 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 read (i.e. no Ethernet transceiver is available for this Ethernet switch port)
Description:	This function shall provide the OUI of Ethernet transceiver referenced by the EthSwtPort. The function shall call EthTrcv_GetPhyIdentifier. The OUI has a size of 24 bit. If a Ethernet transceiver can provide the OUI the 8 most significant bits of the OUI shall be set to 0x00xxxxxx. If no tranceiver is referenced by the EthSwtPort the 8 most significant bits of the OUI shall be set to 0xFFxxxxxx.	

] ()

[SWS_EthSwt_00299][The function EthSwt_GetPortIdentifier shall return the value of the organizationally unique identifier of the indexed Ethernet transceiver that is connected to the indexed port. The indexed port is connected to the indexed EthSwt. The value is provided by calling the function EthTrcv_GetPhyIdentifier of the Ethernet Transceiver Driver.] (SRS_Eth_00123)

[SWS_EthSwt_00300][If development error detection is enabled: the function EthSwt_GetPortIdentifier shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00301][If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_GetPortIdentifier shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00302][If development error detection is enabled and the parameter SwitchPortIdx is not valid,EthSwt_GetPortIdentifier shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)

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

[SWS_EthSwt_00304][The function EthSwt_GetPortIdentifier shall check if the corresponding EthTrcv API EthTrcv_GetPhyIdentifier() of the indexed transceiver driver for the given SwitchPortIdx is available. If this is not the case, the function shall return E_NOT_OK and if development error tracing is activated by

EthSwtDevErrorDetect the ETHSWT_E_INV_API shall be raised.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369, SRS_ETH_00118)

8.3.41 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.	

⌋ ()

[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_00306]⌈ If development error detection is enabled: the function EthSwt_GetSwitchIdentifier shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.⌋ (SRS_BSW_00406)

[SWS_EthSwt_00307]⌈ If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_GetSwitchIdentifier shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

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

8.3.42 EthSwt_WritePortMirrorConfiguration

[SWS_EthSwt_91018] ⌈

Service name:	EthSwt_WritePortMirrorConfiguration
Syntax:	`Std_ReturnType EthSwt_WritePortMirrorConfiguration(`

	uint8 SwitchIdx, uint8 PortIdx, EthSwt_PortMirrorCfgType* const PortMirrorConfigurationPtr)	
Service ID[hex]:	0x36	
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
	PortMirrorConfigurationPtr	Pointer to the memory where the port configuration is stored.
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)
Description:	The given mirror configuration shall be written to the given Ethernet switch port.	

] ()

[SWS_EthSwt_00309][The function EthSwt_WritePortMirrorConfiguration shall write the port mirror configuration in the indexed Ethernet switch port.] (SRS_Eth_00123)

[SWS_EthSwt_00310][If development error detection is enabled: the function EthSwt_WritePortMirrorConfiguration shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00311][If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_WritePortMirrorConfiguration shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

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

8.3.43 EthSwt_ReadPortMirrorConfiguration

[SWS_EthSwt_91019] [

Service name:	EthSwt_ReadPortMirrorConfiguration	
Syntax:	Std_ReturnType EthSwt_ReadPortMirrorConfiguration(uint8 SwitchIdx, uint8 PortIdx, EthSwt_PortMirrorCfgType* PortMirrorConfigurationPtr)	
Service ID[hex]:	0x37	
Sync/Async:	Asynchronous	
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):	PortMirrorConfigurationPtr	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 mirror configuration of the given Ethernet switch port.	

⌋ ()

[SWS_EthSwt_00313]⌈ The function EthSwt_ReadPortMirrorConfiguration shall return the port mirror configuration of the indexed Ethernet switch port.⌋ (SRS_Eth_00123)

[SWS_EthSwt_00314]⌈ If development error detection is enabled: the function EthSwt_ReadPortMirrorConfiguration shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.⌋ (SRS_BSW_00406)

[SWS_EthSwt_00315]⌈ If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_ReadPortMirrorConfiguration shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00316]⌈ If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_ReadPortMirrorConfiguration shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK. ⌋ (SRS_BSW_323, SRS_BSW_369)

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

8.3.44 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 state shall be stored. 0x00 == port mirroring disabled, 0x01 == port mirroring enabled
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	

] ()

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

[SWS_EthSwt_00319][If development error detection is enabled: the function EthSwt_GetPortMirrorState shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00320][If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_GetPortMirrorState shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00321][If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_GetPortMirrorState shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)

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

8.3.45 EthSwt_SetPortMirrorState

[SWS_EthSwt_91022] [

Service name:	EthSwt_SetPortMirrorState	

Syntax:	`Std_ReturnType EthSwt_SetPortMirrorState(`
	`  uint8 SwitchIdx,`
	`  uint8 PortIdx,`
	`  const EthSwt_PortMirrorStateType* PortMirrorStatePtr`
	`)`	

Service ID[hex]:	0x39	
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
	PortMirrorStatePtr	Pointer to the memory where the requested port state is stored. 0x00 == port mirroring disabled, 0x01 == port mirroring enabled
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: the port mirroring state for the indexed Ethernet switch port was set successfully. E_NOT_OK: the port mirror configuration for the indexed Ethernet switch was not set successfully. (i.e. indexed Ethernet switch port is not available)
Description:	Set the current state of port mirroring for the indexed Ethernet switch port	

] ()

[SWS_EthSwt_00323][The function EthSwt_SetPortMirrorState shall return if the setting of the port mirroring state of the indexed Ethernet switch port was sucessfully.] (SRS_Eth_00123)
[SWS_EthSwt_00324][If development error detection is enabled: the function EthSwt_SetPortMirrorState shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)
[SWS_EthSwt_00325][If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_SetPortMirrorState shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)
[SWS_EthSwt_00326][If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_SetPortMirrorState shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)
[SWS_EthSwt_00327][The function EthSwt_SetPortMirrorState shall be pre compile time configurable On/Off by the configuration parameter: EthSwt_SetPortMirrorStateApi.] (SRS_BSW_00171)

8.3.46 EthSwt_SetPortTestMode

[SWS_EthSwt_91022] [

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. The test mode shall be forwarded to the EthTrcv (EthTrcv_SetPhyTestMode) that is referenced by the EthSwtPort. If no tranceiver is referenced the return value shall be E_NOT_OK.	

] ()
[SWS_EthSwt_00328][The function EthSwt_SetPortTestMode shall activate the indexed Ethernet transceiver that is connected to the indexed port to the given test mode. The indexed port is connected to the indexed EthSwt. The value is provided

by calling the function EthTrcv_SetPhyTestMode of the Ethernet Transceiver Driver. ⌋ (SRS_Eth_00123)

[SWS_EthSwt_00329]⌈ If development error detection is enabled: the function EthSwt_SetPortTestMode shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.⌋ (SRS_BSW_00406)

[SWS_EthSwt_00330]⌈ If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_SetPortTestMode shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00331]⌈ If development error detection is enabled and the parameter SwitchPortIdx is not valid,EthSwt_SetPortTestMode shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.⌋ (SRS_BSW_323, SRS_BSW_369)

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

[SWS_EthSwt_00333]⌈ The function EthSwt_SetPortTestMode shall check if the corresponding EthTrcv API EthTrcv_GetPhyIdentifier() of the indexed transceiver driver for the given SwitchPortIdx is available. If this is not the case, the function shall return E_NOT_OK and if development error tracing is activated by EthSwtDevErrorDetect the ETHSWT_E_INV_API shall be raised.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369, SRS_ETH_00118)

8.3.47 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. The loop-back mode shall be forwarded to the EthTrcv (EthTrcv_SetPhyLoopbackMode) that is referenced by the EthSwtPort. If no	

	transceiver is referenced the return value shall be E_NOT_OK.

] ()

[SWS_EthSwt_00334][The function EthSwt_SetPortLoopbackMode shall activate the indexed Ethernet transceiver that is connected to the indexed port to the given test mode. The indexed port is connected to the indexed EthSwt. The value is provided by calling the function EthTrcv_SetPhyTestMode of the Ethernet Transceiver Driver.] (SRS_Eth_00123)

[SWS_EthSwt_00335][If development error detection is enabled: the function EthSwt_SetPortLoopbackMode shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00336][If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_SetPortLoopbackMode shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00337][If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_SetPortLoopbackMode shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)

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

[SWS_EthSwt_00339][The function EthSwt_SetPortLoopbackMode shall check if the corresponding EthTrcv API EthTrcv_SetPhyLoopbackMode() of the indexed transceiver driver for the given SwitchPortIdx is available. If this is not the case, the function shall return E_NOT_OK and if development error tracing is activated by EthSwtDevErrorDetect the ETHSWT_E_INV_API shall be raised.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369, SRS_ETH_00118)

8.3.48 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. The transmission mode shall be forwarded to the EthTrcv (EthTrcv_SetPhyTxMode) that is referenced by the EthSwtPort. If no transceiver is referenced the return value shall be E_NOT_OK.	

⌋ ()

[SWS_EthSwt_00340]⌈ The function EthSwt_SetPortTxMode shall activate the indexed Ethernet transceiver that is connected to the indexed port to the given test mode. The indexed port is connected to the indexed EthSwt. The value is provided by calling the function EthTrcv_SetPhyTxMode of the Ethernet Transceiver Driver.⌋ (SRS_Eth_00123)

[SWS_EthSwt_00341]⌈ If development error detection is enabled: the function EthSwt_SetPortTxMode shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.⌋ (SRS_BSW_00406)

[SWS_EthSwt_00342]⌈ If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_SetPortTxMode shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00343]⌈ If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_SetPortTxMode shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.⌋ (SRS_BSW_323, SRS_BSW_369)

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

[SWS_EthSwt_00345]⌈ The function EthSwt_SetPortTxMode shall check if the corresponding EthTrcv API EthTrcv_SetPhyTxMode() of the indexed transceiver driver for the given SwitchPortIdx is available. If this is not the case, the function shall return E_NOT_OK and if development error tracing is activated by EthSwtDevErrorDetect the ETHSWT_E_INV_API shall be raised.⌋ (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369, SRS_ETH_00118)

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. The transmission mode shall be forwarded to the EthTrcv (EthTrcv_GetCableDiagnosticsResult) that is referenced by the EthSwtPort. If no Ethernet transceiver is referenced by the Ethernet switch port the cable diagnostic shall be set to ETHTRCV_CABLEDIAG_OK.	

