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 Release	4.2.2

Document Change History

Release	Changed by	Change Description
4.2.2	AUTOSAR Release Management	• minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
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 1-1 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.

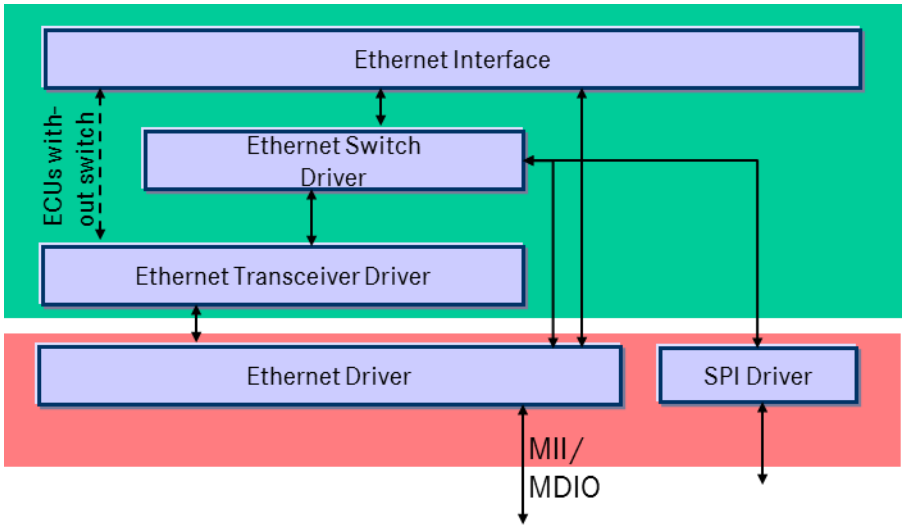


Figure 1-1 Ethernet Switch Driver in layer architecture

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

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

3.1 Related standards and norms

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

- Advanced Shaping and FIFO Functionality: Basically, the kind of shaper and the corresponding FIFO can be configured. Also the scheduling mechanism at the egress port of a switch can be configured. More advanced features and configuration parameters e.g. for AVB are not supported.
- 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

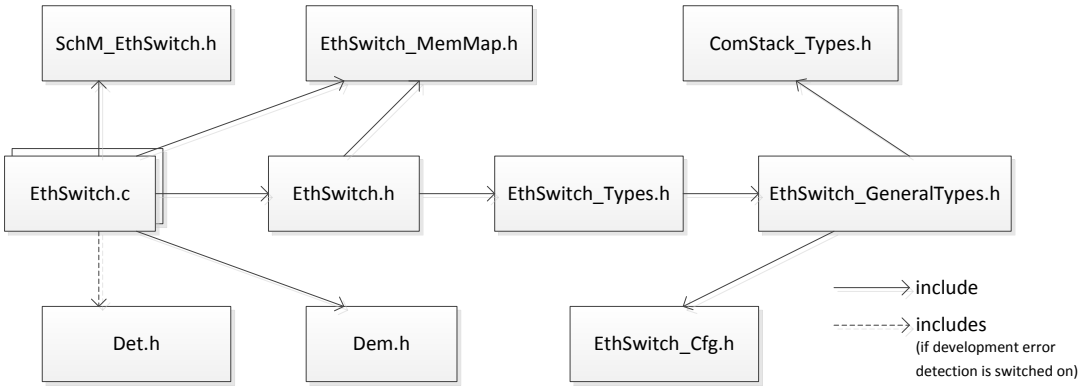


Figure 5-1 Ethernet Switch Driver file structure

6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_EthSwt_00160
SRS_BSW_00345	BSW Modules shall support pre-compile configuration	SWS_EthSwt_00159
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_EthSwt_00131
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_EthSwt_00161
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_EthSwt_00161
SRS_ETH_00086	-	SWS_EthSwt_00001, SWS_EthSwt_00002, SWS_EthSwt_00003, SWS_EthSwt_00004, SWS_EthSwt_00006, SWS_EthSwt_00007, SWS_EthSwt_00008, SWS_EthSwt_00009, SWS_EthSwt_00010, SWS_EthSwt_00011, SWS_EthSwt_00012, SWS_EthSwt_00013, SWS_EthSwt_00014, SWS_EthSwt_00016, SWS_EthSwt_00018, SWS_EthSwt_00019, SWS_EthSwt_00020, SWS_EthSwt_00021, SWS_EthSwt_00022, SWS_EthSwt_00023, SWS_EthSwt_00025, SWS_EthSwt_00026, SWS_EthSwt_00027, SWS_EthSwt_00028, SWS_EthSwt_00029, SWS_EthSwt_00031, SWS_EthSwt_00032, SWS_EthSwt_00033, SWS_EthSwt_00034, SWS_EthSwt_00035, SWS_EthSwt_00037, SWS_EthSwt_00038, SWS_EthSwt_00039, SWS_EthSwt_00040, SWS_EthSwt_00042, SWS_EthSwt_00044, SWS_EthSwt_00045, SWS_EthSwt_00046, SWS_EthSwt_00047, SWS_EthSwt_00049, SWS_EthSwt_00051, SWS_EthSwt_00052, SWS_EthSwt_00053, SWS_EthSwt_00054, SWS_EthSwt_00056, SWS_EthSwt_00058, SWS_EthSwt_00060, SWS_EthSwt_00061, SWS_EthSwt_00062, SWS_EthSwt_00079, SWS_EthSwt_00080, SWS_EthSwt_00081, SWS_EthSwt_00082, SWS_EthSwt_00084, SWS_EthSwt_00086, SWS_EthSwt_00087, SWS_EthSwt_00088, SWS_EthSwt_00089, SWS_EthSwt_00090, SWS_EthSwt_00091, SWS_EthSwt_00092, SWS_EthSwt_00093, SWS_EthSwt_00094, SWS_EthSwt_00095, SWS_EthSwt_00098, SWS_EthSwt_00099, SWS_EthSwt_00106, SWS_EthSwt_00107, SWS_EthSwt_00108, SWS_EthSwt_00109,

		SWS_EthSwt_00110, SWS_EthSwt_00111, SWS_EthSwt_00112, SWS_EthSwt_00113, SWS_EthSwt_00114, SWS_EthSwt_00115, SWS_EthSwt_00116, SWS_EthSwt_00117, SWS_EthSwt_00118, SWS_EthSwt_00119, SWS_EthSwt_00120, SWS_EthSwt_00122, SWS_EthSwt_00123, SWS_EthSwt_00124, SWS_EthSwt_00125, SWS_EthSwt_00126, SWS_EthSwt_00127, SWS_EthSwt_00128, SWS_EthSwt_00129, SWS_EthSwt_00130, SWS_EthSwt_00132, SWS_EthSwt_00133, SWS_EthSwt_00134, SWS_EthSwt_00135, SWS_EthSwt_00136, 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, SWS_EthSwt_00153, SWS_EthSwt_00154, SWS_EthSwt_00155, SWS_EthSwt_00156, SWS_EthSwt_00157, SWS_EthSwt_00158, SWS_EthSwt_00162, SWS_EthSwt_00164, SWS_EthSwt_00165, SWS_EthSwt_00166, SWS_EthSwt_00167, SWS_EthSwt_00168, SWS_EthSwt_00169, SWS_EthSwt_00170, SWS_EthSwt_00171, SWS_EthSwt_00172, SWS_EthSwt_00173, SWS_EthSwt_00174, SWS_EthSwt_00175, SWS_EthSwt_00176, SWS_EthSwt_00177, SWS_EthSwt_00178, SWS_EthSwt_00179, SWS_EthSwt_00180, SWS_EthSwt_00181, SWS_EthSwt_00182, SWS_EthSwt_00183, SWS_EthSwt_00184, SWS_EthSwt_00185, SWS_EthSwt_00186, SWS_EthSwt_00187, SWS_EthSwt_00188, SWS_EthSwt_00189, SWS_EthSwt_00190, SWS_EthSwt_00191, SWS_EthSwt_00192, SWS_EthSwt_00193, SWS_EthSwt_00194, SWS_EthSwt_00195, SWS_EthSwt_00196, SWS_EthSwt_00197, SWS_EthSwt_00198, SWS_EthSwt_00199, SWS_EthSwt_00200, SWS_EthSwt_00201, SWS_EthSwt_00202, SWS_EthSwt_00203, SWS_EthSwt_00204, SWS_EthSwt_00205, SWS_EthSwt_00206, SWS_EthSwt_00207, SWS_EthSwt_00208, SWS_EthSwt_00209, SWS_EthSwt_00210, SWS_EthSwt_00211, SWS_EthSwt_00212, SWS_EthSwt_00213, SWS_EthSwt_00214, SWS_EthSwt_00215, SWS_EthSwt_00216, SWS_EthSwt_00217, SWS_EthSwt_00218, SWS_EthSwt_00219, SWS_EthSwt_00220, SWS_EthSwt_00221, SWS_EthSwt_00222, SWS_EthSwt_00223, SWS_EthSwt_00224, SWS_EthSwt_00225, SWS_EthSwt_00226, SWS_EthSwt_00227, SWS_EthSwt_00228, SWS_EthSwt_00229, SWS_EthSwt_00230, SWS_EthSwt_00231
SRS_ETH_00087	-	SWS_EthSwt_00060, SWS_EthSwt_00061,

		SWS_EthSwt_00086, 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_00128, 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_00228

7 Functional specification

7.1 Ethernet BSW stack

As part of the AUTOSAR Layered Software Architecture according to Figure 7-1, the Ethernet BSW modules also form a layered software stack.
Figure 7-1depicts 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.

