MICROSAR Ethernet Switch Driver

Technical Reference

Generic Ethernet Switch Driver

Version 1.00.00

Authors	Yacine Ouldboukhitine
Status	Released

Document Information

History

Author	Date	Version	Remarks
Yacine Ouldboukhitine	2019-02-25	1.00.00	Creation and first implementation

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_EthernetSwitchDriver.pdf	4.3.1
[2]	AUTOSAR	AUTOSAR_SWS_DEM.pdf	5.0.0
[3]	AUTOSAR	AUTOSAR_BasicSoftwareModules.pdf	V1.0.0

Scope of the Document

This technical reference describes the general use of the 3rd Party Generic Ethernet Switch Driver.

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

3.2.1 Naming conventions definition ... 11
3.2.2 Usages ... 12
3.3.1 Features list ... 13
3.3.2 Additions/ Extensions ... 17
3.3.3 Limitations ... 17
3.3.3.1 Ethernet Switch SPI configurations ... 17
3.3.3.2 Ethernet Switch Port configurations ... 18
3.3.3.3 Ethernet Switch NVM configurations ... 18
3.4.1 Driver Initialization ... 18
3.4.2 Switch Initialization ... 20
3.4.2.1 Synchronous downloading method ... 20
3.4.2.2 Asynchronous downloading method ... 21
3.6.1 Usage ... 24
3.6.2 Configurations ... 25
3.7.1 Development Error Reporting ... 25
3.7.2 Production Code Error Reporting ... 25
3.7.2.1 Definition ... 25
3.7.2.2 Usage ... 27
4.1.1 Static Files ... 30
4.1.2 Dynamic Files ... 30
4.4.1 Compiler Abstraction define ... 34
4.4.2 Include ... 34
4.4.3 Local/Global Constant macros ... 35
4.4.4 Local/Global Functions macros ... 35
4.4.5 Local/Global Data types and structures ... 36
4.4.6 Local/Global data prototypes ... 37
4.4.7 Local/Global data ... 38
4.4.8 Local/Global data prototypes ... 39
4.4.9 Local/Global functions prototypes ... 40
4.4.10 Configuration structure ... 40
4.4.11 Local functions ... 41
4.4.12 Functions implementation ... 42
4.4.13 DEM events ... 42
5.2.1 EthSwt_<VendorID>_<DriverName>_InitMemory ... 45
5.2.2 EthSwt_<VendorID>_<DriverName>_Init ... 46
5.2.3 EthSwt_<VendorID>_<DriverName>_SwitchInit ... 47
5.2.4 EthSwt_<VendorID>_<DriverName>_VSwitchInit ... 48
5.2.5 EthSwt_<VendorID>_<DriverName>_SetSwitchPortMode ... 49
5.2.6 EthSwt_<VendorID>_<DriverName>_GetSwitchPortMode ... 50

5.2.7 EthSwt_<VendorID>_<DriverName>_StartSwitchPortAutoNegotiation 51
5.2.8 EthSwt_<VendorID>_<DriverName>_GetLinkState 52
5.2.9 EthSwt_<VendorID>_<DriverName>_GetBaudRate 53
5.2.10 EthSwt_<VendorID>_<DriverName>_GetDuplexMode 54
5.2.11 EthSwt_<VendorID>_<DriverName>_GetPortMacAddr 55
5.2.12 EthSwt_<VendorID>_<DriverName>_GetArlTable 56
5.2.13 EthSwt_<VendorID>_<DriverName>_GetBufferLevel 57
5.2.14 EthSwt_<VendorID>_<DriverName>_GetDropCount 58
5.2.15 EthSwt_<VendorID>_<DriverName>_GetEtherStats..... 59
5.2.16 EthSwt_<VendorID>_<DriverName>_GetSwitchReg 60
5.2.17 EthSwt_<VendorID>_<DriverName>_SetSwitchReg..... 61
5.2.18 EthSwt_<VendorID>_<DriverName>_ReadTrcvRegister 62
5.2.19 EthSwt_<VendorID>_<DriverName>_WriteTrcvRegister..... 63
5.2.20 EthSwt_<VendorID>_<DriverName>_EnableVlan..... 64
5.2.21 EthSwt_<VendorID>_<DriverName>_StoreConfiguration 65
5.2.22 EthSwt_<VendorID>_<DriverName>_ResetConfiguration..... 66
5.2.23 EthSwt_<VendorID>_<DriverName>_SetMacLearningMode 67
5.2.24 EthSwt_<VendorID>_<DriverName>_GetMacLearningMode..... 68
5.2.25 EthSwt_<VendorID>_<DriverName>_GetVersionInfo 69
5.2.26 EthSwt_<VendorID>_<DriverName>_MainFunction..... 70
5.2.27 EthSwt_<VendorID>_<DriverName>_UpdateMCastPortAssignment..... 71
5.2.28 EthSwt_<VendorID>_<DriverName>_MirrorInit..... 72
5.2.29 EthSwt_<VendorID>_<DriverName>_WritePortMirrorConfiguration . 73
5.2.30 EthSwt_<VendorID>_<DriverName>_ReadPortMirrorConfiguration. 74
5.2.31 EthSwt_<VendorID>_<DriverName>_GetPortMirrorState 75
5.2.32 EthSwt_<VendorID>_<DriverName>_SetPortMirrorState 76
5.2.33 ... EthSwt_<VendorID>_<DriverName>_DeletePortMirrorConfiguration 77

Illustrations

Figure 2-1 AUTOSAR 4.3 Architecture Overview 9
Figure 2-2 AUTOSAR 4.2 Architecture Overview 10
Figure 3-1 Overview Driver generations 11
Figure 3-2 InitMemory sequence 19
Figure 3-3 Init sequence 20
Figure 3-4 Main function Sequence 24
Figure 3-5 Main function time configuration 25
Figure 6-1 Ethernet Switch Driver name configuration 79
Figure 6-2 Ethernet Switch configuration 79

Tables

Table 1-1 Component history 7
Table 3-1 Naming conventions usages 12
Table 3-2 Supported AUTOSAR standard conform features 13
Table 3-3 Feature list supported definition 13
Table 3-4 Feature list generated definition 14
Table 3-5 Feature list supported and generated definition 14
Table 3-6 Feature list 17
Table 3-7 Features provided beyond the AUTOSAR standard 17
Table 3-8 Errors reported to DEM 26
Table 3-9 associate DEM events enumeration 28
Table 4-1 Generated files 30
Table 4-2 Compiler abstraction and Memory map 31
Table 4-3 User block sections in files 34
Table 5-1 EthSwt_30_Generic_ConfigType 44
Table 5-2 EthSwt_<VendorID>_<DriverName>_InitMemory 45
Table 5-3 EthSwt_<VendorID>_<DriverName>_Init 46
Table 5-4 EthSwt_<VendorID>_<DriverName>_SwitchInit 47
Table 5-5 EthSwt_<VendorID>_<DriverName>_VSwitchInit 48
Table 5-6 EthSwt_<VendorID>_<DriverName>_SetSwitchPortMode 49
Table 5-7 EthSwt_<VendorID>_<DriverName>_GetSwitchPortMode 50
Table 5-8 EthSwt_<VendorID>_<DriverName>_StartSwitchPortAutoNegotiation 51
Table 5-9 EthSwt_<VendorID>_<DriverName>_GetLinkState 52
Table 5-10 EthSwt_<VendorID>_<DriverName>_GetBaudRate 53
Table 5-11 EthSwt_<VendorID>_<DriverName>_GetDuplexMode 54
Table 5-12 EthSwt_<VendorID>_<DriverName>_GetPortMacAddr 55
Table 5-13 EthSwt_<VendorID>_<DriverName>_GetArlTable 56
Table 5-14 EthSwt_<VendorID>_<DriverName>_GetBufferLevel 57
Table 5-15 EthSwt_<VendorID>_<DriverName>_GetDropCount 58
Table 5-16 EthSwt_<VendorID>_<DriverName>_GetEtherStats 59
Table 5-17 EthSwt_<VendorID>_<DriverName>_GetSwitchReg 60
Table 5-18 EthSwt_<VendorID>_<DriverName>_SetSwitchReg 61
Table 5-19 EthSwt_<VendorID>_<DriverName>_ReadTrcvRegister 62
Table 5-20 EthSwt_<VendorID>_<DriverName>_WriteTrcvRegister 63
Table 5-21 EthSwt_<VendorID>_<DriverName>_EnableVlan 64
Table 5-22 EthSwt_<VendorID>_<DriverName>_StoreConfiguration 65
Table 5-23 EthSwt_<VendorID>_<DriverName>_ResetConfiguration 66
Table 5-24 EthSwt_<VendorID>_<DriverName>_SetMacLearningMode 67
Table 5-25 EthSwt_<VendorID>_<DriverName>_GetMacLearningMode 68
Table 5-26 EthSwt_<VendorID>_<DriverName>_GetVersionInfo 69