] ()

[SWS_EthSwt_00346][The function EthSwt_GetPortCableDiagnosticsResult shall retrieve the cable diagnostic result of the indexed Ethernet transceiver that is connected to the indexed port. The indexed port is connected to the indexed EthSwt. The value is provided by calling the function EthTrcv_GetCableDiagnosticsResult of the Ethernet Transceiver Driver.] (SRS_Eth_00123)

[SWS_EthSwt_00347][If development error detection is enabled: the function EthSwt_GetPortCableDiagnosticsResult shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00348][If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_GetPortCableDiagnosticsResult shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00349][If development error detection is enabled and the parameter SwitchPortIdx is not valid, EthSwt_GetPortCableDiagnosticsResult shall raise the development error ETHSWT_E_INV_SWITCHPORT_IDX and return E_NOT_OK.] (SRS_BSW_323, SRS_BSW_369)

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

[SWS_EthSwt_00351][The function EthSwt_GetPortCableDiagnosticsResult shall check if the corresponding EthTrcv API EthTrcv_SetPhyTxMode() of the indexed transceiver driver for the given SwitchPortIdx is available. If this is not the case, the function shall return E_NOT_OK and if development error tracing is activated by EthSwtDevErrorDetect the ETHSWT_E_INV_API shall be raised.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369, SRS_ETH_00118)

8.3.50 EthSwt_GetCfgHexDump

[SWS_EthSwt_91026] [

Service name:	EthSwt_GetCfgHexDump

Syntax:	`Std_ReturnType EthSwt_GetCfgHexDump(`
	`  uint8 SwitchIdx,`
	`  uint32* CfgBlockNumber,`

	uint32* CfgBlockLengthPtr, uint8* ResultHexDumpBlockPtr)	
Service ID[hex]:	0x3d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	CfgBlockNumber	Number of the memory block (0...(2^32)-1). The number is a zero based consecutive range with max (2**32)-1. The CfgBlockNumber is the quotient of the total hex dump length (EthSwt_GetCfgHexDumpLength())devided by the length of one block (*(CfgBlockLengthPtr))
Parameters (inout):	CfgBlockLengthPtr	Length of the memory block in bytes that shall be copied (in). Return the number of bytes that where copied (out).
Parameters (out):	ResultHexDumpBlockPtr	Pointer to the location where where the memory block shall be copied. The first 4 bytes represent the absolute start address of the memory block.
Return value:	Std_ReturnType	E_OK: the switch hex dump of the configuration was obtained successfully. E_NOT_OK: the switch hex dump of the configuration was not obtained successfully. (i.e. indexed Ethernet switch is not available)
Description:	Retrieves the switch configuration of the indexed Ethernet switch for a certain block with a certain length.	

] ()

[SWS_EthSwt_00352][The function EthSwt_GetCfgHexDump shall return a memory block of the Ethernet switch configuration. The first 4 byte (byte0 ... byte3) shall contain the absolute start address in big endian format followed by the memory block content (Ethernet switch register). Unused memory bytes shall be padded with 0x00. The length of the block is given as parameter "CfgBlockLengthPtr". The memory block shall be copied to the given location "resultHexDumpBlockPtr". The length of the copied memory block (including the 4 byte address) shall be rewritten to "CfgBlockLengthPtr"] (SRS_Eth_00123)

[SWS_EthSwt_00353][If development error detection is enabled: the function EthSwt_GetCfgHexDump shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00354][If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_GetCfgHexDump shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00355][The function EthSwt_GetCfgHexDump shall be pre compile time configurable On/Off by the configuration parameter: EthSwt_GetCfgHexDumpApi.] (SRS_BSW_00171)

8.3.51 EthSwt_GetCfgHexDumpLength

[SWS_EthSwt_91027] [

Service name:	EthSwt_GetCfgHexDumpLength
Syntax:	Std_ReturnType EthSwt_GetCfgHexDumpLength(uint8 SwitchIdx,

	uint32* ResultTotalLengthPtr)	
Service ID[hex]:	0x3e	
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):	ResultTotalLengthPtr	Total Length of the memory where the configuration of all registers are stored
Return value:	Std_ReturnType	E_OK: the switch hex dump length was obtained successfully. E_NOT_OK: the switch hex dump length was not obtained successfully. (i.e. indexed Ethernet switch is not available)
Description:	Retrieves the total length of the Ethernet switch configuration of the indexed Ethernet switch	

] ()

[SWS_EthSwt_00356][The function EthSwt_GetCfgHexDumpLength shall return the total amount of the memory size of the Ethernet switch configuration.] (SRS_Eth_00123)

[SWS_EthSwt_00357][If development error detection is enabled: the function EthSwt_GetCfgHexDumpLength shall check that the service EthSwt_SwitchInit was previously called. If the check fails, the function shall raise the development error ETHSWT_E_NOT_INITIALIZED and return E_NOT_OK.] (SRS_BSW_00406)

[SWS_EthSwt_00358][If development error detection is enabled and the parameter SwitchIdx is not valid, EthSwt_GetCfgHexDumpLength shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_BSW_00413, SRS_BSW_323, SRS_BSW_369)

[SWS_EthSwt_00359][The function EthSwt_GetCfgHexDumpLength shall be pre compile time configurable On/Off by the configuration parameter: EthSwt_GetCfgHexDumpLengthApi.] (SRS_BSW_00171)

8.4 Call-back notifications

8.4.1 <user>_LinkDown

[SWS_EthSwt_00117] [

Service name:	<User>_LinkDown	
Syntax:	void <User>_LinkDown(uint8* SwitchIdxPtr, uint8* PortIdxPtr)	
Service ID[hex]:	0x19	
Sync/Async:	Synchronous /Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	SwitchIdxPtr	Pointer to the switch index
	PortIdxPtr	Pointer to the port index

Return value:	None
Description:	Shall be called, if a link which is configured for .1X goes down (link loss)

] (SRS_ETH_00086, SRS_ETH_00087)

[SWS_EthSwt_00118] [
The function <User>_LinkDown shall be called if a link which is configured for .1X goes down (link loss). The function returns the Switch index and the Port index, such that the port which went down can be identified.] (SRS_ETH_00119, SRS_ETH_00087)

[SWS_EthSwt_00119] [
The function <User>_LinkDown shall be pre compile time configurable by the <user> with content of EthSwtLinkDownUser.] (SRS_BSW_00171)

8.4.2 <user>_LinkUp

[SWS_EthSwt_00203] [

Service name:	<User>_LinkUp	

Syntax:	`void <User>_LinkUp(`
	`  uint8* SwitchIdxPtr,`
	`  uint8* PortIdxPtr`
	`)`	

Service ID[hex]:	0x1a	
Sync/Async:	Synchronous /Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	SwitchIdxPtr	Pointer to the switch index
	PortIdxPtr	Pointer to the port index
Return value:	None	
Description:	Shall be called, if a link up occurred. In case the hardware is not able to signal a link up via interrupt, this function needs to poll the link status in its main function.	

] (SRS_ETH_00086, SRS_ETH_00087)

[SWS_EthSwt_00204] [
The function <User>_LinkUp shall be called if a link comes up. The function returns the Switch index and the Port index, such that the port which went down 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.

[SWS_EthSwt_00205] [
The function <User>_LinkUp shall be pre compile time configurable by the <user> with content of EthSwtLinkUpUser.] (SRS_BSW_00171)

8.4.3 <user>_PersistentConfigurationResult

[SWS_EthSwt_00193] ⌈

Service name:	<User>_PersistentConfigurationResult
Syntax:	`void <User>_PersistentConfigurationResult( NvM_RequestResultType JobResult )`
Service ID[hex]:	0x1b
Sync/Async:	Synchronous /Asynchronous
Reentrancy:	Non 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:	Shall be called by the EthSwt_NvmSingleBlockCallback

⌋ (SRS_ETH_00086, SRS_ETH_00087)

[SWS_EthSwt_00194] ⌈
The function <User>_PersistentConfigurationResult shall be called by the NvmSingleBlockCallback to inform the caller of EthSwt_StoreConfiguration or EthSwt_ResetConfiguration about the state of the past calls.⌋ (SRS_ETH_00122, SRS_ETH_00087)

[SWS_EthSwt_00195] ⌈
The function <User>_PersistentConfigurationResult shall be pre compile time configurable by the <user> with content of EthSwtPersistentConfigurationResult. ⌋ (SRS_BSW_00171)

8.5 Scheduled functions

[SWS_EthSwt_00114] ⌈

Service name:	EthSwt_MainFunction
Syntax:	`void EthSwt_MainFunction( void )`
Service ID[hex]:	0x1c
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Service to support asynchronous behavior of API calls

⌋ (SRS_ETH_00086)

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

[SWS_EthSwt_00122] ⌈
The EthSwt_MainFunction shall call the API EthSwt_GetCounterValues and shall check each single value of referenced by CounterPtr:

1. If the first value is greater than zero, the function shall raise the development error ETHSWT_E_BUFFEROVERRUN
2. If the second value is greater than zero, the function shall raise the development error ETHSWT_E_CRC
3. If the third value is greater than zero, the function shall raise the development error ETHSWT_E_UNDERSIZEPCKT
4. If the forth value is greater than zero, the function shall raise the development error ETHSWT_E_OVERSIZEPCKT
5. If the fifth value is greater than zero, the function shall raise the development error ETHSWT_E_ALIGNMENT
6. If the sixth value is greater than zero, the function shall raise the development error ETHSWT_E_SQETEST
7. If the seventh value is greater than zero, the function shall raise the development error ETHSWT_E_INDISCARD
8. If the eighth value is greater than zero, the function shall raise the development error ETHSWT_E_INERROR
9. If the ninth value is greater than zero, the function shall raise the development error ETHSWT_E_OUTDISCARD
10. If the tenth value is greater than zero, the function shall raise the development error ETHSWT_E_OUTERROR
11. If the 11th value is greater than zero, the function shall raise the development error ETHSWT_E_SINGLECOLLISION
12. If the 12th value is greater than zero, the function shall raise the development error ETHSWT_E_MULTIPLECOLLISION
13. If the 13th value is greater than zero, the function shall raise the development error ETHSWT_E_DEFFEREDTRANSMISSION
14. If the 14th value is greater than zero, the function shall raise the development error ETHSWT_E_LATECOLLISION
15. If the eleventh value is greater than zero, the function shall raise the development error ETHSWT_E_DROPCOUNTER⌋ ()

[SWS_EthSwt_00116] ⌈
If development error detection for the module EthSwt is enabled the function EthSwt_MainFunction shall raise the development error ETHSWT_E_NOT_INITIALIZED in case it was called before the EthSwt has been initialized. ⌋ (SRS_BSW_00406)

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	*Description*
Dem_SetEventStatus	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.
Det_ReportError	Service to report development errors.
Eth_ReadMii	Reads a transceiver register
Eth_WriteMii	Configures a transceiver register or triggers a function offered by the receiver
EthIf_SwitchEgressTimeStampIndication	Returns an egress timestamp value out of the Switch. If the HW resolution is lower than the Eth_TimeStampType resolution resp. range, than the remaining bits will be filled with 0.
EthIf_SwitchIngressTimeStampIndication	Returns an ingress timestamp value out of the Switch. If the HW resolution is lower than the Eth_TimeStampType resolution resp. range, than the remaining bits will be filled with 0.
EthIf_SwitchMgmtInfoIndication	Ingress Switch management info indication redirected call to upper layers who registered for the call.
EthTrcv_GetBaudRate	Obtains the baud rate of the indexed transceiver
EthTrcv_GetDuplexMode	Obtains the duplex mode of the indexed transceiver
EthTrcv_GetLinkState	Obtains the link state of the indexed transceiver
EthTrcv_GetTransceiverMode	Obtains the state of the indexed transceiver
EthTrcv_SetTransceiverMode	Enables / disables the indexed transceiver
EthTrcv_StartAutoNegotiation	Restarts the negotiation of the transmission parameters used by the indexed transceiver
NvM_GetErrorStatus	Service to read the block dependent error/status information.
NvM_ReadBlock	Service to copy the data of the NV block to its corresponding RAM block.
NvM_WriteBlock	Service to copy the data of the RAM block to its corresponding NV block.
Spi_AsyncTransmit	Service to transmit data on the SPI bus.
Spi_Cancel	Service cancels the specified on-going sequence transmission.
Spi_ReadIB	Service for reading synchronously one or more data from an IB SPI Handler/Driver Channel specified by parameter.
Spi_SetAsyncMode	Service to set the asynchronous mechanism mode for SPI busses handled asynchronously.
Spi_SetupEB	Service to setup the buffers and the length of data for the EB SPI Handler/Driver Channel specified.
Spi_SyncTransmit	Service to transmit data on the SPI bus
Spi_WriteIB	Service for writing one or more data to an IB SPI

	Handler/Driver Channel specified by parameter.

] ()

[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

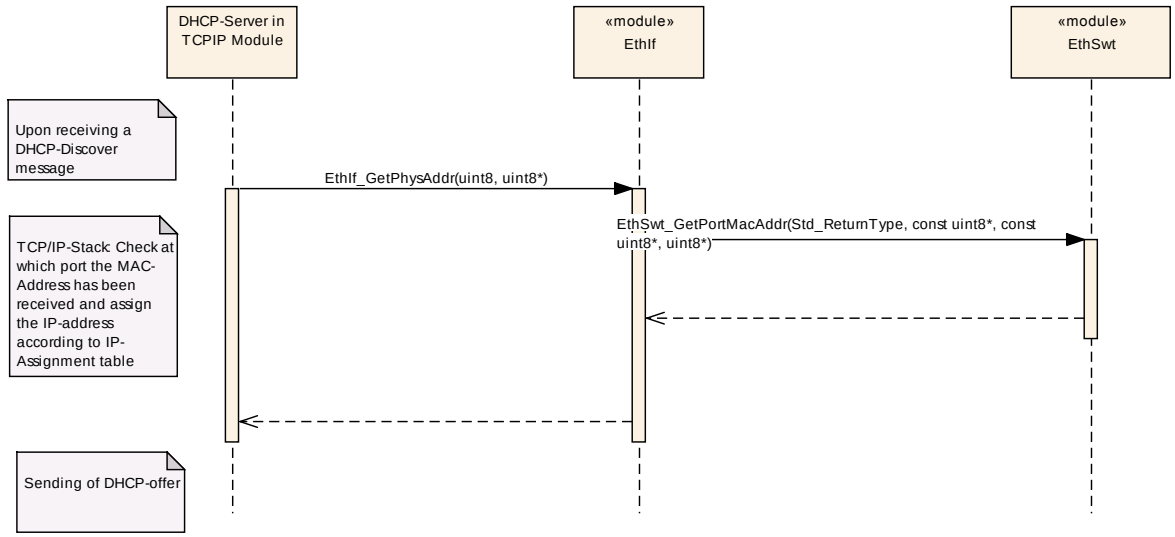
No configurable interfaces defined.

8.7 Service Interfaces

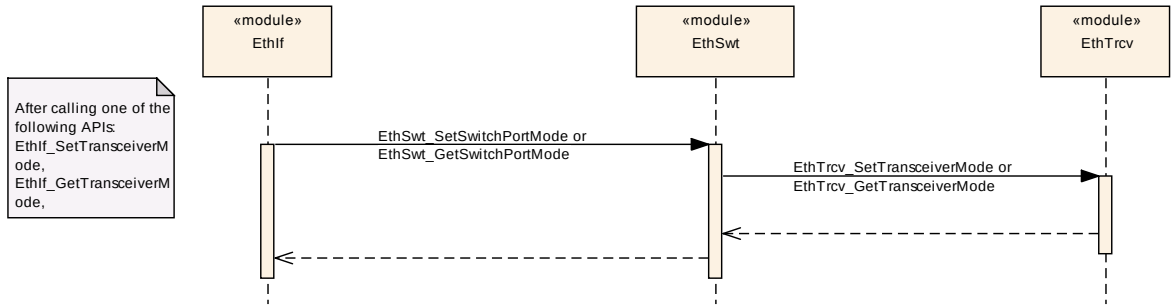
No direct access is necessary from the application layer.

9 Sequence diagrams

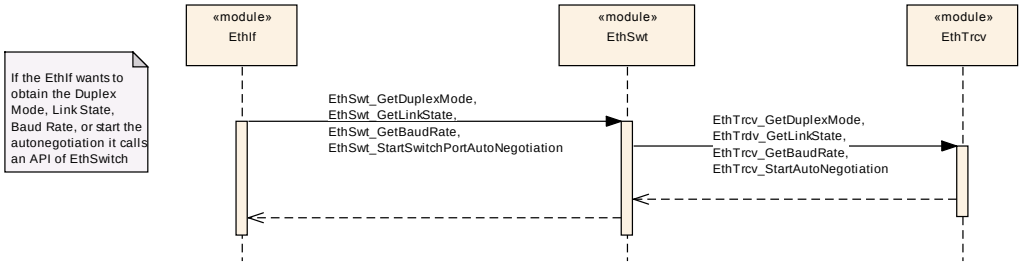
The following sequence diagram shows the interaction between the DHCP-Server in the TCP/IP-module and the Ethernet Switch Driver:



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



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



9.1 Switch Management support

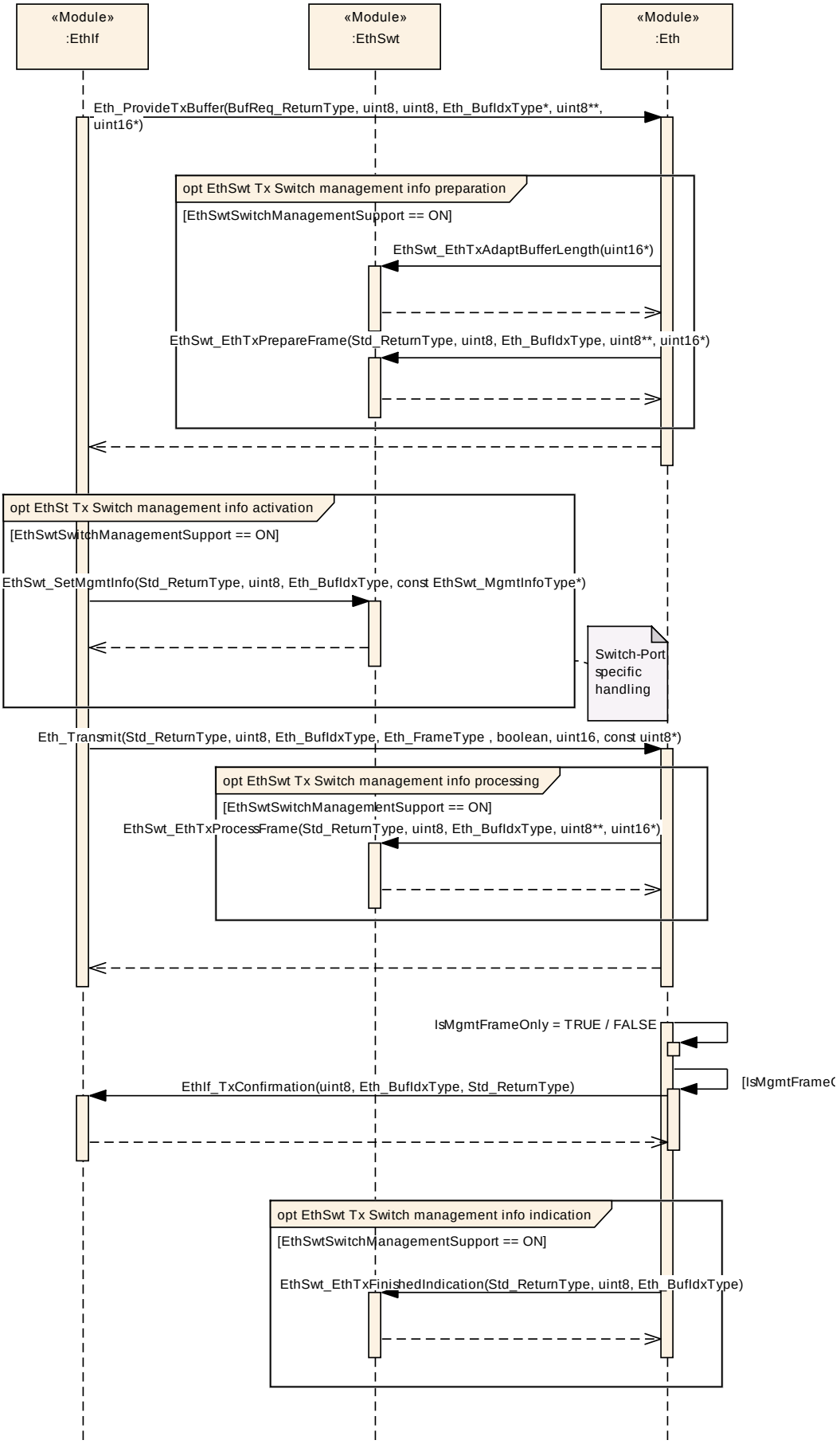

«Module»
:EthIf
«Module»
:EthSwt
«Module»
:Eth
Eth_ProvideTxBuffer(BufReq_ReturnType, uint8, uint8, Eth_BufIdxType*, uint8**, uint16*)
opt EthSwt Tx Switch management info preparation
[EthSwtSwitchManagementSupport == ON]
EthSwt_EthTxAdaptBufferLength(uint16*)
EthSwt_EthTxPrepareFrame(Std_ReturnType, uint8, Eth_BufIdxType, uint8**, uint16*)
opt EthSt Tx Switch management info activation
[EthSwtSwitchManagementSupport == ON]
EthSwt_SetMgmtInfo(Std_ReturnType, uint8, Eth_BufIdxType, const EthSwt_MgmtInfoType*)
Switch-Port specific handling
Eth_Transmit(Std_ReturnType, uint8, Eth_BufIdxType, Eth_FrameType , boolean, uint16, const uint8*)
opt EthSwt Tx Switch management info processing
[EthSwtSwitchManagementSupport == ON]
EthSwt_EthTxProcessFrame(Std_ReturnType, uint8, Eth_BufIdxType, uint8**, uint16*)
IsMgmtFrameOnly = TRUE / FALSE
[IsMgmtFrame(
EthIf_TxConfirmation(uint8, Eth_BufIdxType, Std_ReturnType)
opt EthSwt Tx Switch management info indication
[EthSwtSwitchManagementSupport == ON]
EthSwt_EthTxFinishedIndication(Std_ReturnType, uint8, Eth_BufIdxType)

Figure 9 Switch Management support for transmission

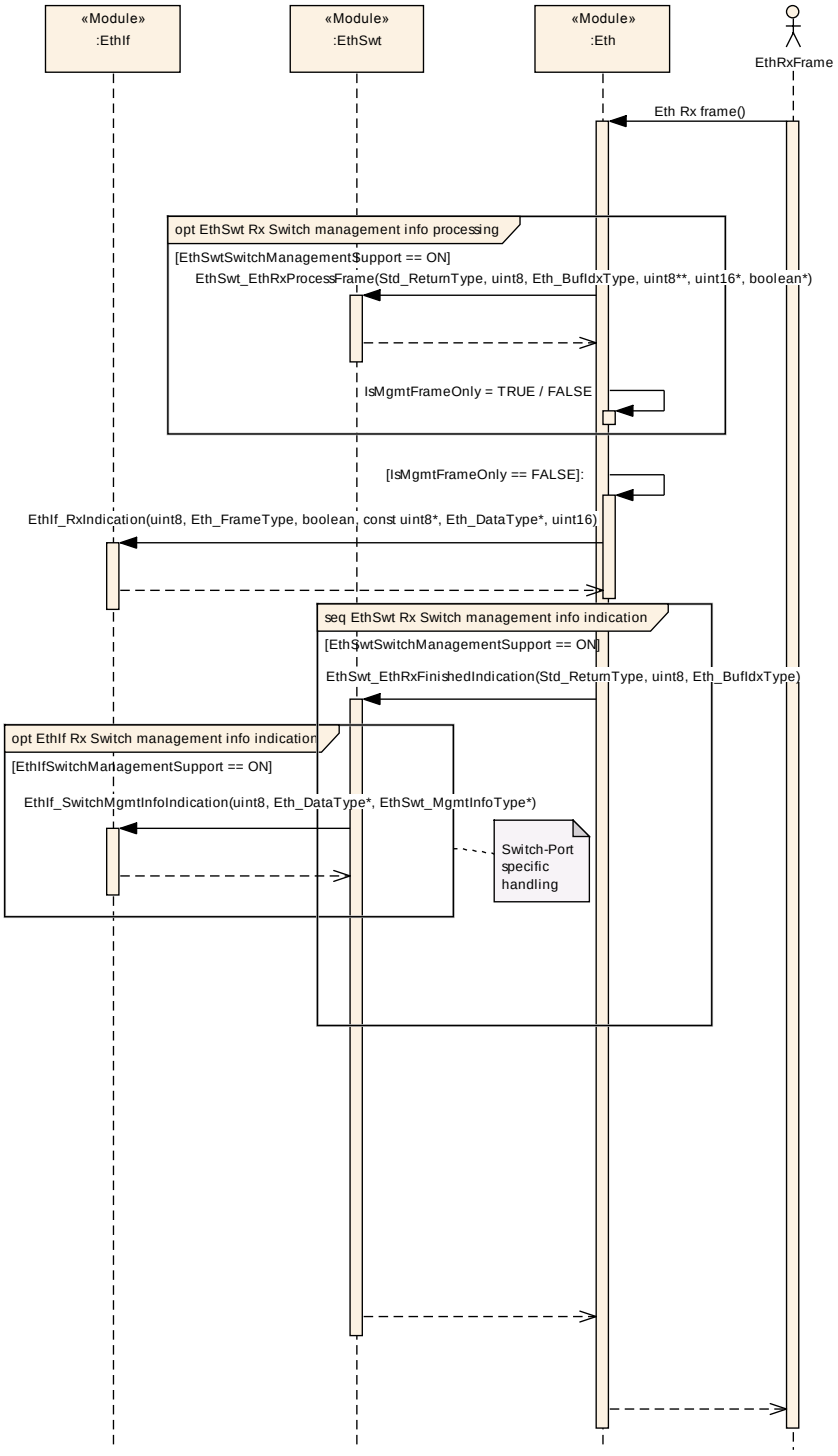


Figure 10 Management support for reception

10 Configuration specification

Chapter 10.1 specifies the structure (containers) and the parameters of the module EthSwt.

10.1 Containers and configuration parameters

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

10.1.1 EthSwt

SWS Item	**ECUC_EthSwt_00046 :**
Module Name	*EthSwt*
Module Description	Configuration of the EthSwt (Ethernet Switch Driver) module.
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		
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			

SWS Item	**ECUC_EthSwt_00073 :**
Name	EthSwtDropDoubleTagged
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		

SWS Item	**ECUC_EthSwt_00004 :**		
Name	EthSwtIdx		
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		

SWS Item	**ECUC_EthSwt_00110 :**		
Name	EthSwtManagementEthCtrlRef		
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		

SWS Item	**ECUC_EthSwt_00111 :**		
Name	EthSwtManagementPortRef		
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***

EthSwtDemEventParameterRefs	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		
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			

SWS Item	**ECUC_EthSwt_00006 :**		
Name	ETHSWT_E_ACCESS		
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		