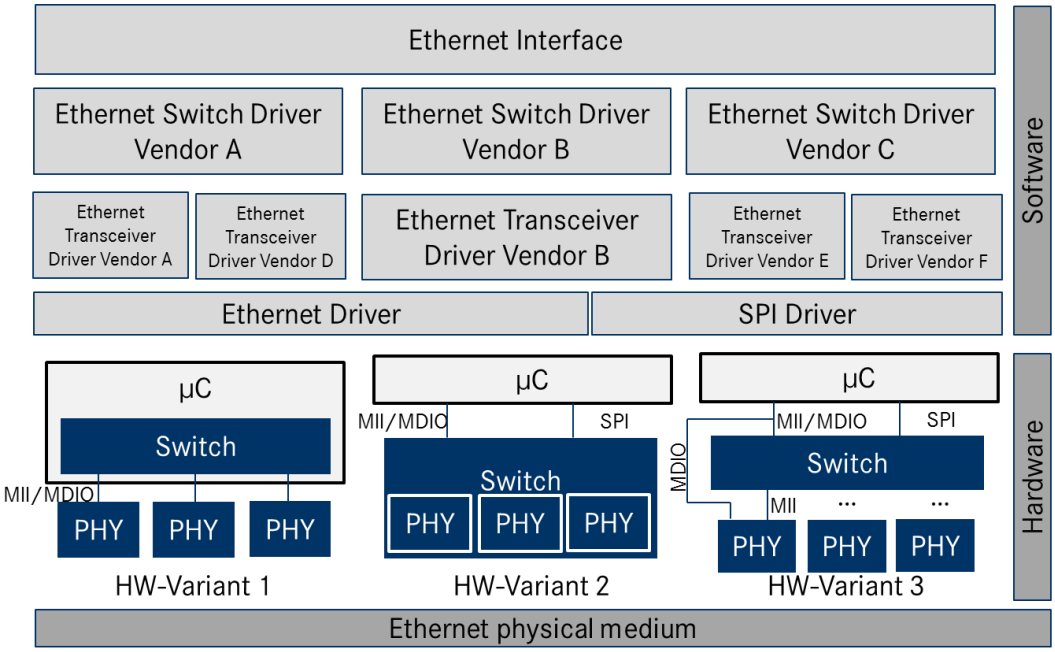


Figure 7-1 Basic Structure of the Ethernet BSW stack. (Note: The different hardware variants are alternative setups)

7.1.1 Indexing scheme

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

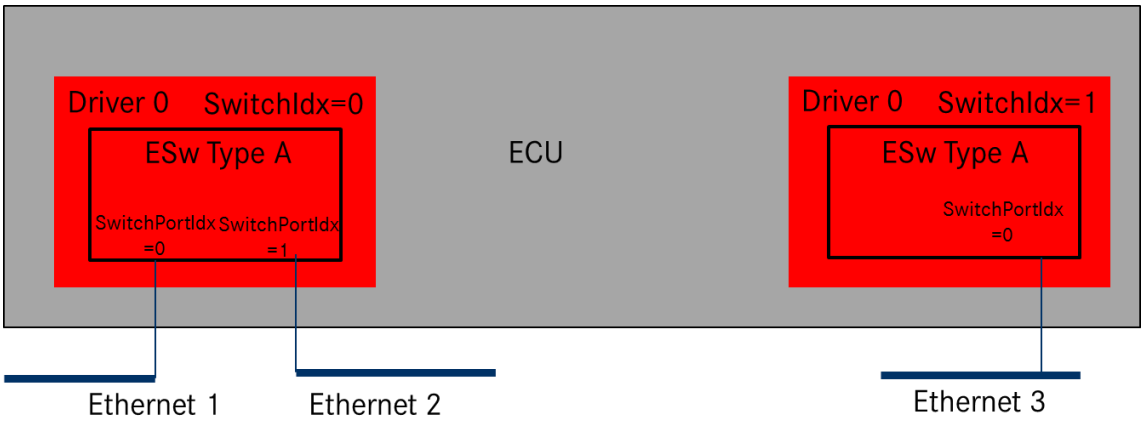


Figure 7-2 Ethernet Switch Driver indexing scheme

[SWS_EthSwt_00099] ⌈The Ethernet Switch Driver shall use a zero-based index to abstract the access for upper software layers.⌋ (SRS_ETH_00086)

[SWS_EthSwt_00130] ⌈The SwitchPortIdx is an index for a port at the switch.⌋ (SRS_ETH_00086)

[SWS_EthSwt_00120] ⌈The parameter EthSwtIdx within the configuration shall correspond to the argument used in the API.⌋ (SRS_ETH_00086)

[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_ETH_00086)

[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_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_ETH_00086)

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

[SWS_EthSwt_00226] [The switch driver shall support a learning phase which can be divided into several sequential steps.](SRS_ETH_00086)

Note: After assembly and initial power-up of the network, three learning phases follow which include MAC-Learning and IP-Address Assignment. Afterwards the learned parameters are stored to one or several non-volatile memories to make them available for subsequent start-ups. This process is shown in Figure 7-3. As an example for triggering this process, the DCM receives a diagnostic request via a bus system or a broadcast message in the Ethernet network. This diagnostic request can be forwarded to an SWC or CCD which triggers the auto-configuration process. However, the trigger is not part of this specification.

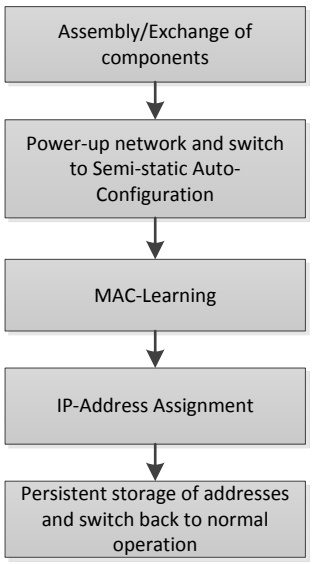


Figure 7-3 Learning Process

MAC-Learning (Optional Step): In this phase, messages need to be sent through the network and the switch will learn new MAC addresses (cf. Figure 7-4). These MAC-addresses will be stored in addition to predefined addresses, e.g. multicast MAC addresses which are configured during the vehicle network design. If static learning is executed, i.e. MAC address will be persistently stored, it might be possible to add dynamically learned entries in the tables.

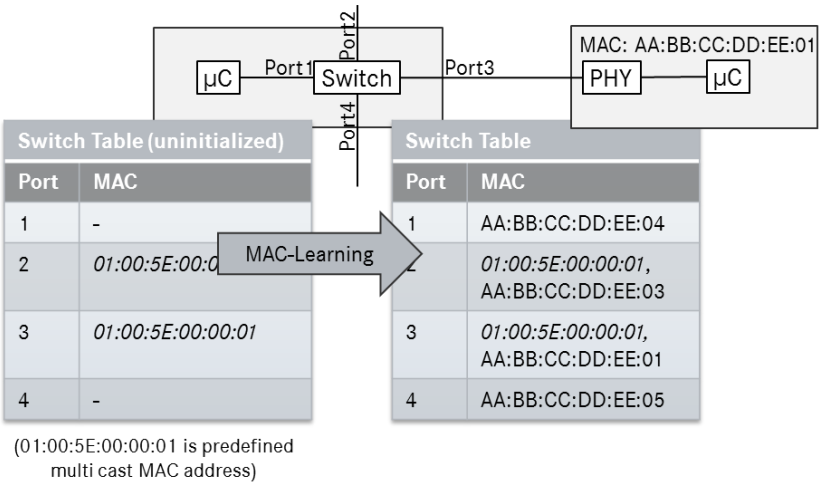


Figure 7-4 MAC-learning within the switch

IP-Address Assignment: In this phase, ECUs without a predefined IP-address will start to acquire an IP-address via DHCP (cf. Figure 7-5). Thus, these ECUs will run a DHCP-client while the ECU with the switch will run a DHCP server. In order to be able to assign always the same IP-address to a certain node, the DHCP server needs the information at which port the MAC address has been received. This port information can be interpreted as a “domain name” in the internet which is resolved to an IP address using a domain name server (DNS). With this port information the DHCP-server will assign the IP-address according to the IP-Assignment Table to the node. As mentioned above, this allows the assignment of MAC addresses by the Tier 1 and assignment of IP addresses by the OEM. With this mechanism it is also possible to assign different IP addresses to several VLANs at the same port. For this purpose, the IP-Assignment Table needs to be extended with a VLAN-column. Please note that the MAC-Learning-Phase can be combined with this phase.