Table 5-27 EthSwt_<VendorID>_<DriverName>_MainFunction 70
Table 5-28 EthSwt_<VendorID>_<DriverName>_UpdateMCastPortAssignment 71
Table 5-29 EthSwt_<VendorID>_<DriverName>_MirrorInit 72
Table 5-30 EthSwt_<VendorID>_<DriverName>_WritePortMirrorConfiguration..... 73
Table 5-31 EthSwt_<VendorID>_<DriverName>_ReadPortMirrorConfiguration 74
Table 5-32 EthSwt_<VendorID>_<DriverName>_GetPortMirrorState..... 75
Table 5-33 EthSwt_<VendorID>_<DriverName>_SetPortMirrorState 76
Table 5-34 EthSwt_<VendorID>_<DriverName>_DeletePortMirrorConfiguration 77
Table 5-35 Services used by the EthSwt 78
Table 7-1 Glossary 81
Table 7-2 Abbreviations..... 81

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
1.00.xx	First implementation of the Generic Ethernet Switch Driver

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module EthSwt as specified in see [1].

Supported AUTOSAR Release*:	4.2.x	
Supported Configuration Variants:	Pre-compile	
Vendor ID:	ETHSWT _VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	ETHSWT_MODULE_ID	89 decimal (according to ref. **Error! Reference source not found.**)

* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

Note
The table above make only references to the “module” itself not to the Generated Drivers for more information’s please refer to 3.1 Naming Conventions

The complexity of the different use cases, possible Configurations/Implementations and the integrity of the MICROSAR Stack regarding the Standard Ethernet Switch driver could show the limits of the Standard component and cannot support the application specifications.

Vector provides a 3rd party Generic Ethernet Switch driver to solve this specific behavior. The Driver is compatible with our Tool configuration (DaVinci) and fully integrated in the MICROSAR stack.

The Generic Ethernet Switch driver consist of the standard Ethernet Switch driver API. Which are part of the AUTOSAR requirements, with additional users’ code sections which need to be fulfilled.

This document describes the implementation and configuration (functionality and API) of the MICROSAR Generic Ethernet Switch driver.

Caution
The user/integrator is responsible for the implementation
Please read the requirements and expectations of every functions carefully.
A wrong implementation could lead to a corruption of the entire MICROSAR Stack

2.1 Architecture Overview

The figure 2-1 shows where the EthSwt is located in the AUTOSAR architecture.