No Included Containers

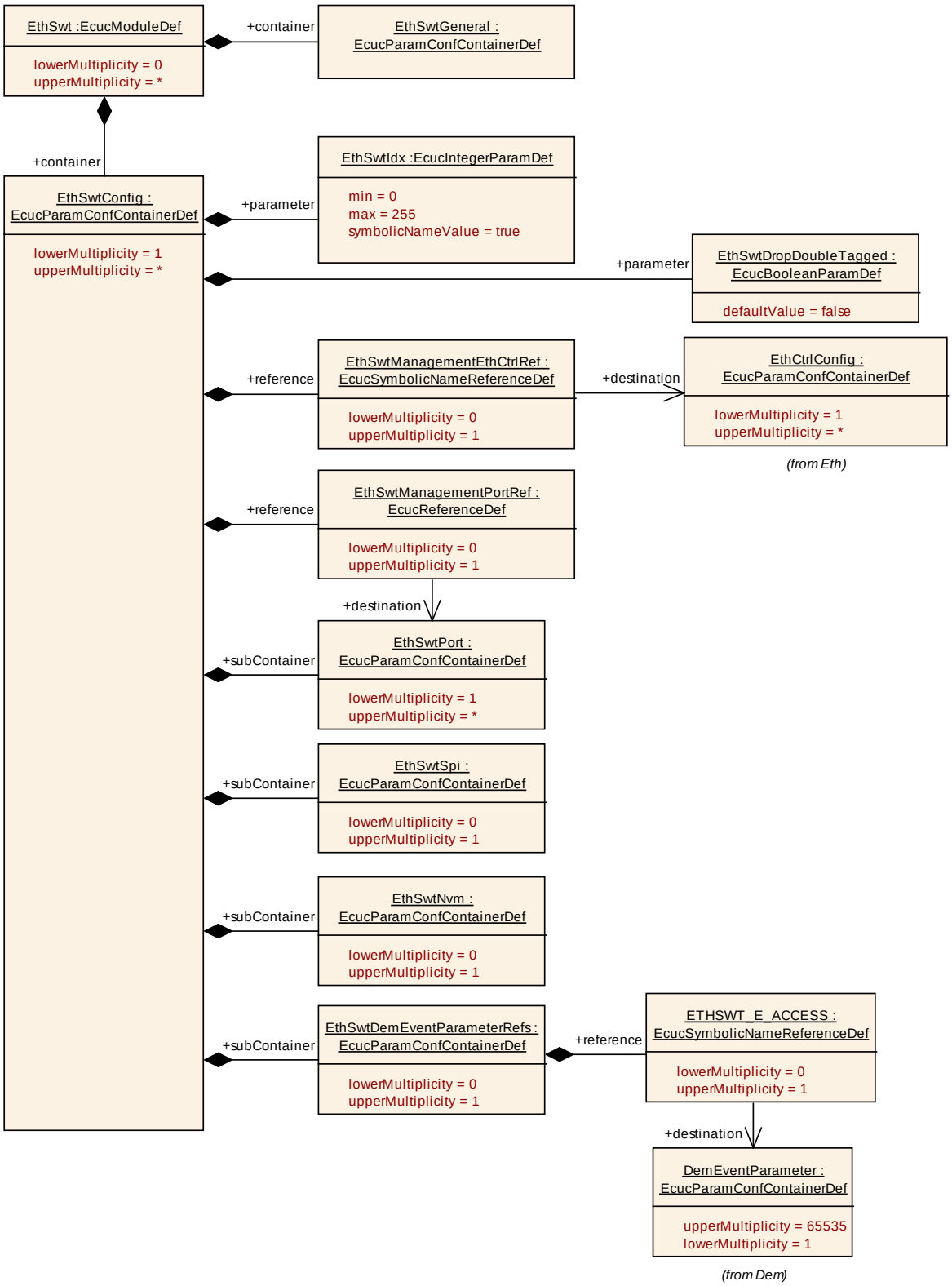


Figure 11: EthSwt

10.1.4 EthSwtGeneral

SWS Item	**ECUC_EthSwt_00003 :**		
Container Name	EthSwtGeneral		
Description	General configuration of Ethernet Switch Driver module.		
Configuration Parameters			

SWS Item	**ECUC_EthSwt_00002 :**		
Name	EthSwtDevErrorDetect		
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		

SWS Item	**ECUC_EthSwt_00055 :**		
Name	EthSwtEnableVlanApi		
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		

SWS Item	**ECUC_EthSwt_00052 :**		
Name	EthSwtGetArlTableApi		
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		

SWS Item	**ECUC_EthSwt_00040 :**		
Name	EthSwtGetBufferLevelApi		
Description	Enables / Disables API to fetch the switch buffer utilization.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants

	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthSwt_00093 :**		
Name	EthSwtGetCfgHexDumpApi		
Description	Enables / Disables EthSwt_GetCfgHexDump API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthSwt_00094 :**		
Name	EthSwtGetCfgHexDumpLengthApi		
Description	Enables / Disables EthSwt_GetCfgHexDumpLength API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthSwt_00053 :**		
Name	EthSwtGetDropCountApi		
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		

SWS Item	**ECUC_EthSwt_00061 :**		
Name	EthSwtGetMacLearningModeApi		
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		

SWS Item	**ECUC_EthSwt_00092 :**
Name	EthSwtGetPortCableDiagnosticsResultApi
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		

SWS Item	**ECUC_EthSwt_00083 :**		
Name	EthSwtGetPortIdentifierApi		
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		

SWS Item	**ECUC_EthSwt_00051 :**		
Name	EthSwtGetPortMacAddrApi		
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		

SWS Item	**ECUC_EthSwt_00087 :**		
Name	EthSwtGetPortMirrorStateApi		
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		

SWS Item	**ECUC_EthSwt_00082 :**		
Name	EthSwtGetPortSignalQualityApi		
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

SWS Item	**ECUC_EthSwt_00065 :**		
Name	EthSwtGetRxStatsApi		
Description	Enables / Disables EthSwt_GetEtherStats API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthSwt_00084 :**		
Name	EthSwtGetSwitchIdentifierApi		
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		

SWS Item	**ECUC_EthSwt_00066 :**		
Name	EthSwtGetSwitchRegApi		
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		

SWS Item	**ECUC_EthSwt_00100 :**		
Name	EthSwtGetTxErrorCounterValuesApi		
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		

SWS Item	**ECUC_EthSwt_00099 :**
Name	EthSwtGetTxStatsApi
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		

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

SWS Item	**ECUC_EthSwt_00033 :**		
Name	EthSwtIndex		
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		

SWS Item	**ECUC_EthSwt_00048 :**		
Name	EthSwtLinkDownUser		
Description	Defines the <User> function name for the <User>_LinkDown callback.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthSwt_00068 :**
Name	EthSwtLinkUpUser

Description	Defines the <User> function name for the <User>_LinkUp callback.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthSwt_00102 :**		
Name	EthSwtLowPowerModeSupport		
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		

SWS Item	**ECUC_EthSwt_00071 :**		
Name	EthSwtMainFunctionPeriod		
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			

SWS Item	**ECUC_EthSwt_00108 :**
Name	EthSwtManagementSupportApi
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		

SWS Item	**ECUC_EthSwt_00109 :**		
Name	EthSwtMgmtInfoIndicationTimeout		
Description	This parameter specifies the timeout while the Switch driver is waiting for management information out of the Switch for reception. The value 0 deactivates the timeout supervision. Unit: 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	scope: local		

SWS Item	**ECUC_EthSwt_00062 :**		
Name	EthSwtPersistentConfigurationResult		
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		

SWS Item	**ECUC_EthSwt_00063 :**		
Name	EthSwtPersistentConfigurationResultUser		
Description	Defines the <User> function name for the <User>_PersistentConfigurationResult callback.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthSwt_00064 :**		
Name	EthSwtPublicCddHeaderFile		
Description	Defines header files for callback functions which shall be included in case of CDDs.		
Multiplicity	0..*		
Type	EcucStringParamDef		
Default value	--		
maxLength	32		
minLength	1		
regularExpression	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthSwt_00086 :**		
Name	EthSwtReadPortMirrorConfigurationApi		
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		

SWS Item	**ECUC_EthSwt_00069 :**		
Name	EthSwtReadTrcvRegisterApi		
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		

SWS Item	**ECUC_EthSwt_00049 :**		
Name	EthSwtResetConfigurationApi		
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

SWS Item	**ECUC_EthSwt_00104 :**		
Name	EthSwtSetForwardingModeApi		
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		

SWS Item	**ECUC_EthSwt_00060 :**		
Name	EthSwtSetMacLearningModeApi		
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		

SWS Item	**ECUC_EthSwt_00090 :**		
Name	EthSwtSetPortLoopbackModeApi		
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		

SWS Item	**ECUC_EthSwt_00088 :**		
Name	EthSwtSetPortMirrorStateApi		
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		

SWS Item	**ECUC_EthSwt_00089 :**		
Name	EthSwtSetPortTestModeApi		
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		

SWS Item	**ECUC_EthSwt_00091 :**		
Name	EthSwtSetPortTxModeApi		
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		

SWS Item	**ECUC_EthSwt_00067 :**		
Name	EthSwtSetSwitchRegApi		
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		

SWS Item	**ECUC_EthSwt_00050 :**		
Name	EthSwtStoreConfigurationApi		
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		

SWS Item	**ECUC_EthSwt_00105 :**		
Name	EthSwtVerifyConfigApi		
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		

SWS Item	ECUC_EthSwt_00031 :		
Name	EthSwtVersionInfoApi		
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		

SWS Item	ECUC_EthSwt_00085 :		
Name	EthSwtWritePortMirrorConfigurationApi		
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		

SWS Item	ECUC_EthSwt_00070 :		
Name	EthSwtWriteTrcvRegisterApi		
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		