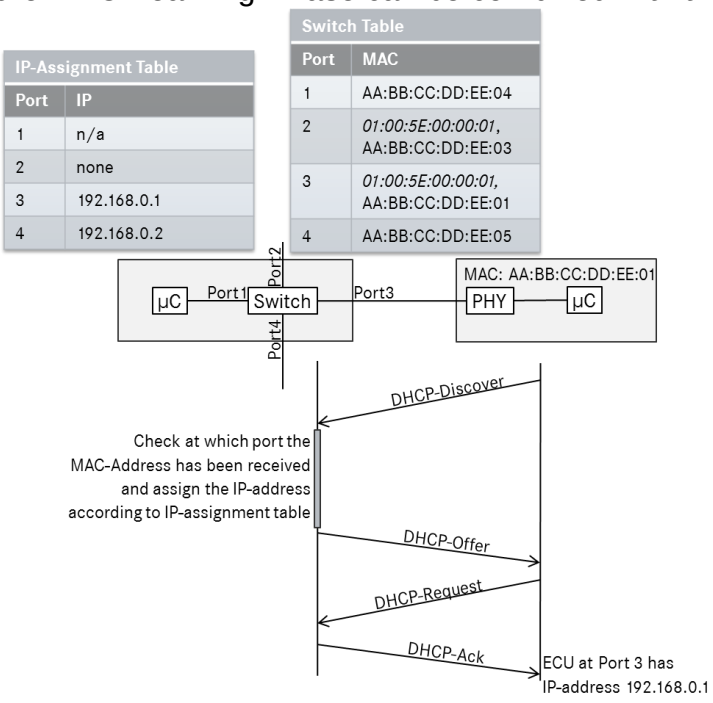


Figure 7-5 IP-address assignment via DHCP

[SWS_EthSwt_00136] ⌈The Ethernet Switch driver shall support an API which allows to store learned parameters like address resolution tables in a persistent manner by using the API EthSwt_StoreConfiguration. This persistent storage can be done in an NVRAM of the host CPU which runs the Ethernet Switch driver. Alternatively, this can be done in a memory of the switch itself. The trigger for storing the learned configuration or resetting the stored configuration can be done e.g. by a DCM.⌋(SRS_ETH_00086)

[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_00086)

[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_00086)

As shown in Figure 7-6, the switch consists of a certain number of ports. Each port has its own set of egress FIFOs in which the incoming packets will be buffered. How the messages in the FIFOs will be forwarded depends mainly on the shaping and port scheduling mechanisms. Thus, the parameterization of the egress port influences the latency of messages within the network. Please note that the egress port structures in Figure 7-6 are meant as an example. Other structures with different FIFO numbers are possible.

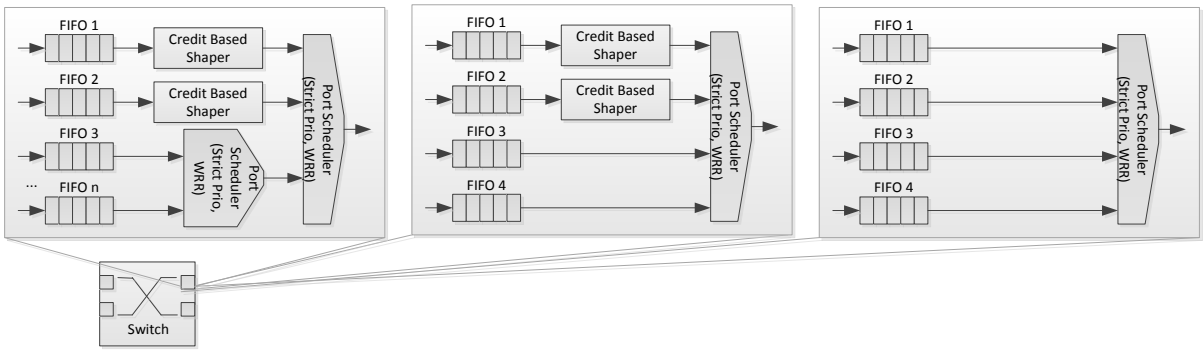


Figure 7-6 Ethernet Egress Port Structure

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

[SWS_EthSwt_00132] ⌈ The configuration of the Ethernet switch driver shall support different Ethernet egress port structures by the configuration EthSwtPortEgress. ⌋ (SRS_ETH_00086)

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_00086)

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 parameter EthSwtPortVlanForwarding.
⌋ (SRS_ETH_00086)

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_00086)

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_00086)

[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_00086)

7.2 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

⌋ (SRS_ETH_00086)

7.3 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_ETH_00086)

7.4 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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed:

[SWS_EthSwt_00002]Imported Types ⌈

Module	*Imported Type*
Dem	Dem_EventIdType
	Dem_EventStatusType
Eth_GeneralTypes	EthTrcv_BaudRateType
	EthTrcv_DuplexModeType
	EthTrcv_LinkStateType
	EthTrcv_ModeType
NvM	NvM_BlockIdType
	NvM_RequestResultType
Spi	Spi_AsyncModeType
	Spi_ChannelType
	Spi_DataBufferType
	Spi_NumberOfDataType
	Spi_SequenceType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋ (SRS_ETH_00086)

8.2 Type definitions

[SWS_EthSwt_00003] ⌈
EthSwt.h shall include Eth_GeneralTypes.h for include of general Ethernet stack type declarations. ⌋(SRS_ETH_00086)

[SWS_EthSwt_00004] ⌈
The types specified in SWS_EthernetSwitchDriver shall be declared in Eth_GeneralTypes.h⌋(SRS_ETH_00086)

8.2.1 EthSwt_StateType

[SWS_EthSwt_00123]EthSwt_StateType ⌈

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] EthSwt_ConfigType ⌈

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_MacVlanType

[SWS_EthSwt_00110]EthSwt_MacVlanType ⌈

Name:	EthSwt_MacVlanType		
Type:	Structure		
Element:	uint8[6]	MacAddr	Specifies the MAC address [0..255,0..255,0..255,0..255,0..255,0..255]
	uint16	VlanId	Specifies the VLAN address 0..65535
	uint8	SwitchPort	Port of the switch 0..255
Description:	The interpretation of this value is not specified, i.e. whether it is number of used bytes or number of used memory cells, etc.		

⌋ (SRS_ETH_00086)

8.2.4 EthSwt_MacLearningType

[SWS_EthSwt_00227]EthSwt_MacLearningType ⌈

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)

8.3 Function definitions

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

8.3.1 EthSwt_Init

[SWS_EthSwt_00006] EthSwt_Init ⌈

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_ETH_00086)

[SWS_EthSwt_00008] ⌈
The function EthSwt_Init shall change the state of the component from ETHSWT_STATE_UNINIT to ETHSWT_STATE_INIT. ⌋ (SRS_ETH_00086)

[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_ETH_00086)

8.3.2 EthSwt_SwitchInit

[SWS_EthSwt_00010] EthSwt_SwitchInit ⌈

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.] (SRS_ETH_00086)

[SWS_EthSwt_00012][
EthSwt_SwitchInit shall change the state of the component from ETHSWT_STATE_INIT to ETHSWT_STATE_ACTIVE.] (SRS_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

8.3.3 EthSwt_SetSwitchPortMode

[SWS_EthSwt_00018] EthSwt_SetSwitchPortMode [

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00022] ⌈
The function EthSwt_SetSwitchPortMode shall be pre compile time configurable On/Off by the configuration parameter: EthTrcvSetTransceiverModeApi. ⌋ (SRS_ETH_00086)

[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_ETH_00086)

[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_00086)

8.3.4 EthSwt_GetSwitchPortMode

[SWS_EthSwt_00025] EthSwt_GetSwitchPortMode ⌈

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00029] ⌈

The function EthSwt_GetSwitchPortMode shall be pre compile time configurable On/Off by the configuration parameter: EthTrcvGetTransceiverModeApi.] (SRS_ETH_00086)

[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_ETH_00086)

8.3.5 EthSwt_StartSwitchPortAutoNegotiation

[SWS_EthSwt_00031] EthSwt_StartSwitchPortAutoNegotiation ⌈

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00035] [
The function EthSwt_StartSwitchPortAutoNegotiation shall be pre compile time configurable On/Off by the configuration parameter: EthTrcvStartAutoNegotiationApi.] (SRS_ETH_00086)

[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_ETH_00086)

8.3.6 EthSwt_GetLinkState

[SWS_EthSwt_00037] EthSwt_GetLinkState ⌈

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00042] ⌈
The function EthSwt_GetLinkState shall be pre compile time configurable On/Off by the configuration parameter: EthTrcvGetLinkStateApi.] (SRS_ETH_00086)