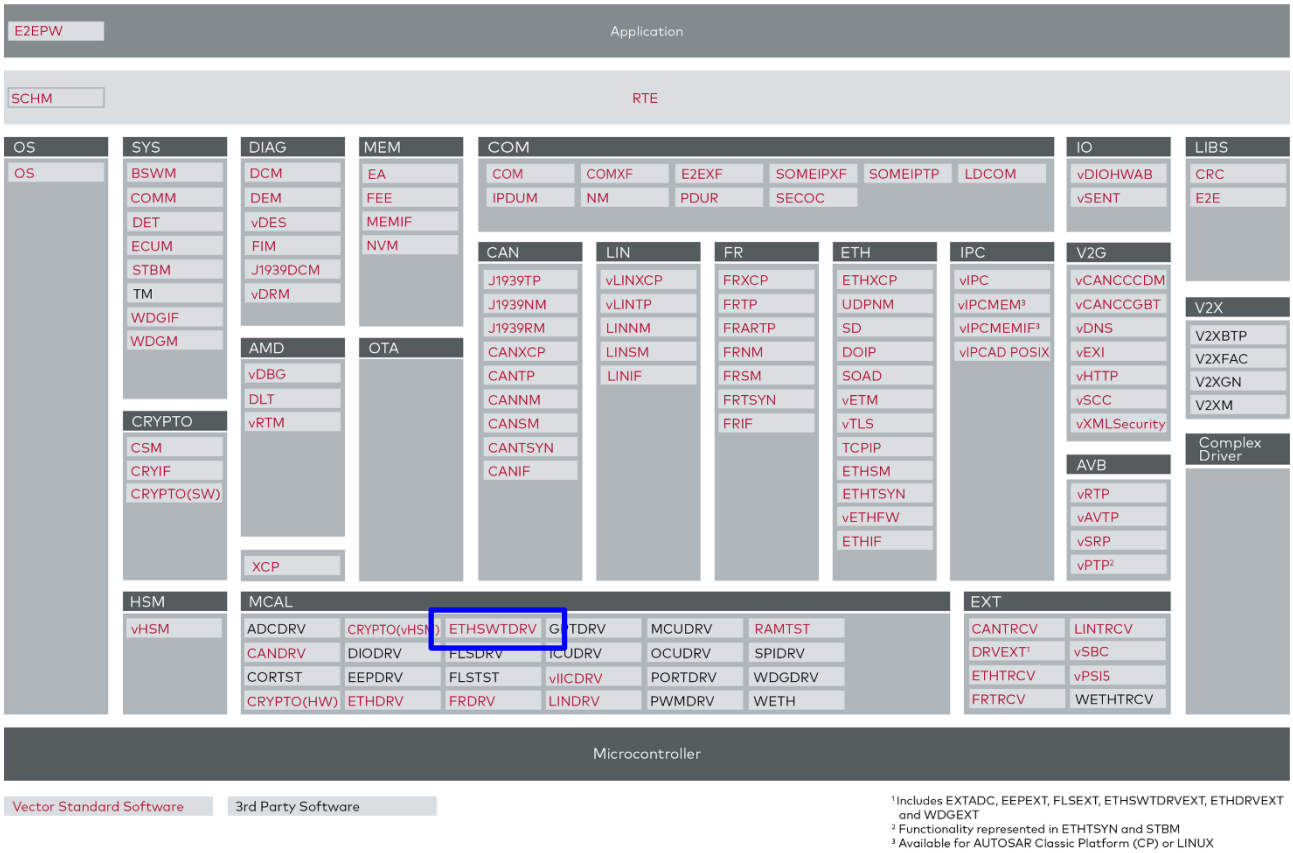


Figure 2-1 AUTOSAR 4.3 Architecture Overview

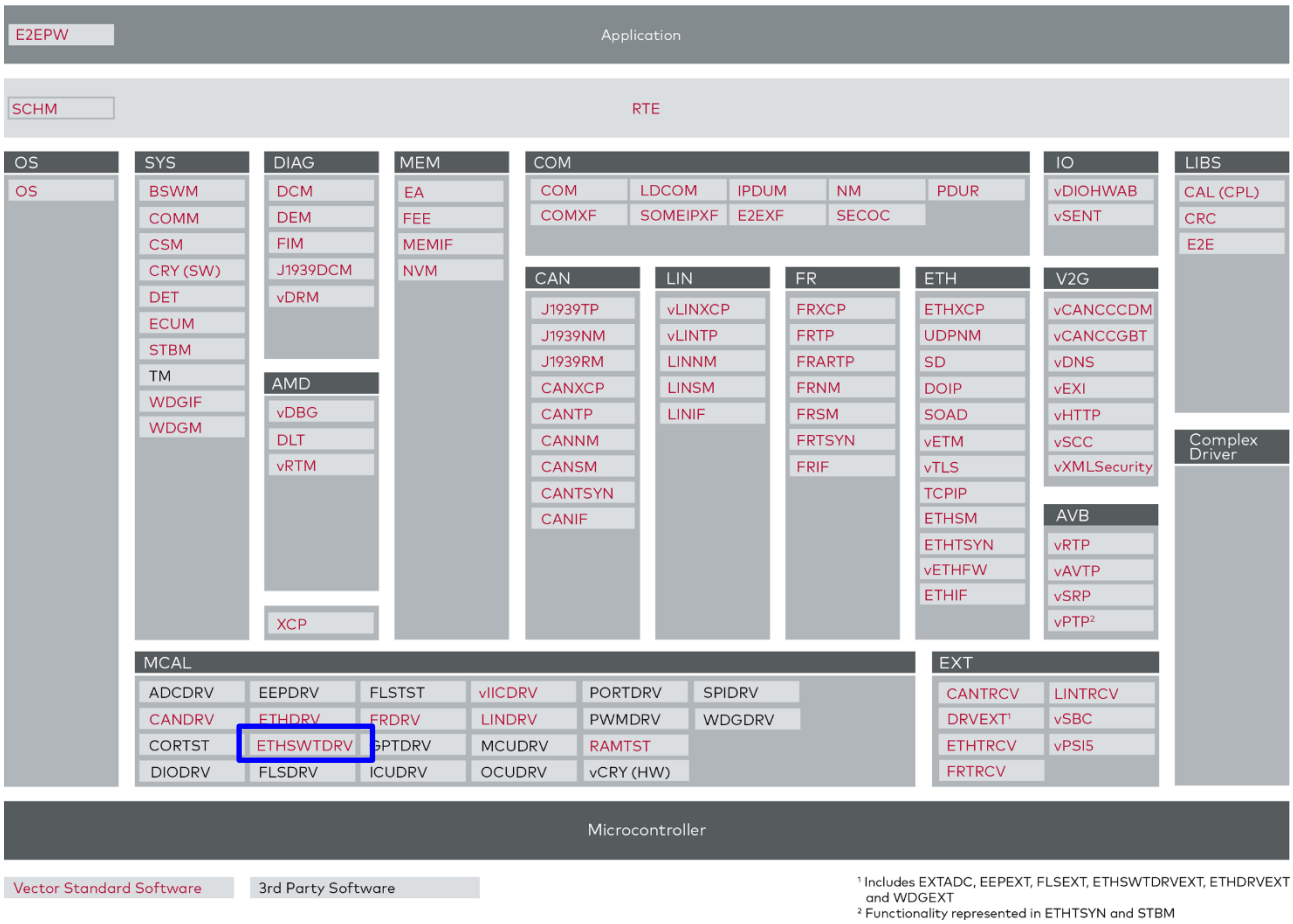


Figure 2-2 AUTOSAR 4.2 Architecture Overview

3 Functional Description

3.1 Global functional Description

The Global idea of the *Generic Ethernet Switch* driver is to provide one generated file which contains multiple drivers.

The drivers are configured via DaVinci tools and generated in one file. figure 4 illustrate the mechanism.

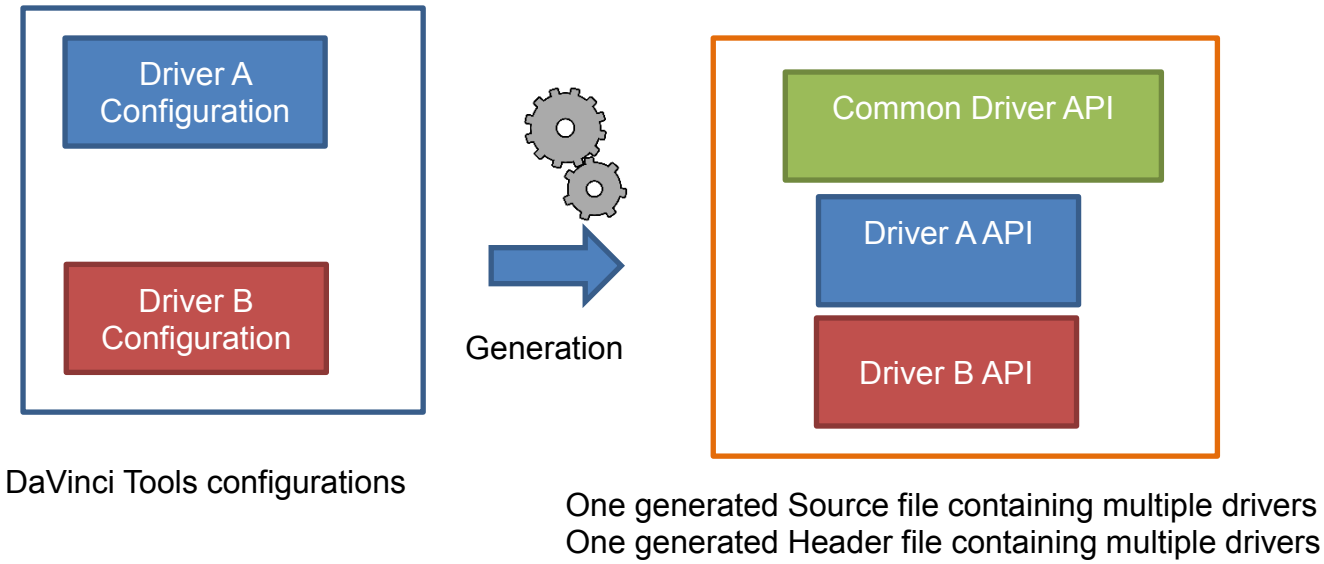


Figure 3-1 Overview Driver generations

The users/integrators should configure and implement the functionality for each generated driver configured.

3.2 Naming conventions

3.2.1 Naming conventions definition

The naming conventions is based on AUTOSAR rules and specifications, for each Generic Driver configured the users need to specify the Vendor ID and the driver name.

The Vendor ID is the number assigned by the AUTOSAR consortium for a Company member

E.g.: Vector Vendor ID **30**

The Driver name represents the Name of the *Generic Ethernet Switch Driver*, which is defined by the users/integrators.

E.g. Driver name: **MyEthernetSwitchDriver**

Caution
The Generic Ethernet Switch Driver names shall be unique

3.2.2 Usages

Based on the *Generic Ethernet Switch Driver* configuration (see Configuration)

The generated driver's functions, macros and configurations will be based on the naming rules Table 3-1

Type	Definition
Macro(s), Preprocessor(s)	EthSwt_<VendorId>_<DriverName>_MyDriverMacro
Variable(s)	EthSwt_<VendorID>_<DriverName>_MyDriverVariable
Function(s)	EthSwt_<VendorID>_<DriverName>_DriverFunctionName(…)

Table 3-1 Naming conventions usages

Example
Vendor ID: "*30*"
Driver name: "*Generated* "

void EthSwt_<*VendorID*>_<*DriverName*>_GetVersionInfo(..)
ETHSWT_<*VendorID*>_<*DriverName*>_GET_PORT_MAC_ADDR_API
vuint8 EthSwt_<*VendorID*>_<*DriverName*>_LocalReturnVal;

The naming rules defined above are important and shall not be modified manually, to prevent corruption (see User Code Integration).