No Included Containers

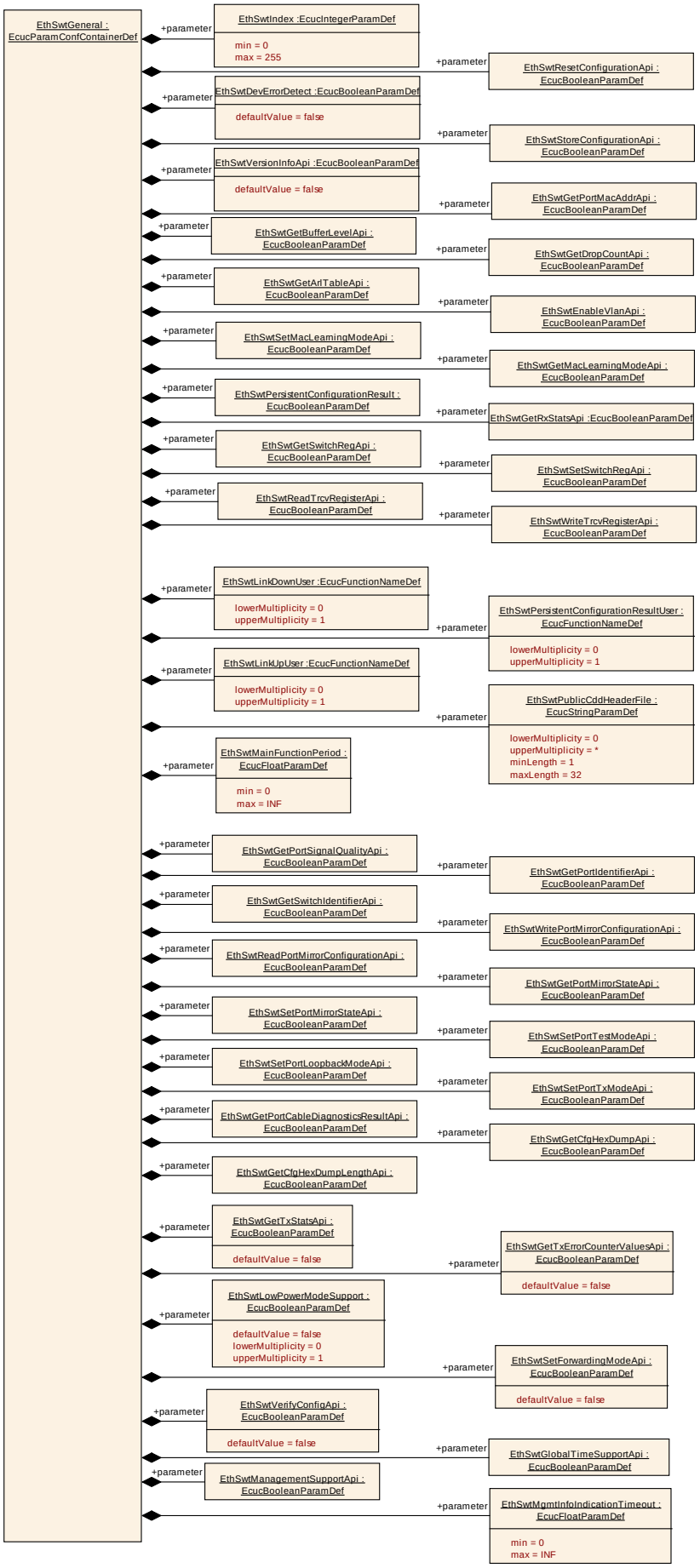

EthSwtGeneral : EcucParamConfContainerDef
+parameter
EthSwtIndex :EcucIntegerParamDef
min = 0
max = 255
EthSwtResetConfigurationApi : EcucBooleanParamDef
EthSwtDevErrorDetect :EcucBooleanParamDef
defaultValue = false
EthSwtStoreConfigurationApi : EcucBooleanParamDef
EthSwtVersionInfoApi :EcucBooleanParamDef
defaultValue = false
EthSwtGetPortMacAddrApi : EcucBooleanParamDef
EthSwtGetBufferLevelApi : EcucBooleanParamDef
EthSwtGetDropCountApi : EcucBooleanParamDef
EthSwtGetArlTableApi : EcucBooleanParamDef
EthSwtEnableVlanApi : EcucBooleanParamDef
EthSwtSetMacLearningModeApi : EcucBooleanParamDef
EthSwtGetMacLearningModeApi : EcucBooleanParamDef
EthSwtPersistentConfigurationResult : EcucBooleanParamDef
EthSwtGetRxStatsApi :EcucBooleanParamDef
EthSwtGetSwitchRegApi : EcucBooleanParamDef
EthSwtSetSwitchRegApi : EcucBooleanParamDef
EthSwtReadTrcvRegisterApi : EcucBooleanParamDef
EthSwtWriteTrcvRegisterApi : EcucBooleanParamDef
EthSwtLinkDownUser :EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
EthSwtPersistentConfigurationResultUser : EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
EthSwtLinkUpUser :EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
EthSwtPublicCddHeaderFile : EcucStringParamDef
lowerMultiplicity = 0
upperMultiplicity = *
minLength = 1
maxLength = 32
EthSwtMainFunctionPeriod : EcucFloatParamDef
min = 0
max = INF
EthSwtGetPortSignalQualityApi : EcucBooleanParamDef
EthSwtGetPortIdentifierApi : EcucBooleanParamDef
EthSwtGetSwitchIdentifierApi : EcucBooleanParamDef
EthSwtWritePortMirrorConfigurationApi : EcucBooleanParamDef
EthSwtReadPortMirrorConfigurationApi : EcucBooleanParamDef
EthSwtGetPortMirrorStateApi : EcucBooleanParamDef
EthSwtSetPortMirrorStateApi : EcucBooleanParamDef
EthSwtSetPortTestModeApi : EcucBooleanParamDef
EthSwtSetPortLoopbackModeApi : EcucBooleanParamDef
EthSwtSetPortTxModeApi : EcucBooleanParamDef
EthSwtGetPortCableDiagnosticsResultApi : EcucBooleanParamDef
EthSwtGetCfgHexDumpApi : EcucBooleanParamDef
EthSwtGetCfgHexDumpLengthApi : EcucBooleanParamDef
EthSwtGetTxStatsApi : EcucBooleanParamDef
defaultValue = false
EthSwtGetTxErrorCounterValuesApi : EcucBooleanParamDef
defaultValue = false
EthSwtLowPowerModeSupport : EcucBooleanParamDef
defaultValue = false
lowerMultiplicity = 0
upperMultiplicity = 1
EthSwtSetForwardingModeApi : EcucBooleanParamDef
defaultValue = false
EthSwtVerifyConfigApi : EcucBooleanParamDef
defaultValue = false
EthSwtGlobalTimeSupportApi : EcucBooleanParamDef
EthSwtManagementSupportApi : EcucBooleanParamDef
EthSwtMgmtInfoIndicationTimeout : EcucFloatParamDef
min = 0
max = INF

Figure 12: EthSwtGeneral

10.1.5 EthSwtPort

SWS Item	**ECUC_EthSwt_00005 :**		
Container Name	EthSwtPort		
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			

SWS Item	**ECUC_EthSwt_00047 :**		
Name	EthSwtPortEnableLinkDownCallback		
Description	Enables the callback <User>_LinkDown for this EthSwtPort if an IEEE802.1X link loss is detected. <User> is defined by EthSwtLinkDownUser.		
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, VARIANT-POST-BUILD
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthSwt_00013 :**		
Name	EthSwtPortIdx		
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		

SWS Item	**ECUC_EthSwt_00072 :**	
Name	EthSwtPortMacLayerType	
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		

SWS Item	**ECUC_EthSwt_00054 :**		
Name	EthSwtPortPhysicalLayerType		
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		

SWS Item	**ECUC_EthSwt_00032 :**
Name	EthSwtPortPredefinedMacAddresses
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	--		
maxLength	--		
minLength	--		
regularExpression	[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		

SWS Item	**ECUC_EthSwt_00101 :**		
Name	EthSwtPortRole		
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.		

SWS Item	**ECUC_EthSwt_00112 :**
Name	EthSwtPortTimeStampSupport
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		

SWS Item	**ECUC_EthSwt_00041 :**		
Name	EthSwtPortTrcvRef		
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
	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***
EthSwtPortEgress	1	Configuration of one Ethernet Switch Port Egress behavior.
EthSwtPortIngress	1	Configuration of one Ethernet Switch Port ingress behavior.
EthSwtPortVlanMembership	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.