[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_00086)

8.3.7 EthSwt_GetBaudRate

[SWS_EthSwt_00044] EthSwt_GetBaudRate ⌈

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00049] [
The function EthSwt_GetBaudRate shall be pre compile time configurable On/Off by the configuration parameter: EthTrcvGetBaudRateApi.] (SRS_ETH_00086)

[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 and if development error tracing is activated by EthSwtDevErrorDetect the ETHSWT_E_INV_API shall be raised.] (SRS_ETH_00086)

8.3.8 EthSwt_GetDuplexMode

[SWS_EthSwt_00051] EthSwt_GetDuplexMode [

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00056] ⌈
The function EthSwt_GetDuplexMode shall be pre compile time configurable On/Off by the configuration parameter: EthTrcvGetDuplexModeApi. ⌋ (SRS_ETH_00086)

[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_ETH_00086)

8.3.9 EthSwt_GetPortMacAddr

[SWS_EthSwt_00060] EthSwt_GetPortMacAddr ⌈

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_00086, 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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00230] [
The function EthSwt_GetPortMacAddr shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetPortMacAddrApi.] (SRS_ETH_00086)

8.3.10 EthSwt_GetArlTable

[SWS_EthSwt_00111] EthSwt_GetArlTable [

Service name:	EthSwt_GetArlTable	

Syntax:	`Std_ReturnType EthSwt_GetArlTable(`
	`  uint8 SwitchIdx,`
	`  uint32 startEntry,`
	`  uint32 numberOfElements,`
	`  EthSwt_MacVlanType[] ArlTable`
	`)`	

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

	startEntry	Number of elements which are skipped in the internal data structure of the switch table
Parameters (inout):	numberOfElements	In: Maximum number of elements which can be written in to the ArlTable Out: Number of elements which are returned in the ArlTable
Parameters (out):	ArlTable	Returns the ARL table of the switch consisting of a list of structs with MAC-address, VLAN-ID and port.
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: ARL table could not be obtained
Description:	Obtains the address resolution table of a switch	

] (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_00086, SRS_ETH_00087)

[SWS_EthSwt_00197] [
The ArlTable shall start with at the startEntry and shall contain the following number of elements defined by numberOfElements. The numberOf Elements shall not exceed the size of the ArlTable. All unused entries in the ArlTable shall be filled with zeros.] (SRS_ETH_00086, 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_ETH_00086)

[SWS_EthSwt_00229] [
The function EthSwt_GetArlTable shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetArlTableApi.] (SRS_ETH_00086)

8.3.11 EthSwt_GetBufferLevel

[SWS_EthSwt_00079] EthSwt_GetBufferLevel [

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00084] ⌈
The function EthSwt_GetBufferLevel shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetBufferLevelApi. ⌋ (SRS_ETH_00086)

8.3.12 EthSwt_GetDropCount

[SWS_EthSwt_00231] EthSwt_GetDropCount ⌈

Service name:	EthSwt_GetDropCount	

Syntax:	`Std_ReturnType EthSwt_GetDropCount(`
	`  uint8 SwitchIdx,`
	`  uint8 CountValues,`
	`  uint32[]* DropCount`
	`)`	

Service ID[hex]:	0x0c	
Sync/Async:	Synchronous /Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout):	CountValues	In: Maximal number of values which can be written von DropCount Out: Number of values which are in the returned in the DropCount list.
Parameters (out):	DropCount	The interpretation of this list of values is switch dependent
Return value:	Std_ReturnType	E_OK: success E_NOT_OK: drop counter could not be obtained
Description:	Reads a list with drop counter values of the corresponding switch.	

] (SRS_ETH_00086)

[SWS_EthSwt_00106] ⌈

The function EthSwt_GetDropCount shall read a list of values of the switch.
The meaning of these values is switch dependent. However, the list DropCount[] shall contain the following values in the given order, where the maximal possible value shall denote an invalid value, e.g. if this counter is not available:

1.) dropped packets due to buffer overrun
2.) dropped packets due to CRC errors
3.) number of undersize packets which were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise will formed. (see IETF RFC 1757)
4.) number of oversize packets which are longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed. (see IETF RFC 1757)
5.) number of alignment errors, i.e. packets which are received and are not an integral number of octets in length and do not pass the CRC.
6.) SQE test error according to IETF RFC1643 dot3StatsSQETestErrors
7.) The number of inbound packets which 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)
8.) total number of erroneous inbound packets
9.) The number of outbound packets which 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)
10.) total number of erroneous outbound packets
11.) 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)
12.) 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)
13.) 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)
14.) 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)

](SRS_ETH_00086)

[SWS_EthSwt_00107] [
If development error detection is enabled: the function EthSwt_GetDropCount 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_ETH_00086)

[SWS_EthSwt_00108] [
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_GetDropCount shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_ETH_00086)

[SWS_EthSwt_00109] ⌈
The function EthSwt_GetDropCount shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetDropCountApi. ⌋ (SRS_ETH_00086)

8.3.13 EthSwt_GetEtherStats

[SWS_EthSwt_00198] EthSwt_GetEtherStats ⌈

Service name:	EthSwt_GetEtherStats	

Syntax:	`Std_ReturnType EthSwt_GetEtherStats(`
	`  uint8 SwitchIdx,`
	`  uint8 SwitchPortIdx,`
	`  uint32[]* etherStats`
	`)`	

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):	etherStats	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 the following list according to IETF RFC2819, where the maximal possible value shall denote an invalid value, e.g. if this counter is not available: 1. etherStatsDropEvents 2. etherStatsOctets 3. etherStatsPkts 4. etherStatsBroadcastPkts 5. etherStatsMulticastPkts 6. etherStatsCrcAlignErrors 7. etherStatsUndersizePkts 8. etherStatsOversizePkts 9. etherStatsFragments 10. etherStatsJabbers 11. etherStatsCollisions 12. etherStatsPkts64Octets 13. etherStatsPkts65to127Octets 14. etherStatsPkts128to255Octets 15. etherStatsPkts256to511Octets 16. etherStatsPkts512to1023Octets 17. etherStatsPkts1024to1518Octets	

⌋ (SRS_ETH_00086)

[SWS_EthSwt_00199] ⌈
The function EthSwt_GetEtherStats shall read a list of values for a certain port of the switch according to IETF RFC 2819. ⌋(SRS_ETH_00086)

[SWS_EthSwt_00200] ⌈

If development error detection is enabled: the function EthSwt_GetEtherStats 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_ETH_00086)

[SWS_EthSwt_00201] [
If development error detection is enabled and the parameter SwitchIdx is not valid,EthSwt_GetEtherStats shall raise the development error ETHSWT_E_INV_SWITCH_IDX and return E_NOT_OK.] (SRS_ETH_00086)

[SWS_EthSwt_00202] [
The function EthSwt_GetEtherStats shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetEtherStatsApi.] (SRS_ETH_00086)

8.3.14 EthSwt_GetSwitchReg

[SWS_EthSwt_00206] EthSwt_GetSwitchReg [

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00210] [
The function EthSwt_GetSwitchReg shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetSwitchRegApi.] (SRS_ETH_00086)

8.3.15 EthSwt_SetSwitchReg

[SWS_EthSwt_00211] EthSwt_SetSwitchReg [

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00215] [

The function EthSwt_SetSwitchReg shall be pre compile time configurable On/Off by the configuration parameter: EthSwtSetSwitchRegApi.] (SRS_ETH_00086)

8.3.16 EthSwt_ReadTrcvRegister

[SWS_EthSwt_00216] EthSwt_ReadTrcvRegister [

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00220] [
The function EthSwt_ReadTrcvRegister shall be pre compile time configurable On/Off by the configuration parameter: EthSwtReadTrcvRegisterApi.] (SRS_ETH_00086)

8.3.17 EthSwt_WriteTrcvRegister

[SWS_EthSwt_00221] EthSwt_ReadTrcvRegister ⌈

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00225] ⌈
The function EthSwt_WriteTrcvRegister shall be pre compile time configurable On/Off by the configuration parameter: EthSwtWriteTrcvRegisterApi. ⌋ (SRS_ETH_00086)

8.3.18 EthSwt_EnableVlan

[SWS_EthSwt_00172] EthSwt_EnableVlan ⌈

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00177] [
The function EthSwt_EnableVlan shall be pre compile time configurable On/Off by the configuration parameter: EthSwtEnableVlanApi.] (SRS_ETH_00086)

8.3.19 EthSwt_StoreConfiguration

[SWS_EthSwt_00086] EthSwt_StoreConfiguration ⌈

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_00086)

[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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00090] ⌈
The function EthSwt_StoreConfiguration shall be pre compile time configurable On/Off by the configuration parameter: EthSwtStoreConfigurationApi. ⌋ (SRS_ETH_00086)