The Ethernet Interface (**EthIf**) will used the *Generated Ethernet Switch* services by calling the APIs using the naming rules defined above.

There are no specifications to users/integrators to follow this rule during his own implementation. The users/integrators are free to define the variable name(s), function(s) and macro(s) as their own rules.

Further the users/integrators should take care that the definitions are concise and maintains the integrity of his own definition.

For example: global variables shall not create problems due to multiple usage in different driver.

3.3 Features

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the table

> Table 3-2 Supported AUTOSAR standard conform features

Vector Informatik provides further EthSwt functionality beyond the AUTOSAR standard. The corresponding features are listed in table

The following features specified in [1] are supported:

Supported/Unsupported AUTOSAR Standard Conform Features
Depends on the implementation of the 3rd party driver. *for more information please refer to the feature list table*

Table 3-2 Supported AUTOSAR standard conform features

3.3.1 Features list

The definition of Generation/Implementation of the *Generic Ethernet Switch Driver* differs from the Ethernet Switch standard component. To understand the feature list table and know the current availability read the definition of “**Supported**” and “**Generated**” below.

Supported

Supported	Description
✓	The feature is supported ** To see if the feature is Implemented/Generated see the description of “Supported and Generated”*
✘	The feature is not supported

Table 3-3 Feature list supported definition

Generated

Generated	Description
✓	The generated data or the feature itself is/are generated ** To see if the feature is Implemented/Generated see the description of "Supported and Generated"*
✗	The feature is not supported

Table 3-4 Feature list generated definition

Supported & Generated

Supported	Generated	Description
✓	✓	The feature or/and the generated data are supported and generated. *The feature can be used.*
✓	✗	The feature is supported but no generated data and/or APIs are available. The users/integrators must define this themselves. *The feature can be used.*
✗	✗	The feature is not supported. The generated data and APIs are not available. *The feature shall not be used.*

Table 3-5 Feature list supported and generated definition

The feature table below lists all the features and configurations available for the *Generic Ethernet Switch Driver*.

Feature	Short Description	Supported	Generated
Ethernet Switch Error Detection Report			
Development Error Detection	Detection and report errors	✗	✗
Detection Events manager **1	Report Dem events	✓	✗
DEM events parameters **1	Provide macro access to the Dem events configured	✓	✓
Ethernet Switch information			
Get Buffer level API	Provide API to read the Buffer level	✓	✓
Get DropCount API	Provide API to get the drop count value	✓	✓
Get Ethernet statistics API	Provide API to get the Ethernet Statistics	✓	✓
Get ARL table API	Provide API to get the current ARL table	✓	✓
Ethernet Switch MAC Learning mode			
Set Mac Learning mode API	Provide API to set the Mac learning mode	✓	✓
Get Mac Learning mode API	Provide API to get the Mac learning mode	✓	✓
Ethernet Switch registers management			
Get switch registers API	Provide API to get the content of a switch register	✓	✓
Set switch registers API	Provide API to set a value into a switch register	✓	✓
Ethernet Transceiver registers management			
Read Transceiver registers API	Provide API to read the content of the Transceiver register	✓	✓
Write Transceiver registers API	Provide API to write a value in the Transceiver register	✓	✓
Ethernet Switch state management			
Link Up user callback	Call a user function when the link is detected as up in the function GetLinkState()	✗	✗
Link Down use callback	Call a user function when the link is detected as down in the function GetLinkState()	✗	✗

Ethernet Switch MAC management			
Reset configuration API	Provide API to cancel the previous ARL table saved in the NVRAM	✓	✓
Store configuration API	Provide API to store the ARL Table to the NVRAM	✓	✓
Get port mode MAC Address API	Provide API to get the port corresponding to the MAC address	✓	✓
Ethernet Switch Frame management			
Update Multicast port Assignement API	Provide API to assign multicast on a specific port	✓	✓
Frame management support API	Provide API for the frame management	✗	✗
Global time support API	Provide API for the Global time support	✗	✗
Ethernet Switch Mirror mode			
Mirror mode	Mirror the received data from one port to another one	✓	✓
Delete port mirror configuration API	Provide API to delete an actual port mode configuration	✓	✓
Write port mirror configuration API	Provide API to write mirror mode configuration to a specific port	✓	✓
Set port mirror state API	Provide API to set the mirror mode status to a specific port	✓	✓
Get port mirror state API	Provide API to get the mirror state from a specific port	✓	✓
Read port mirror configuration API	Provide API to read port mirror configuration from a specific port	✓	✓
NVM			
Callback function EthSwtPersistentConfigurationResult	Callback function when EthSwtPersistentConfigurationResult is enabled	✗	✗
Mirroring NVM block **2	Reference to the NVM block to store the port mirror configuration	✓	✗
NVM block **2	Reference the NVM block used to store the ARL table	✓	✗
Ethernet Switch port configurations			
Ethernet Switch port configurations **3	Configure the Ethernet Switch port(s)	✓	✗
Ethernet Switch SPI configurations			

Ethernet Switch SPI configurations **3	Configure the Ethernet Switch SPI configuration(s)	✓	✗
Ethernet Switch Version			
Get version info API	Provide API to get the actual version of the Generic driver	✓	✓
Generic Ethernet Switch Driver			
Multiple Driver generation	Generated multiple Ethernet switch driver	✓	✓
Generic Ethernet switch driver name	Allow the users to configure the name of the Generic driver	✓	✓
Generic Ethernet switch Vendor ID	Allow the users to configure the Vendor ID of the Generic driver	✓	✓

Table 3-6 Feature list

1 DEM events parameters configured in **DaVinci are generated. DEM events reporting is not implemented in the functions (more information's see DEM usage).

**2 NVM Blocks configurations of the *Generic Ethernet Switch driver* are not generated (different from NVM module itself which could be generated if active).

(for more information's refer to 3.3.3 Limitations)

**3 The SPI and Port configurations configured in the *Generic Ethernet Switch* module will not be generated.

(different from the Port or SPI module themselves which could be generated if active).

(for more information's refer to 3.3.3 Limitations)

3.3.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond the AUTOSAR Standard
Generation of the *Generic Ethernet Switch Driver*

Table 3-7 Features provided beyond the AUTOSAR standard

3.3.3 Limitations

3.3.3.1 Ethernet Switch SPI configurations

The Ethernet Switch SPI configurations will not be generated

The Users/Integrators should (if needed) implement by its own.

3.3.3.2 Ethernet Switch Port configurations

The Ethernet Switch Port configurations will not be generated (only the symbolic names)

The Users/Integrators should (if needed) implement by its own.

3.3.3.3 Ethernet Switch NVM configurations

The Ethernet Switch NVM configurations will not be generated

The Users/Integrators should (if needed) implement by its own.

3.4 Initialization

3.4.1 Driver Initialization

The initialization is performed via 2 functions EthSwt_<VendorId>_<DriverName>Init() and EthSwt_<VendorId>_<DriverName>InitMemory().

EthSwt_<VendorId>_<DriverName>InitMemory() is used to prepare, clean or others steps which ensures that the Ethernet Switch Driver memory is initialized (more information EthSwt_<VendorID>_<DriverName>_InitMemory) .

Note

This function can be left empty if the memory is already initialized by another software module

EthSwt_<VendorId>_<DriverName>Init() is used to set the Ethernet Switch driver initialized (more information EthSwt_<VendorID>_<DriverName>_Init) .