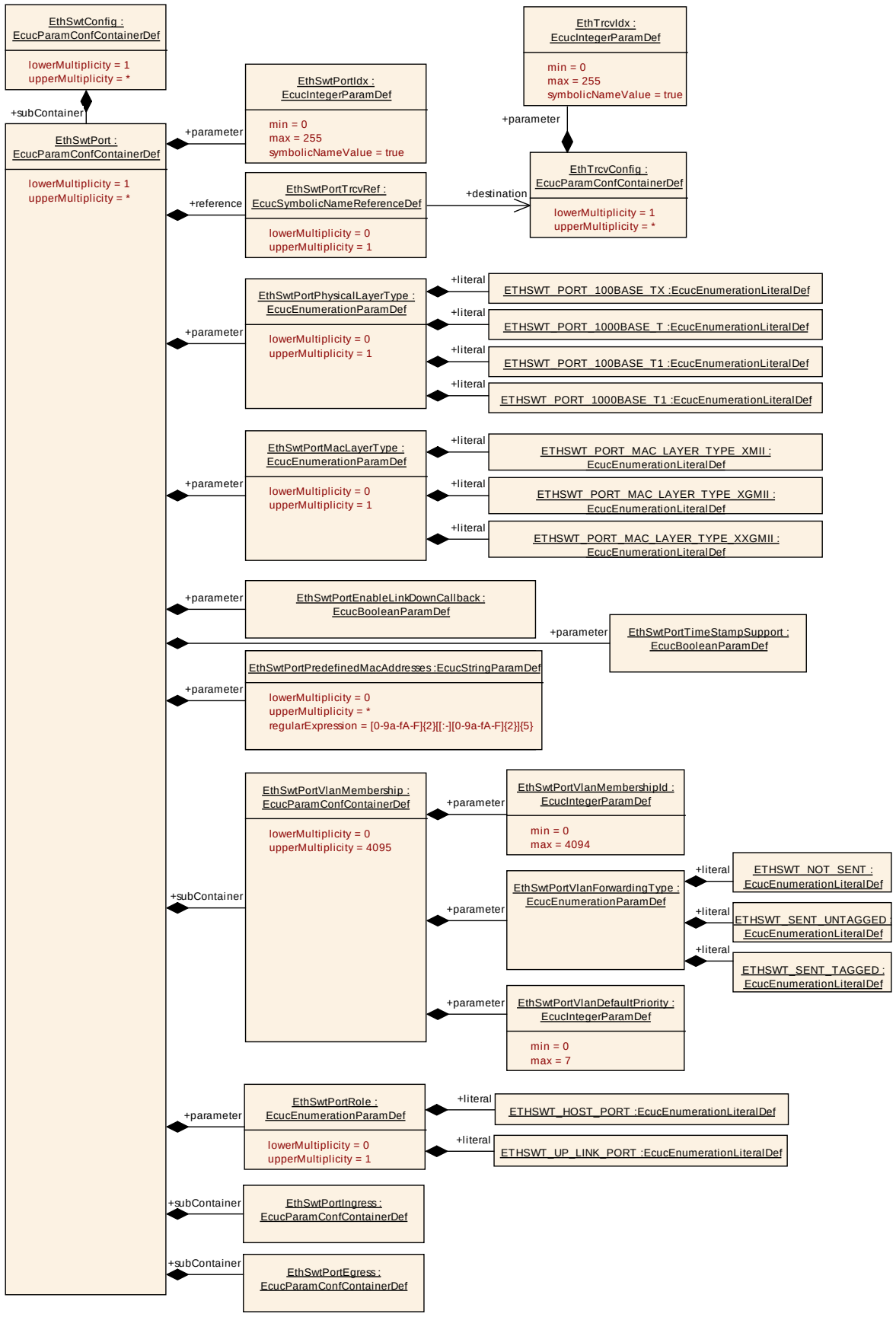


Figure 13: EthSwtPort

Please note that the functional behavior of the ingress and egress port of a switch is implemented in hardware in the switch devices (see [9]). 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
Description	Configuration of one Ethernet Switch Port ingress behavior.
Configuration Parameters	

SWS Item	**ECUC_EthSwt_00096 :**		
Name	EthSwtPortIngressDefaultPriority		
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.		

SWS Item	**ECUC_EthSwt_00095 :**		
Name	EthSwtPortIngressDefaultVlan		
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.

SWS Item	**ECUC_EthSwt_00097 :**		
Name	EthSwtPortIngressDropUntagged		
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.		

SWS Item	**ECUC_EthSwt_00015 :**		
Name	EthSwtPortIngressVlanModification		
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		

SWS Item	**ECUC_EthSwt_00023 :**		
Name	EthSwtPortTrafficClassAssignment		
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.
EthSwtPriorityRegeneration	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.
EthSwtPriorityTrafficClassAssignment	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).

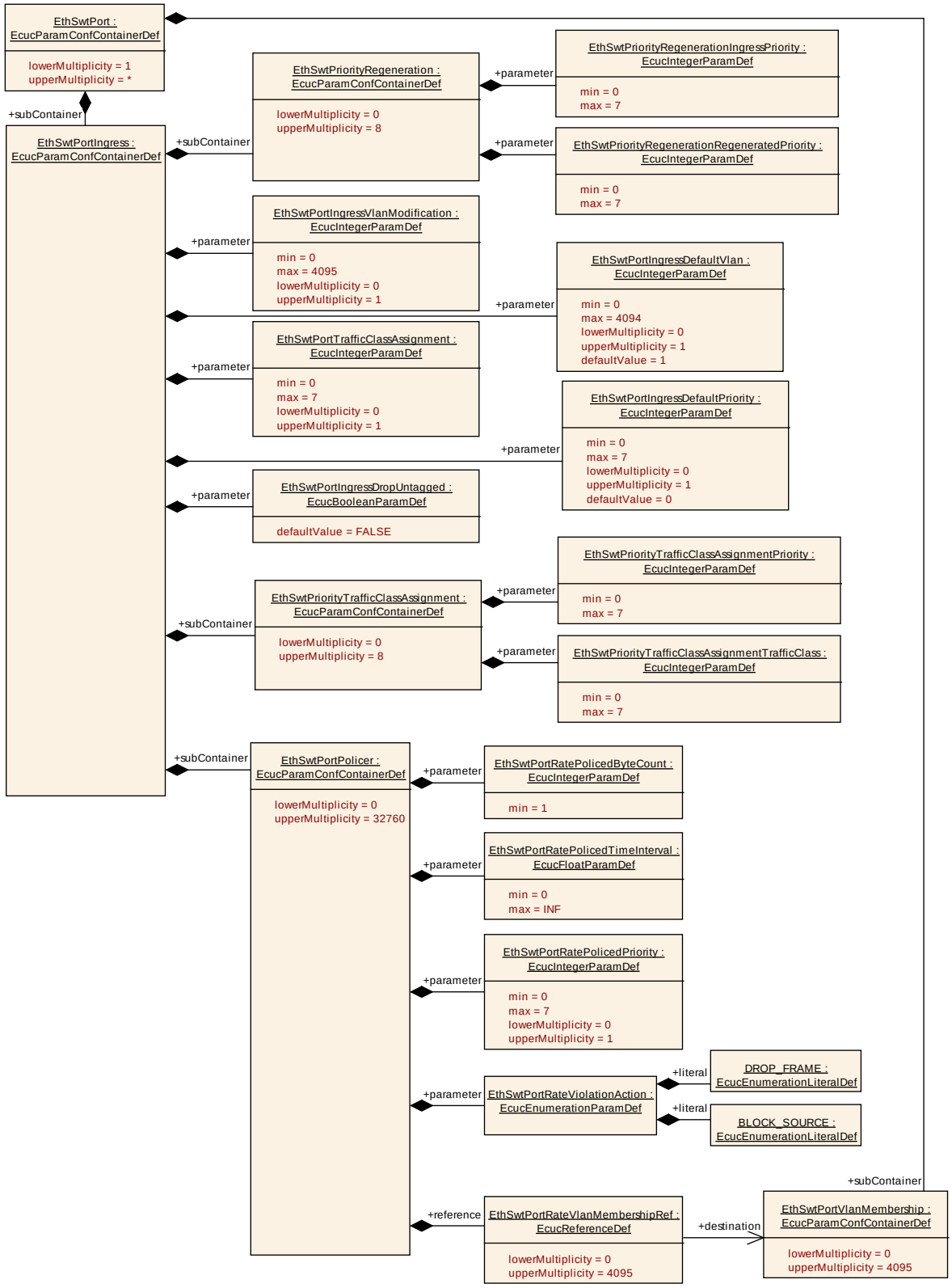


Figure 14: EthSwtPortIngress

10.1.7 EthSwtPortPolicer

SWS Item	**ECUC_EthSwt_00074 :**		
Container Name	EthSwtPortPolicer		
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			

SWS Item	**ECUC_EthSwt_00075 :**		
Name	EthSwtPortRatePolicedByteCount		
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		

SWS Item	**ECUC_EthSwt_00077 :**		
Name	EthSwtPortRatePolicedPriority		
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.		

SWS Item	**ECUC_EthSwt_00076 :**		
Name	EthSwtPortRatePolicedTimeInterval		
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		

SWS Item	**ECUC_EthSwt_00078 :**		
Name	EthSwtPortRateViolationAction		
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		

SWS Item	**ECUC_EthSwt_00081 :**		
Name	EthSwtPortRateVlanMembershipRef		
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 EthSwtPriorityRegeneration

SWS Item	**ECUC_EthSwt_00057 :**
Container Name	EthSwtPriorityRegeneration
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			

SWS Item	**ECUC_EthSwt_00058 :**		
Name	EthSwtPriorityRegenerationIngressPriority		
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		

SWS Item	**ECUC_EthSwt_00059 :**		
Name	EthSwtPriorityRegenerationRegeneratedPriority		
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

10.1.9 EthSwtPriorityTrafficClassAssignment

SWS Item	**ECUC_EthSwt_00027 :**		
Container Name	EthSwtPriorityTrafficClassAssignment		
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			

SWS Item	**ECUC_EthSwt_00028 :**		
Name	EthSwtPriorityTrafficClassAssignmentPriority		
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		

SWS Item	**ECUC_EthSwt_00029 :**		
Name	EthSwtPriorityTrafficClassAssignmentTrafficClass		
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

10.1.10 EthSwtPortEgress

SWS Item	**ECUC_EthSwt_00007 :**
Container Name	EthSwtPortEgress
Description	Configuration of one Ethernet Switch Port Egress behavior.
Configuration Parameters	

SWS Item	**ECUC_EthSwt_00008 :**		
Name	EthSwtPortEgressLastSchedulerRef		
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.