8.3.20 EthSwt_ResetConfiguration

[SWS_EthSwt_00091] EthSwt_ResetConfiguration ⌈

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_00086, 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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00095] [
The function EthSwt_ResetConfiguration shall be pre compile time configurable On/Off by the configuration parameter: EthSwtResetConfigurationApi.] (SRS_ETH_00086)

8.3.21 EthSwt_SetMacLearningMode

[SWS_EthSwt_00182] EthSwt_SetMacLearningMode [

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_00086, 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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00186] ⌈
The function EthSwt_SetMacLearningMode shall be pre compile time configurable On/Off by the configuration parameter: EthSwtSetMacLearningModeAPI. ⌋ (SRS_ETH_00086)

8.3.22 EthSwt_GetMacLearningMode

[SWS_EthSwt_00187] EthSwt_GetMacLearningMode ⌈

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_00086, 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_ETH_00086)

[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_ETH_00086)

[SWS_EthSwt_00191] ⌈
The function EthSwt_GetMacLearningMode shall be pre compile time configurable On/Off by the configuration parameter: EthSwtGetMacLearningModeApi. ⌋ (SRS_ETH_00086)

8.3.23 EthSwt_NvmSingleBlockCallback

[SWS_EthSwt_00125] EthSwt_NvmSingleBlockCallback ⌈

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_00086, 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_00086, SRS_ETH_00087)

[SWS_EthSwt_00127] ⌈
The function EthSwt_NvmSingleBlockCallback shall always return E_OK according to SWS_NvM_00368. ⌋ (SRS_ETH_00086, 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_ETH_00086, SRS_ETH_00087)

[SWS_EthSwt_00129] ⌈
The function EthSwt_NvmSingleBlockCallback shall be pre compile time configurable On/Off by the existence of the container EthSwtNvm. ⌋ (SRS_ETH_00086)

8.3.24 EthSwt_GetVersionInfo

[SWS_EthSwt_00058] EthSwt_GetVersionInfo ⌈

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_ETH_00086)

8.4 Call-back notifications

8.4.1 <user>_LinkDown

[SWS_EthSwt_00117] <User>_LinkDown ⌈

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_00086, SRS_ETH_00087)

[SWS_EthSwt_00119] ⌈
The function <User>_LinkDown shall be pre compile time configurable by the <user> with content of EthSwtLinkDownUser. ⌋ (SRS_ETH_00086)

8.4.2 <user>_LinkUp

[SWS_EthSwt_00203] <User>_LinkUp ⌈

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_00086, 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_ETH_00086)

8.4.3 <user>_PersistentConfigurationResult

[SWS_EthSwt_00193] <User>_PersistentConfigurationResult ⌈

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_00086, SRS_ETH_00087)

[SWS_EthSwt_00195] ⌈

The function <User>_PersistentConfigurationResult shall be pre compile time configurable by the <user> with content of EthSwtPersistentConfigurationResult.] (SRS_ETH_00086)

8.5 Scheduled functions

[SWS_EthSwt_00114] EthSwt_MainFunction [

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_ETH_00086)

[SWS_EthSwt_00122] [
The EthSwt_MainFunction shall call the API EthSwt_GetDropCount and shall check each single value of DropCount[]:

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⌋ (SRS_ETH_00086)

[SWS_EthSwt_00116] ⌈
If development error detection for the module EthSwt is enabled the function EthSwt_MainFunction shall raise the development error ETHSWT_E_UNINIT in case it was called before the EthSwt has been initialized. ⌋ (SRS_ETH_00086)

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] Optional Interfaces ⌈

API function	*Description*
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. OBD Events Suppression shall be ignored for this computation.
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
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.

] (SRS_ETH_00086)

[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_00086)

8.6.3 Configurable interfaces

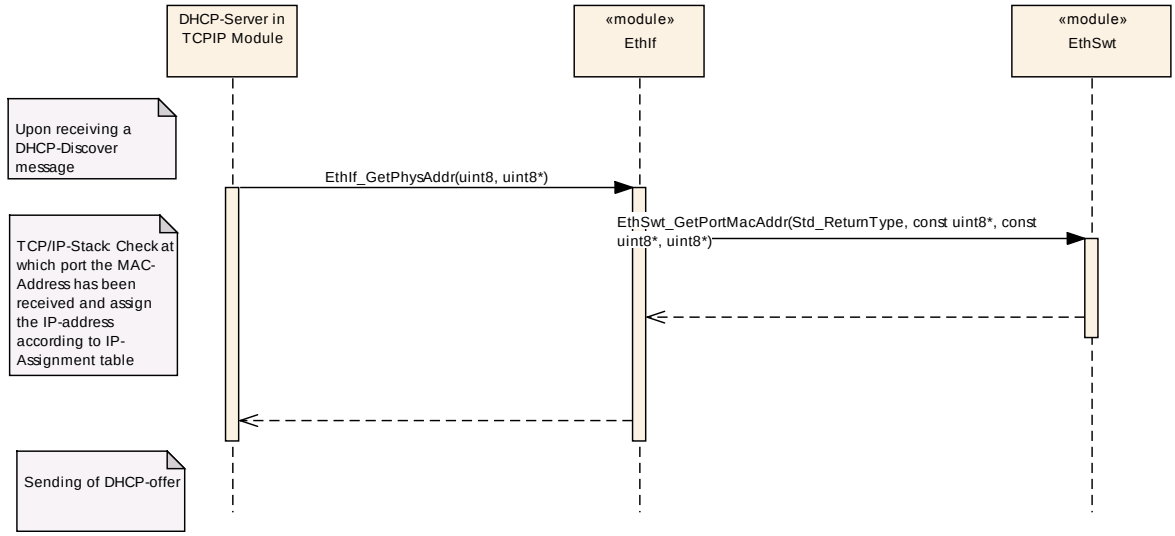
No configurable interfaces defined.

8.7 Service Interfaces

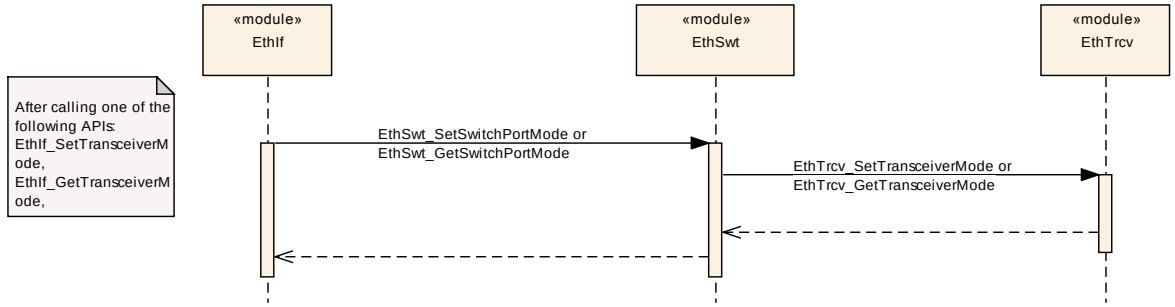
No direct access is necessary from the application layer.

9 Sequence diagrams

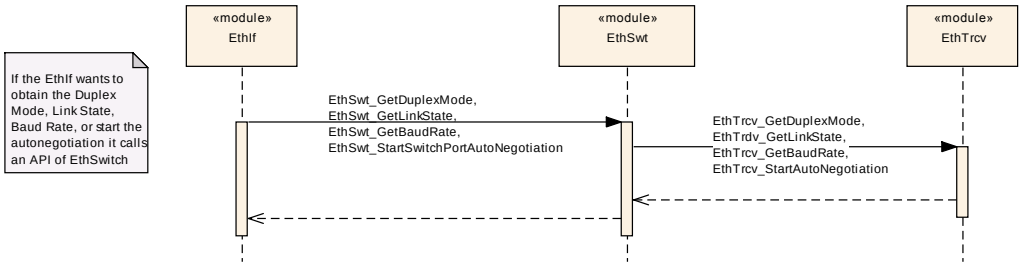
The following sequence diagram shows the interaction between the DHCP-Server in the TCP/IP-module and the Ethernet Switch Driver:



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:



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 0. Please note that the functional behavior of the ingress and egress port of a switch is implemented in hardware in the switch devices (see [8]). Thus, the configuration as shown in Figure 10-2 has to be written to the switch device.

10.1.1 Variants

[SWS_EthSwt_00159] ⌈
VARIANT-PRE-COMPILE only supports pre-compile time configurable parameters. ⌋ (SRS_BSW_00345)

[SWS_EthSwt_00160] ⌈
VARIANT-LINK-TIME includes mainly link-time and some pre-compile configurable parameters. ⌋ (SRS_BSW_00344)

[SWS_EthSwt_00161] ⌈
VARIANT-POST-BUILD includes post-build-time, link-time and some pre-compile time configurable parameters. ⌋ (SRS_BSW_00404, SRS_BSW_00405)

10.1.2 EthSwt

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

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthSwtConfig	1..*	Configuration of one Ethernet Switch.
EthSwtGeneral	1	General configuration of Ethernet Switch Driver module.