Caution

Use only this function to initialize the Driver.

The Users/Integrators shall not load or download the configuration to the Ethernet Switch Device

The *Generic Ethernet Switch* driver provide the statics functions interfaces EthSwt_30_Generic_Init() and EthSwt_30_Generic_InitMemory(). These are used as entry point of the **EcuM** module to initialize the *Generic Ethernet Switch Driver.*

The registration of the static interface functions in the **EcuM** module is automatically performed, no action from user(s)/integrator(s) is required.

This “Static function” will call all the respective generic driver function configured during the MICROSAR stack initialization as shown in the examples below

Caution
Do not modify the content of the functions **EthSwt_30_Generic_Init()** and **EthSwt_30_Generic_InitMemory()**

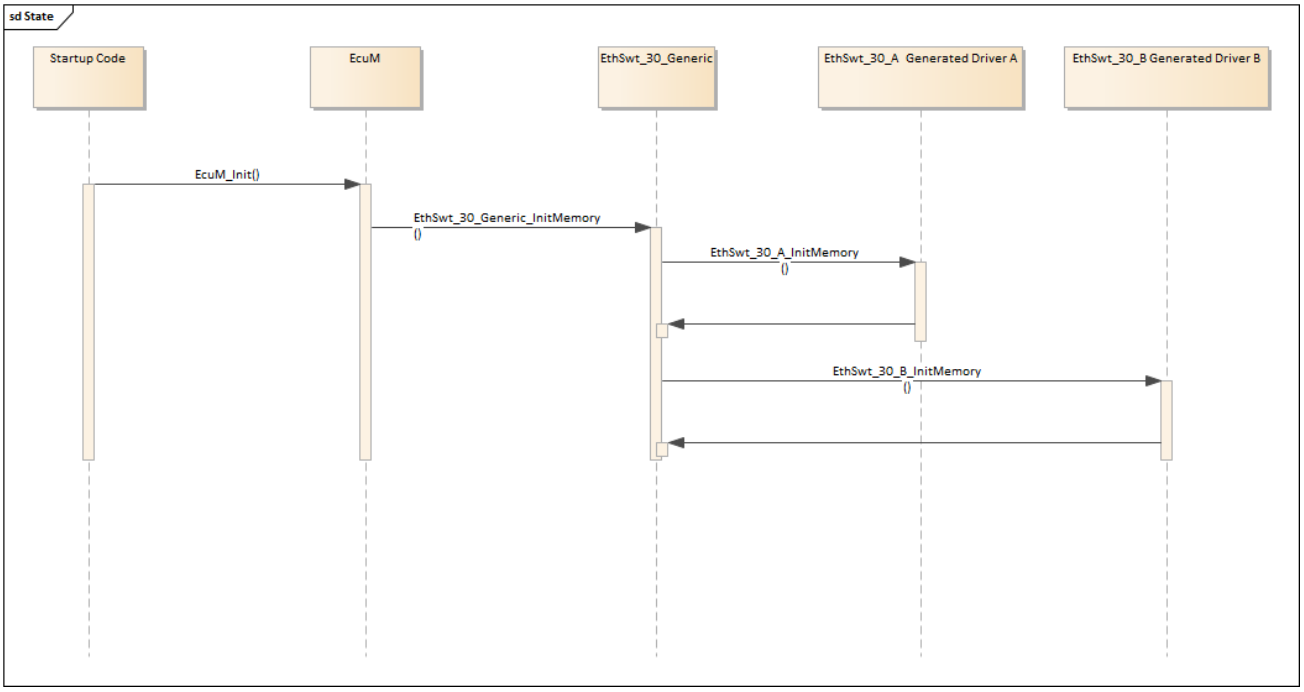


Figure 3-2 InitMemory sequence

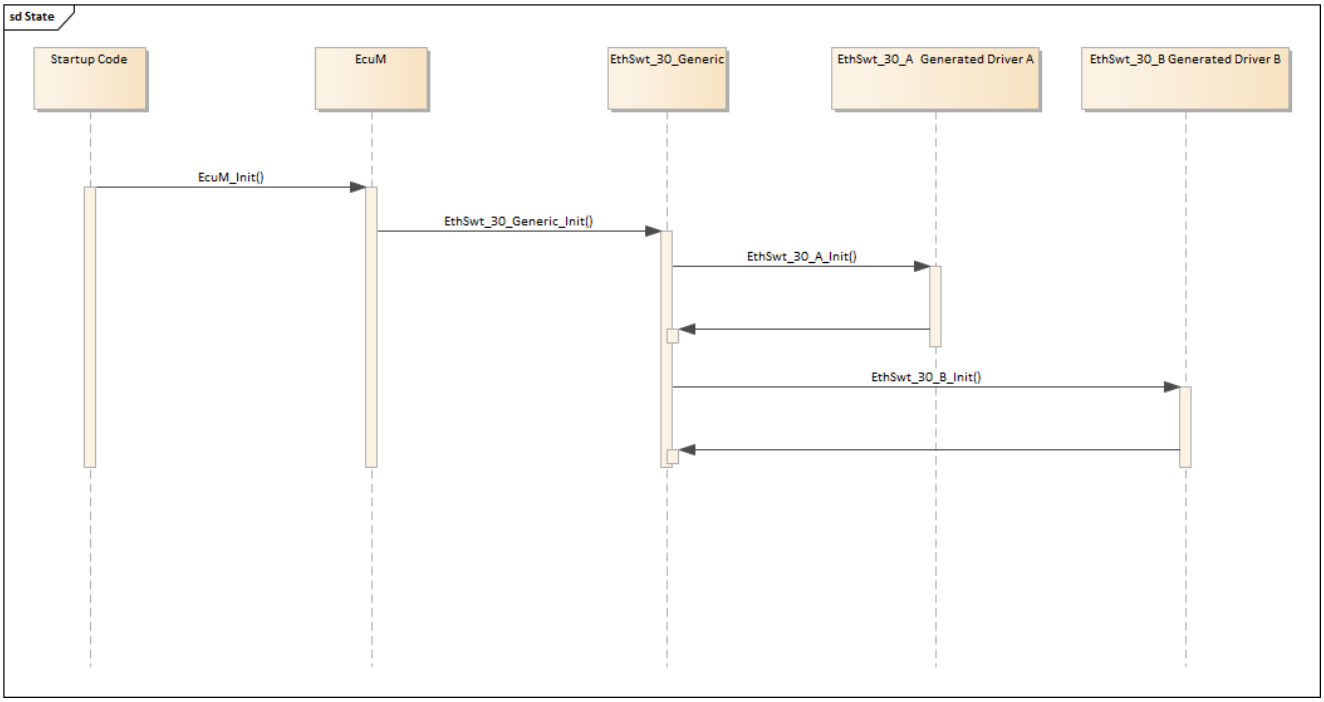


Figure 3-3 Init sequence

3.4.2 Switch Initialization

The functions *EthSwt_<VendorId>_<DriverName>_SwitchInit()* and *EthSwt_<VendorID>_<DriverName>_VSwitchInit()* are used to synchronously or asynchronously download the configuration to the Ethernet switch device.

3.4.2.1 Synchronous downloading method

The Synchronous method is defined to download the configuration directly when EthSwt_<VendorId>_<DriverName>_SwitchInit() is called.

The speed of the serial communication protocol used and the amount of data to load will consume runtime and can harm an eventual timeout process. In this situation the function can’t be called in the EcuM module during the initialization process.

Note
There are no restrictions and/or special configuration needed to use the synchronous downloading method.

However, the users/integrators must be aware of the situation described above (runtime consummation).