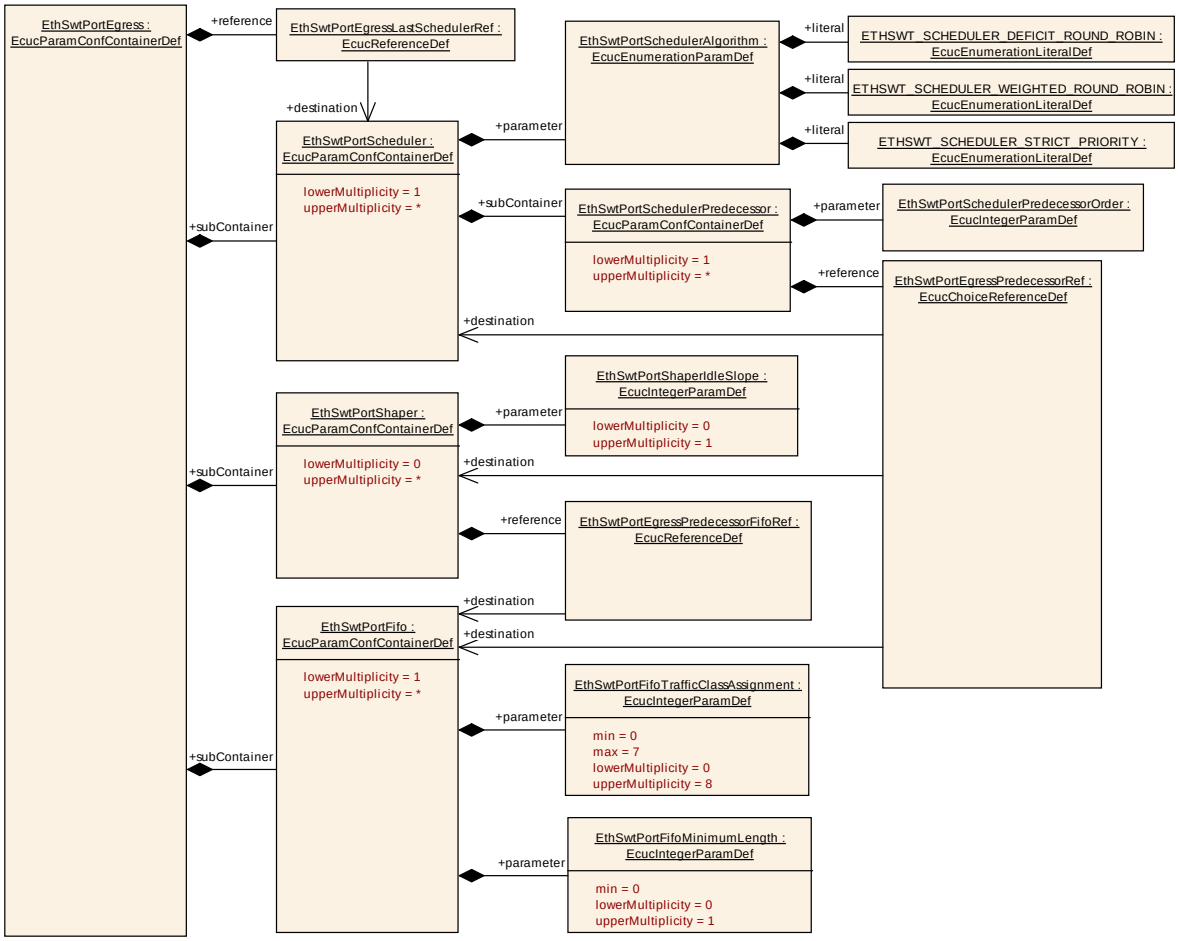


Figure 15: EthSwtPortEgress

10.1.11 EthSwtPortScheduler

SWS Item	ECUC_EthSwt_00017 :		
Container Name	EthSwtPortScheduler		
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			

SWS Item	ECUC_EthSwt_00018 :
Name	EthSwtPortSchedulerAlgorithm
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***
EthSwtPortSchedulerPredecessor	1..*	Defines an ordered list of predecessors for this scheduler.

10.1.12 EthSwtPortSchedulerPredecessor

SWS Item	**ECUC_EthSwt_00019 :**		
Container Name	EthSwtPortSchedulerPredecessor		
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			

SWS Item	**ECUC_EthSwt_00020 :**		
Name	EthSwtPortSchedulerPredecessorOrder		
Description	Defines the order of the scheduler predecessors.		
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		

SWS Item	**ECUC_EthSwt_00010 :**		
Name	EthSwtPortEgressPredecessorRef		
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

10.1.13 EthSwtPortShaper

SWS Item	**ECUC_EthSwt_00021 :**		
Container Name	EthSwtPortShaper		
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			

SWS Item	**ECUC_EthSwt_00042 :**	
Name	EthSwtPortShaperIdleSlope	
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	

SWS Item	**ECUC_EthSwt_00009 :**		
Name	EthSwtPortEgressPredecessorFifoRef		
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

10.1.14 EthSwtPortFifo

SWS Item	**ECUC_EthSwt_00011 :**		
Container Name	EthSwtPortFifo		
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			

SWS Item	**ECUC_EthSwt_00098 :**		
Name	EthSwtPortFifoMinimumLength		
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		

SWS Item	**ECUC_EthSwt_00012 :**		
Name	EthSwtPortFifoTrafficClassAssignment		
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

10.1.15 EthSwtPortVlanMembership

SWS Item	**ECUC_EthSwt_00079 :**
Container Name	EthSwtPortVlanMembership
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			

SWS Item	**ECUC_EthSwt_00056 :**		
Name	EthSwtPortVlanDefaultPriority		
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		

SWS Item	**ECUC_EthSwt_00026 :**		
Name	EthSwtPortVlanForwardingType		
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		

SWS Item	**ECUC_EthSwt_00080 :**		
Name	EthSwtPortVlanMembershipId		
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

10.1.16 EthSwtSpi

SWS Item	**ECUC_EthSwt_00030 :**		
Container Name	EthSwtSpi		
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.

10.1.17 EthSwtSpiSequence

SWS Item	**ECUC_EthSwt_00034 :**		
Container Name	EthSwtSpiSequence		
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			

SWS Item	**ECUC_EthSwt_00036 :**
Name	EthSwtSpiAccessSynchronous
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		

SWS Item	**ECUC_EthSwt_00035 :**		
Name	EthSwtSpiSequenceName		
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

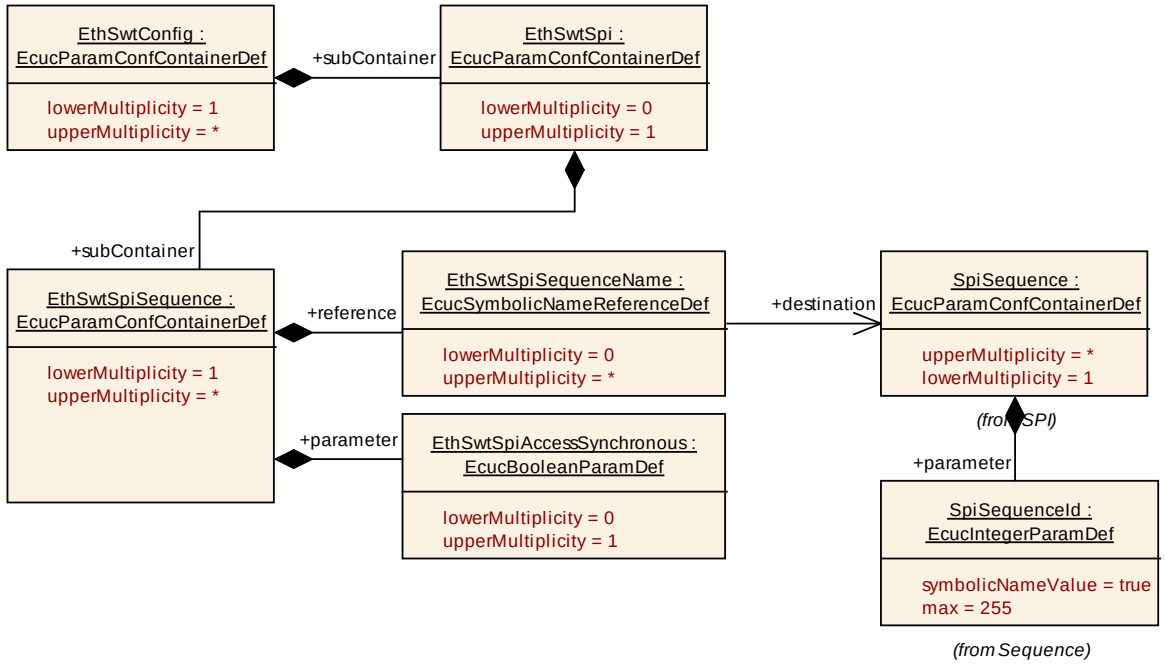


Figure 16: EthSwt Spi interaction

10.1.18 EthSwtNvm

SWS Item	**ECUC_EthSwt_00043 :**		
Container Name	EthSwtNvm		
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			

SWS Item	**ECUC_EthSwt_00044 :**		
Name	EthSwtNvmBlockDescriptorRef		
Description	Reference to the Nvm block description in the Nvm module configuration.		
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

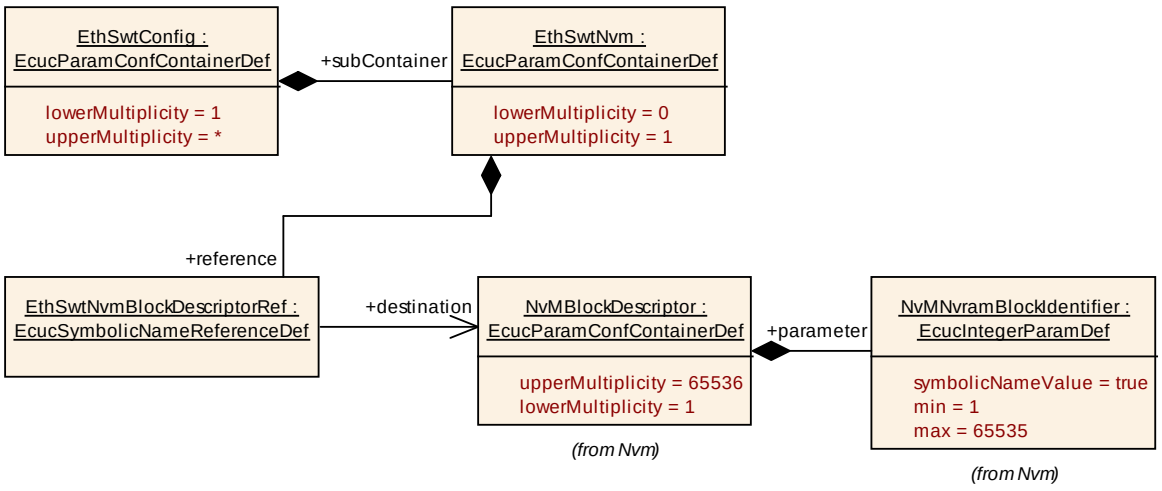


Figure 17: EthSwt NvM interaction