10.1.3 EthSwtConfig

SWS Item	**ECUC_EthSwt_00001 :**
Container Name	EthSwtConfig
Description	Configuration of one Ethernet Switch.

Configuration Parameters

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		

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_ReportErrorStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
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).

10.1.4 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_ReportErrorStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
Configuration Parameters	

SWS Item	**ECUC_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 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

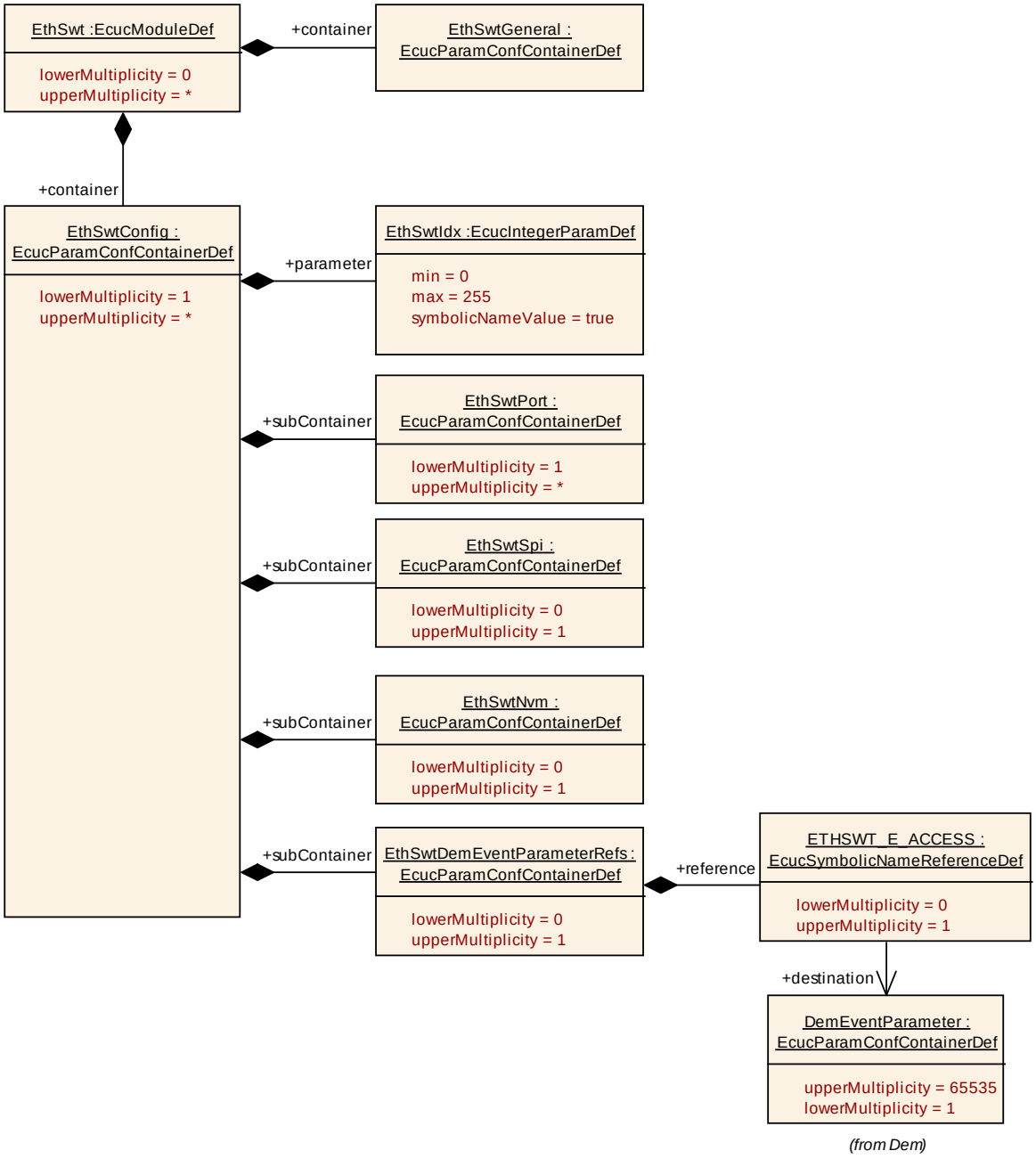


Figure 10-1: EthSwt

10.1.5 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 Default Error Tracer (Det) detection and notification ON or OFF. • true: enabled (ON). • false: disabled (OFF).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
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	--		
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	--		
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	--		
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_GetDropCount API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

No Included Containers

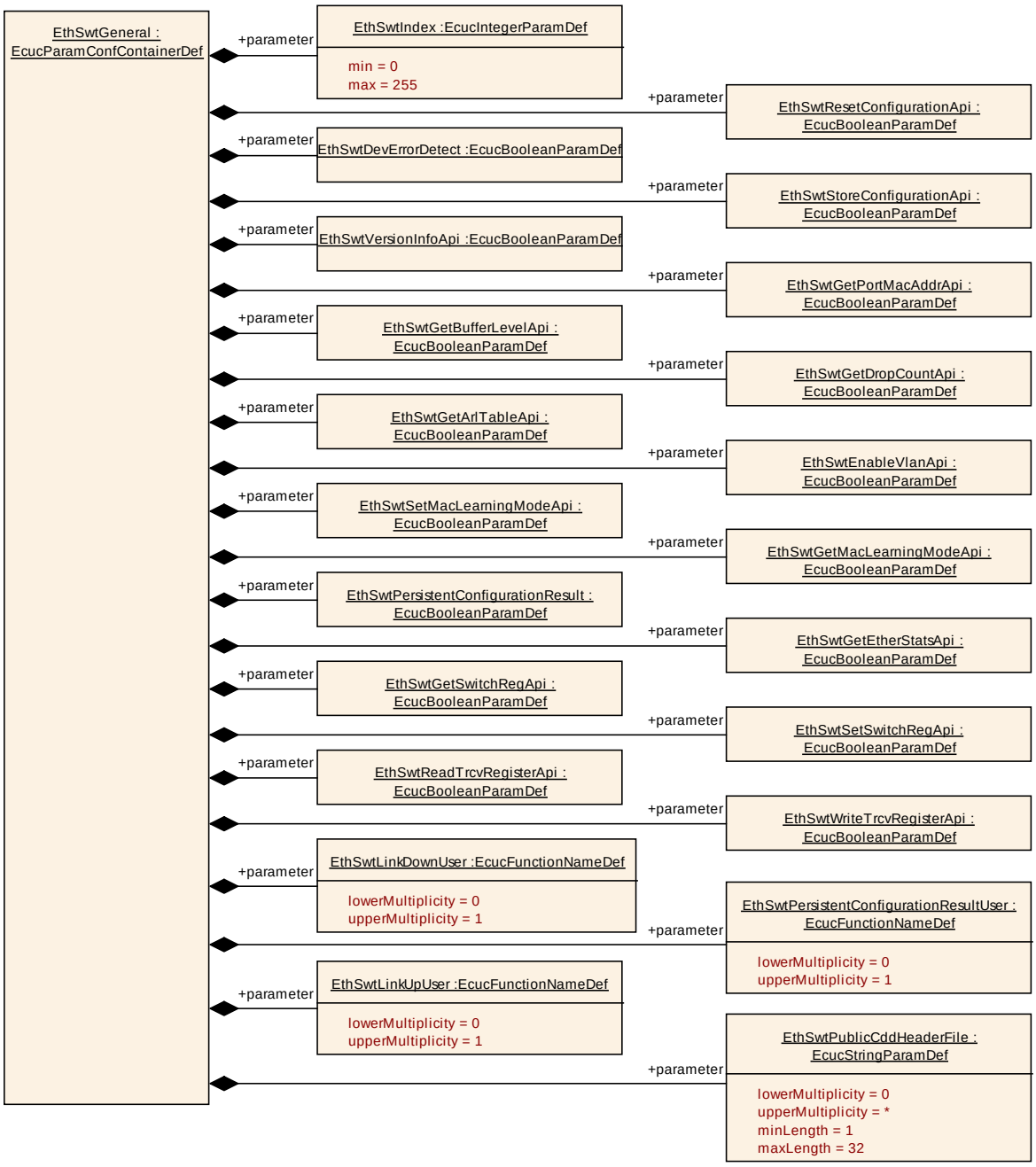


Figure 10-2: EthSwtGeneral

10.1.6 EthSwtPort

SWS Item	**ECUC_EthSwt_00005 :**
Container Name	EthSwtPort
Description	Configuration of one Ethernet Switch Port.
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	--		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
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_00054 :**		
Name	EthSwtPortPhysicalLayerType		
Description	Defines the physical layer type on this EthSwtPort.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	ETHSWT_PORT_BASE_T		Physical layer: baseT
	ETHSWT_PORT_BROAD_R_REACH		Physical layer: broadRReach
	ETHSWT_PORT_RTPGE		Physical layer: rtpge
	ETHSWT_PORT_X_MII		Physical layer: xMII
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	--	
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 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_00022 :**		
Name	EthSwtPortSpeed		
Description	Defines the communication speed in Mbit per second on this EthSwtPort in case no EthSwtPortTrcvRef is defined. Is optional if EthSwtPortTrcvRef is defined.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	ETHSWT_PORT_SPEED_10		Communication speed of 10 Mbit per second.
	ETHSWT_PORT_SPEED_100		Communication speed of 100 Mbit per second.
	ETHSWT_PORT_SPEED_1000		Communication speed of 1000 Mbit per second.
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_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 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		

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.