EthSwt_<VendorID>_<DriverName>_VSwitchInit() can be left empty.

3.4.2.2 Asynchronous downloading method

An asynchronous method for downloading the configuration can be implemented in order to reduce the runtime consummation.

The function EthSwt_<VendorID>_<DriverName>_SwitchInit() only checks the result of the download meanwhile The background function called EthSwt_<VendorID>_<DriverName>_VSwitchInit() perform the downloading steps.

The function EthSwt_<VendorID>_<DriverName>_SwitchInit() only check if a global buffer containing the “downloading state” of the Ethernet Switch Device is finished. The function returns E_OK if finished, and E_NOT_OK otherwise.

The function EthSwt_<VendorID>_<DriverName>_VSwitchInit() is called in a Background Task or idle (Task with lower priority) downloads the configuration.

The background Task sets the “downloading state” in the global buffer related to the Ethernet Switch Device to “finished” when the data is successfully downloaded to the device. The background task will then continue with the next one.

Caution
Be aware of concurrency task which downloads the configuration to the same Ethernet Switch device.

Asynchronous example

```
/* EthSwt_30_Generic.c */

Std_ReturnType EthSwt_<VendorID>_<DriverName _SwitchInit(
uint8 SwitchIdx)
{
  Std_ReturnType RetVal;

   /* Check if the configuration has been successfully downloaded */
  If ( InitDone[SwitchIdx] == TRUE ){
    RetVal = E_OK;
  }
  else {
  RetVal = E_NOT_OK;
  }
  return RetVal;
  }
```

Asynchronous example

```
#include "EthSwt_30_Generic.h"
/* Task definition */
TASK(BackgroundTask)
{
  /* The BackgroundTask task is configured to be the idle task.
   * The task is preemptable and has the lowest priority.*/

  boolean     InitDone[ETHSWT_SWITCH_NUM];
  uint8_least SwtIdx;

  /* initialize local state variable */
  for( SwtIdx = 0; SwtIdx < ETHSWT_SWITCH_NUM; SwtIdx++ )
  {
    InitDone[SwtIdx] = FALSE;
  }

  /* Task endless loop */
  while( 1 )
  {
    for( SwtIdx = 0; SwtIdx < ETHSWT_SWITCH_NUM; SwtIdx++ )
    {
      /* check if initialization was already performed */
      if( InitDone[SwtIdx] == FALSE )
      {
        /* trigger initialization */
        if( EthSwt_<VendorID>_<DriverName>_VSwitchInit((uint8)SwtIdx) == E_OK )
        {
          /* initialization successfully finished */
          InitDone[SwtIdx] = TRUE;
        }
        else
        {
          /* error handling */
        }
      }
    }
  }
}
```

3.5 States

There is no state machine or Driver state implemented in the *Generic Ethernet Switch driver*.

3.6 Main Functions

The main functions or *Generic Ethernet Switch Driver* Main Task is a periodical function called by the Scheduler manager. The main function which allow the users/integrators to perform periodic processing. One of the common use cased is to report DEM events.

Note
Ensure that the scheduler pre-emption mechanisms do not affect the driver functionality during the implementation of the generic Ethernet Switch main function.

3.6.1 Usage

The *Generic Ethernet Switch Driver* provides a static function EthSwt_30_Generic_Mainfunction() used as an interface to call all *Generic Driver* Main functions.

The scheduler will always call the static main function, an example with two generic drivers (A & B) is shown below.

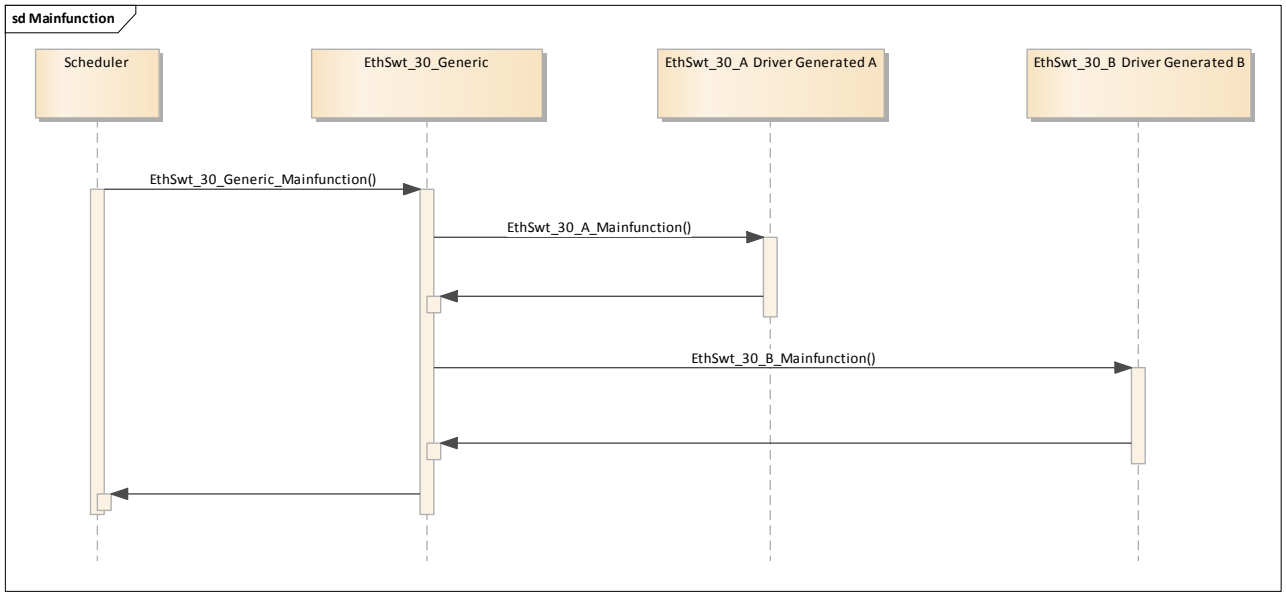


Figure 3-4 Main function Sequence

3.6.2 Configurations

The scheduled period is configured by the Users/Integrators using the EthSwt_MainFunction() Period parameter in the DaVinci configurator.

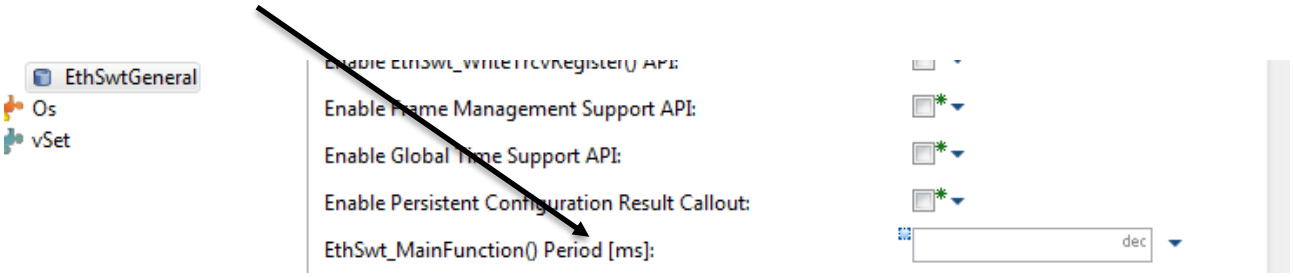


Figure 3-5 Main function time configuration

The static main function (EthSwt_30_MainFunction) is automatically registered to the scheduler module. No configuration from the users/integrators is required.

Caution
The configured scheduled period is the same for all *Generic Ethernet Switch Driver* included in the static main function.
It is not possible to define different scheduled period values for each main function.

3.7 Error Handling

3.7.1 Development Error Reporting

Development Error Reporting is not supported (refer to Feature list table).

The DET module is not included in the *Generic Ethernet Switch Driver*.

3.7.2 Production Code Error Reporting

Production Error code shall not be used for mirroring purpose. These features shall only be implemented in the EthSwt_30_Generic.c.

3.7.2.1 Definition

The production code related error are reported to the DEM by default using the service `Dem_ReportErrorStatus()`

If another module is used for production code error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Dem_ReportErrorStatus()`.

Production error reporting is enabled as Pre-compile parameters (ETHSWT_<VendorID>_<DriverName>_ DEV_ERROR_REPORT == STD_ON) when one of DEM event parameters has been defined and assign to a *Generic Ethernet Switch Driver*.

Note
The function `Dem_ReportErrorStatus()` is not implemented in the Driver.

The users/Integrators shall call this function only when DEM events parameters are configured

The include directive to the DEM module (#include “Dem.h”) is generated.

The errors reported to DEM are described in the following table:

Error Code	Description
E_ACCESS	Access to switch hardware failed.
E_BUFFEROVERRUN	Buffer overrun detected.
E_CRC	CRC error detected.
E_UNDERSIZEPCKT	Undersized frame detected.
E_OVERSIZEPCKT	Oversized frame detected.
E_ALIGNMENT	Alignment error detected.
E_SQETEST SQE	test error detected.
E_INDISCARD	Non- erroneous frame discarded on ingress side.
E_INERROR	Erroneous frame on ingress side detected.
E_OUTDISCARD	Non- erroneous frame discarded on egress Side.
E_OUTERROR	Erroneous frame on egress side detected.
E_SINGLECOLLISION	Single collision detected.
E_MULTIPLECOLLISION	Multiple collisions detected
E_DEFFEREDTRANSMISSION	Deferred transmission detected.
E_LATECOLLISION	Late collision detected.

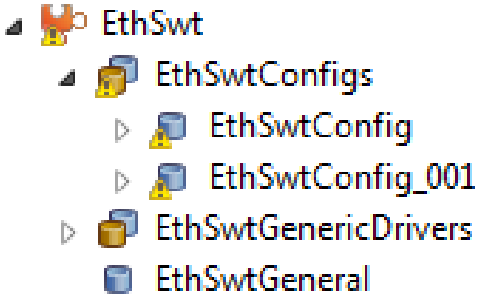
Table 3-8 Errors reported to DEM

3.7.2.2 Usage

The users/Integrators can access the DEM events parameters configured via the macro:

EthSwt_30_Generic_GetDemEventsOf_<Name_of_the_EthernetSwitchConfig>(<DemEventsParameters>)

<Name_Of_the_EthernetSwitchConfig> represent the *Short Name* of the Ethernet Switch config container in the example below



EthSwtConfig/EthSwtConfig_001 represent the *Short name* of the Ethernet Switch configuration.

the following two macros will be generated assuming that EthSwtConfig/ EthSwtConfig_001 contain DEM event parameters:

EthSwt_30_Generic_GetDemEventsOf_EthSwtConfig(DemEventsParameters)

EthSwt_30_Generic_GetDemEventsOf_EthSwtConfig_001(DemEventsParameters)

Note
These macros are located in EthSwt_30_Generic.c

<DemEventsParameters> represent an enumeration of the DEM events listed in the table 3-9.

The DEM events enumeration called EthSwt_30_Generic_DemEventsDefinitionType is defined in the file EthSwt_30_Generic_Types.h (generated file).

DEM events	Associate enumeration
E_ACCESS	ETHSWT_30_GENERIC_E_ACCESS
E_BUFFEROVERRUN	ETHSWT_30_GENERIC_E_BUFFEROVERRUN
E_CRC	ETHSWT_30_GENERIC_E_CRC
E_UNDERSIZEPCKT	ETHSWT_30_GENERIC_E_UNDERSIZEPCKT
E_OVERSIZEPCKT	ETHSWT_30_GENERIC_E_OVERSIZEPCKT
E_ALIGNMENT	ETHSWT_30_GENERIC_E_ALIGNMENT

E_SQETEST SQE	ETHSWT_30_GENERIC_E_SQETEST
E_INDISCARD	ETHSWT_30_GENERIC_E_INDISCARD
E_INERROR	ETHSWT_30_GENERIC_E_INERROR
E_OUTDISCARD	ETHSWT_30_GENERIC_E_OUTDISCARD
E_OUTERROR	ETHSWT_30_GENERIC_E_OUTERROR
E_SINGLECOLLISION	ETHSWT_30_GENERIC_E_SINGLECOLLISION
E_MULTIPLECOLLISION	ETHSWT_30_GENERIC_E_MULTIPLECOLLISION
E_DEFFEREDTRANSMISSION	ETHSWT_30_GENERIC_E_DEFFEREDTRANSMISSION
E_LATECOLLISION	ETHSWT_30_GENERIC_E_LATECOLLISION
***No parameters*	ETHSWT_30_GENERIC_E_NO_PARAMETER

Table 3-9 associate DEM events enumeration

** When no DEM event parameter is configured (for a configuration which contain at list one DEM parameter available) the previous macro xx_*GetDemEventsOf_xx(..)* will return ETHSWT_30_GENERIC_E_NO_PARAMETER.

For each DEM events an enumeration is available.

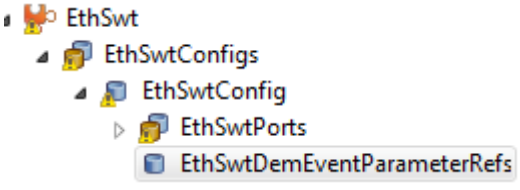
Caution
The users/integrators shall only use the elements defined in the enumeration to access to the DEM events parameters configured.

Example

The example below shown how the users/integrator can use/access to the Dem Events parameters configured by using the methodology defined above.

Definition of an Ethernet Switch configuration container called *EthswtConfig*.

Definition of a DEM events parameters container called *EthSwtDemEventParameterRefs*



The container EthSwtDemEventParameterRefs is configure with the following attributes

Short Name:	EthSwtDemEventParameterRefs
E_ACCESS:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0
E_ALIGNMENT:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_001
E_BUFFEROVERRUN:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_003
E_CRC:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_004
E_DEFFEREDTRANSMISSION:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_005
E_INDISCARD:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_007
E_INERROR:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_006
E_LATECOLLISION:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_008
E_MULTIPLECOLLISION:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_009
E_OUTDISCARD:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_010
E_OUTERROR:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_011
E_OVERSIZEPCKT:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_012
E_SINGLECOLLISION:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_013
E_SQETEST:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_014
E_UNDERSIZEPCKT:	/ActiveEcuC/Dem/DemConfigSet/DemEventParameter0_002

After generation the macro *EthSwt_30_Generic_GetDemEventsOf_EthSwtConfig( )* is available in the file EthSwt_30_Generic.c.

By calling:

EthSwt_30_Generic_GetDemEventsOf_EthSwtConfig

(*ETHSWT_30_GENERIC_E_ACCESS*)

The macro above will return the symbolic name:

DemConf_DemEventParameter_DemEventParameter0

4 Integration

This chapter gives necessary information for the integration of the MICROSAR EthSwt into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the EthSwt contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

The component has not static files (see Global functional description)

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool [config tool].