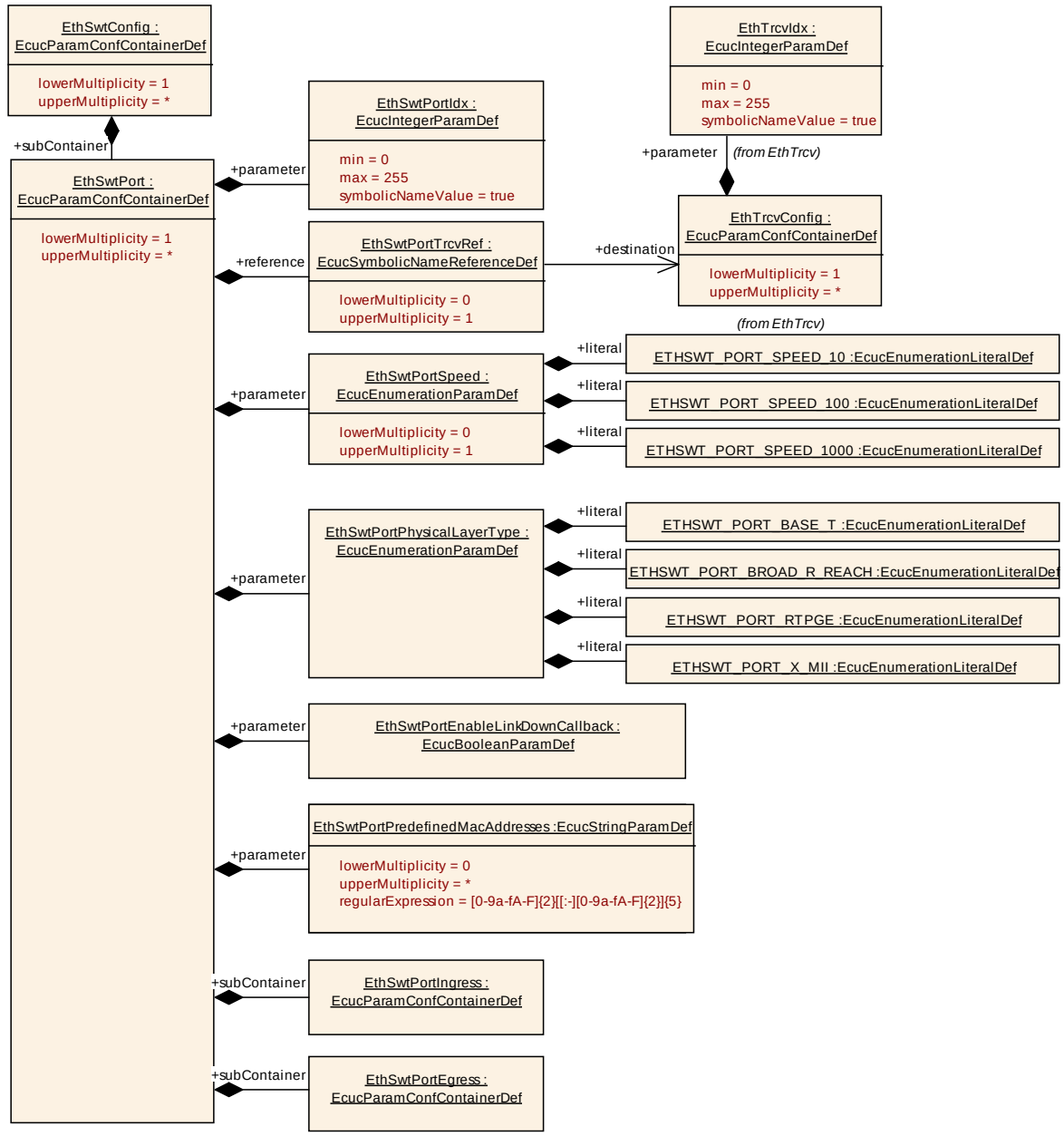


Figure 10-3: EthSwtPort

Please note that the functional behavior of the ingress and egress port of a switch is implemented in hardware in the switch devices (see [8]). Thus, the configuration as shown in Figure 10-3 and described in the following has to be written to the switch device or is related to the switch configuration.

10.1.7 EthSwtPortIngress

SWS Item	**ECUC_EthSwt_00014 :**
Container Name	EthSwtPortIngress
Description	Configuration of one Ethernet Switch Port ingress behavior.
Configuration Parameters	

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	1 .. 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		

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 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		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
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).

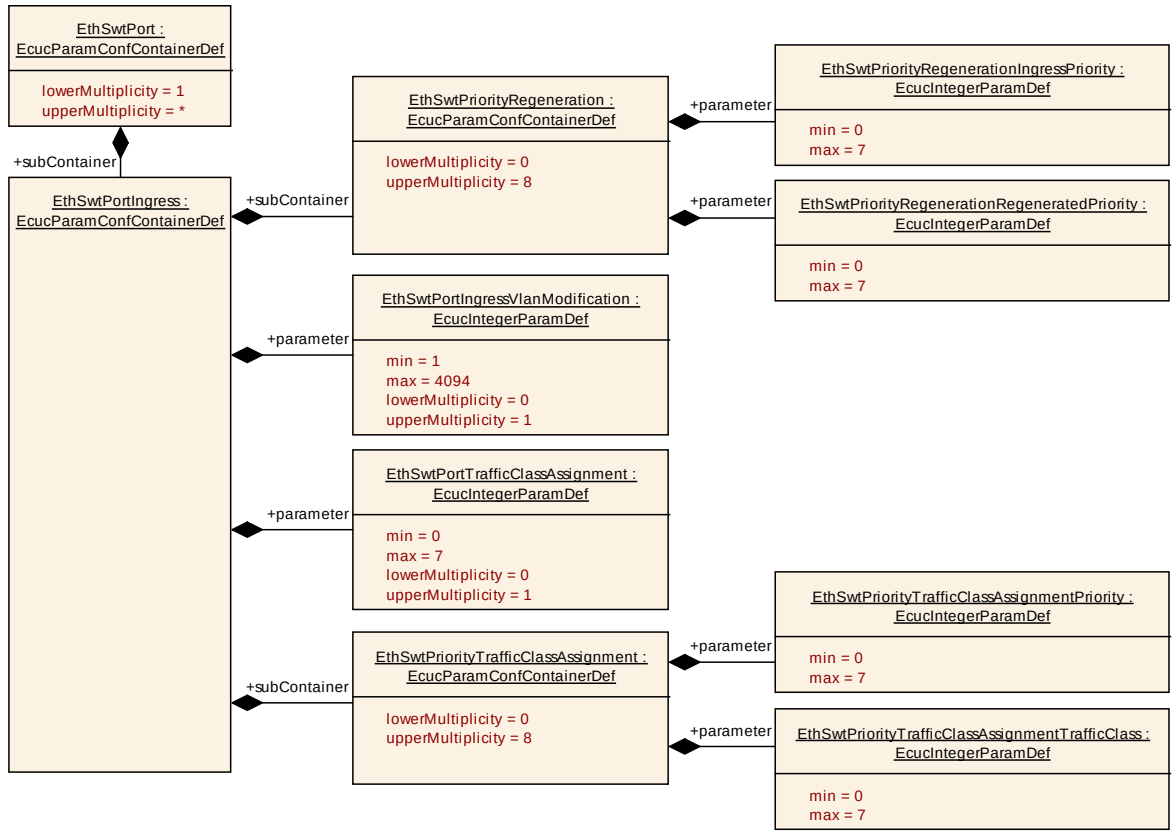


Figure 10-4: EthSwtPortIngress

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.
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).
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]		
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*
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.
EthSwtPortVlanForwarding	0..*	Defines how messages with a specific VLAN Id shall be handled at this egress port wrt. their VLAN Id.

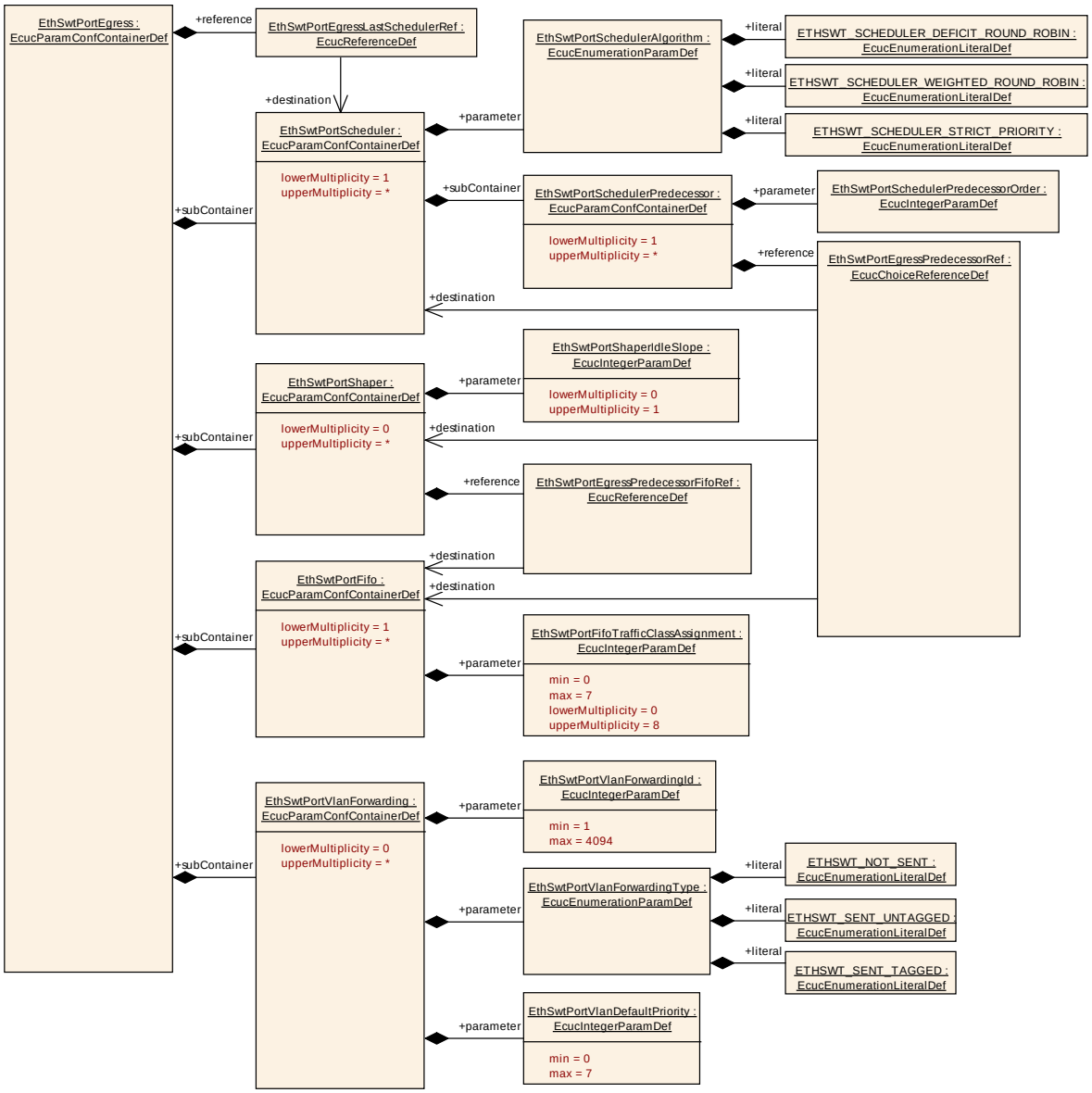


Figure 10-5: EthSwtPortEgress

10.1.11 EthSwtPortScheduler

SWS Item	**ECUC_EthSwt_00017 :**
Container Name	EthSwtPortScheduler
Description	Represents a Scheduler in the egress port.
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.
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]		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
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.		
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]		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
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.		
Configuration Parameters			

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 EthSwtPortVlanForwarding

SWS Item	**ECUC_EthSwt_00024 :**
Container Name	EthSwtPortVlanForwarding
Description	Defines how messages with a specific VLAN Id shall be handled at this egress port wrt. their VLAN Id.
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_00025 :**		
Name	EthSwtPortVlanForwardingId		
Description	Determines the VLAN Id the VlanForwarding shall apply to.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 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		

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		

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

No Included Containers

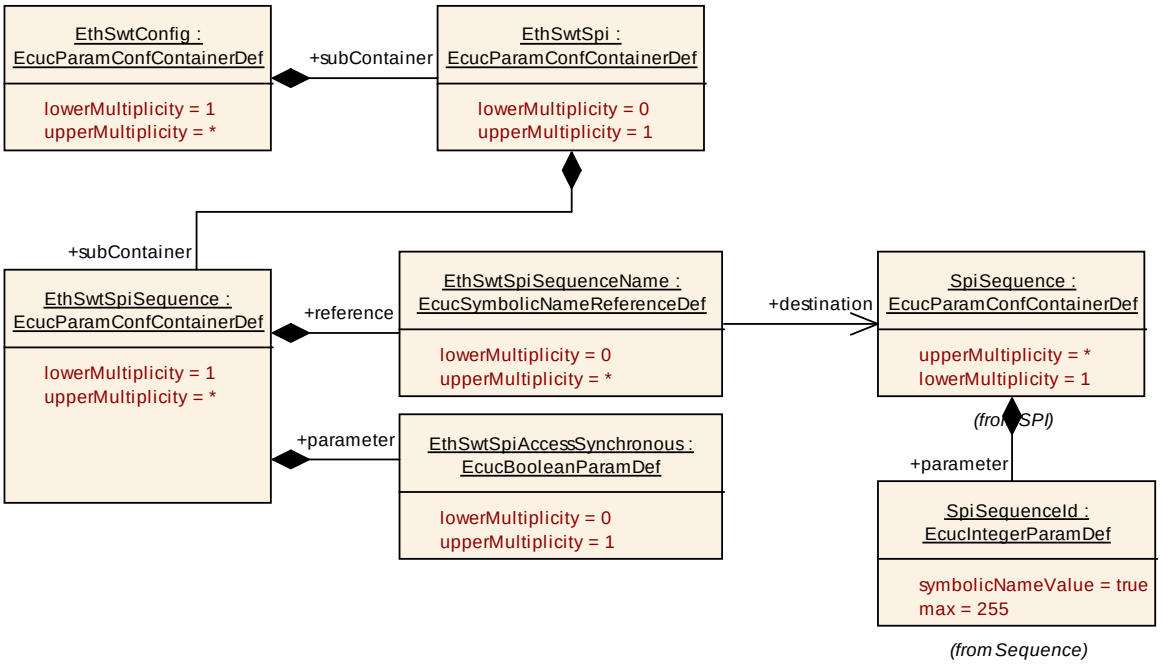


Figure 10-6: EthSwt Spi interaction

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

No Included Containers

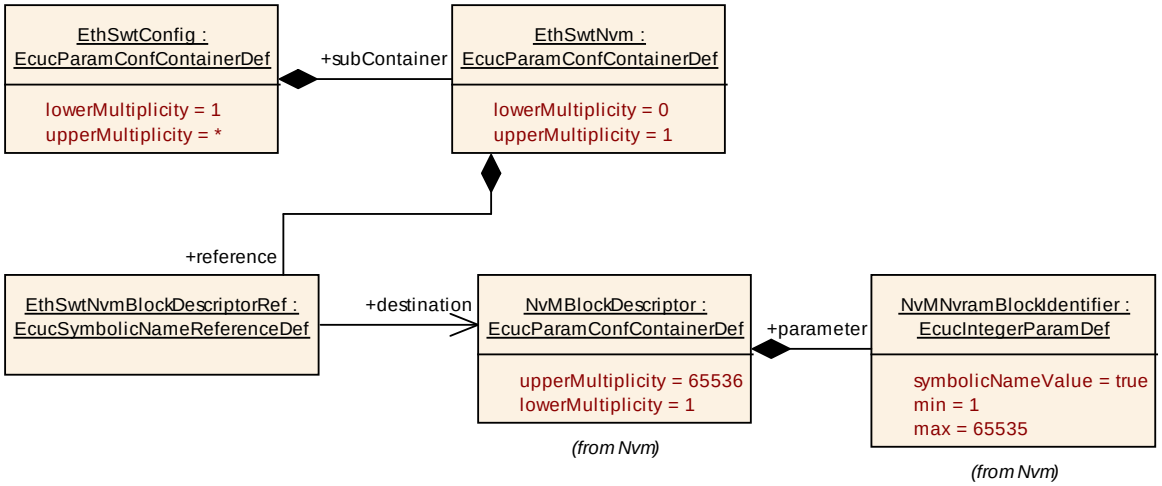


Figure 10-7: EthSwt NvM interaction