File Name	Description
EthSwt_30_Generic.c	Contains the generated driver API, which must be extended by an implementation.
EthSwt_30_Generic_Mirror.c	Contains the generated driver API for mirroring purpose, which can be extended by an implementation if Mirror is enabled
EthSwt_30_Generic.h	Contains the generated driver API prototypes.
EthSwt_30_Generic_Mirror.h	Contains the generated driver API prototypes for mirroring.
EthSwt_30_Generic_Types.h	Generated file providing the driver specific data types.
EthSwt_30_Generic_MemMap.h	Contains the generated memory map section, which shall be used within the implementation and can be modified/extended. (file must be included into the common MemMap.h)
EthSwt_30_Generic_Compiler_Cfg.h	Contains the generated compiler abstraction defines, which shall be used within the implementation and can be modified/extended. (file must be included into the common Compiler_Cfg.h)
EthSwt_30_Generic_Cfg.h	Contains the configurations

Table 4-1 Generated files

Note
The path for all the generated files except *EthSwt_30_Generic_Cfg.h* is: {*MyDaVinciProjectDirectoy*}/Appl/Source

The path for *EthSwt_30_Generic_Cfg.h* is: {*MyDaVinciProjectDirectoy*}/Appl/GenData

4.2 Critical Sections

There is no critical section(s) defined and used in the *Generic Ethernet Switch Driver.*

4.3 Compiler Abstraction and Memory Mapping

The objects (e.g. variables, functions, constants) are declared by compiler independent definitions – the compiler abstraction definitions. Each compiler abstraction definition is assigned to a memory section.

Memory Mapping Sections / Compiler Abstraction Definitions	ETHSWT_CONST	ETHSWT_VAR	ETHSWT_CODE
ETHSWT_<VendorID>_<DriverName>_START_SEC_CODE			■
ETHSWT_30_GENERIC_START_SEC_CODE			■
ETHSWT_<VendorID>_<DriverName>_STOP_SEC_CODE			■
ETHSWT_30_GENERIC_STOP_SEC_CODE			■
ETHSWT_<VendorID>_<DriverName>_CONST	■		
ETHSWT_30_GENERIC_CONST	■		

Table 4-2 Compiler abstraction and Memory map

The Compiler Abstraction is generated in the file `EthSwt_30_Generic_Compiler_Cfg.h` for every Ethernet Switch Generic Driver. This file must be included in the `Compiler_Cfg.h`.

The Memory Mapping is generated in the file `EthSwt_30_Generic_MemMap.h` for every Ethernet Switch Generic Driver. This file must be included in the `MemMap.h`.

4.4 User Code Integration

a user code block is an area where the user/integrator can implement code

Example

```
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!           <USERBLOCK ETH_<VendorID>_<DriverName>_<Name>
 **********************************************************************************************/
<UserCode>
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!           </USERBLOCK>
 **********************************************************************************************/
```

The user block area is always delimited by 2 tags comment respectively one for open the user code block **<USERBLOCK ETHSWT_<VendorID>_<DriverName>_<AreaName>** and **</USERBLOCK>**

Caution

The UserBlock comments shall not be remove otherwise this customized code between the UserBlock comments will be deleted.

The UserBlock contains the VendorID and DriverName. If the VendorID and DriverName are changed in the configuration tool the name of the UserBlock will change as well and create a new UserBlock. The generator will handle the old UserBlock as an unrecognized UserBlock (located at the end of the file), shown in the following example:

Example

If the name of the UserBlock is changed, the generator will move this unrecognized UserBlock at the end of the file, shown below. A new UserBlock with the correct name will replace the unrecognized UserBlock.

```
#if 0
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!           <USERBLOCK ETHSWT_<VendorID>_<DriverName>_<Name>
 **********************************************************************************************/
<UserCode>
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!           </USERBLOCK>
 **********************************************************************************************/
#endif
```

Caution
The user/integrator code shall be implemented only in the User block code area predefined.

The table 4-3 list the user block name for each file

File	User Block section
EthSwt_30_Generic_Compiler.h	Compiler abstraction defines
EthSwt_30_Generic_MemMap.h	Compiler abstraction defines
EthSwt_30_Generic_Types.h	Include Global constant macros Global function macros Config structures Global data prototypes Global functions prototypes
EthSwt_30_Generic.h	Include Global constant macros Global function macros Global data types and structures
EthSwt_30_Generic_Mirror.h	Include Local constant macros Global function macros Global data types and structures Global data types prototypes
EthSwt_30_Generic.c	Include Local constant macros Local function macros Local data types and structures DEM events Global data Local functions Functions implementation
EthSwt_30_Generic_Mirror.c	Include Local constant macros Local function macros Local data types and structures Local data types prototypes Global data

	Functions implementation
EthSwt_30_Generic_Cfg.h	*No User Block*

Table 4-3 User block sections in files

4.4.1 Compiler Abstraction define

Additional macro definitions can be added in the UserBlock sections `ETHSWT_<VendorID>_<DriverName>_COMPILER_ABSTRACTION_DEFINES` shown in the following example.

Example
The macro definitions between the UserBlocks `ETHSWT_<VendorID>_<DriverName>_COMPILER_ABSTRACTION_DEFINES` will not be overwritten by the code-generator. The user can add an additional macro definition like `#define ETHSWT_<VendorID>_<DriverName>_<NEW_DEFINE>`.

```
/**********************************************************************************************************
 * DO NOT CHANGE THIS COMMENT
<USERBLOCK ETHSWT_<VendorID>_<DriverName>_COMPILER_ABSTRACTION_DEFINES>
 *********************************************************************************************************/
#define ETH_<VendorID>_<DriverName>_<NEW_DEFINE>
/**********************************************************************************************************
 * DO NOT CHANGE THIS COMMENT!    </USERBLOCK>
 *********************************************************************************************************/
```

4.4.2 Include

Additional header files can be added in the UserBlock sections `ETHSWT_30_GENERIC_INCLUDES` shown in the following example.

Example
The included `'<Header_File>.h'` entry will not be deleted by the generator due to the UserBlock sections `ETHSWT_30_GENERIC_INCLUDES`.

```
**********************************************************************************************************
 *  INCLUDES
 *********************************************************************************************************/
#include "EthSwt_30_Generic.h"
/**********************************************************************************************************
 * DO NOT CHANGE THIS COMMENT!          <USERBLOCK ETHSWT_30_GENERIC_INCLUDES>
 *********************************************************************************************************/
#include "<Header_File>.h"
/**********************************************************************************************************
 * DO NOT CHANGE THIS COMMENT!          </USERBLOCK>
 *********************************************************************************************************/
```

4.4.3 Local/Global Constant macros

Additional local, global constant macros can be added in the UserBlock sections ETHSWT_30_GENERIC_LOCAL_CONSTANT_MACROS and ETHSWT_30_GENERIC_GLOBAL_CONSTANT_MACROS

shown in the following example.

Example
The local constant macro <NEW_DEFINE> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_LOCAL_CONSTANT_MACROS.

```
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!         <USERBLOCK ETHSWT_30_GENERIC_LOCAL_CONSTANT_MACRO
S>
 *********************************************************************************************/
#define <NEW_DEFINE>        <CONST>
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!         </USERBLOCK>
 *********************************************************************************************/
```

Example
The local constant macro <NEW_DEFINE> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_GLOBAL_CONSTANT_MACROS.

```
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!         <USERBLOCK ETHSWT_30_GENERIC_GLOBAL_CONSTANT_MACR
OS>
 *********************************************************************************************/
#define <NEW_DEFINE>        <CONST>
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!         </USERBLOCK>
 *********************************************************************************************/
```

4.4.4 Local/Global Functions macros

Additional local, global function macros can be added in the UserBlock sections ETHSWT_30_GENERIC_LOCAL_FUNCTION_MACROS and

ETHSWT_30_GENERIC_GLOBAL_FUNCTION_MACROS

shown in the following example.

Example
The function macro <NEW_DEFINE> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_LOCAL_FUNCTION_MACROS.

```
/**********************************************************************************
 * DO NOT CHANGE THIS COMMENT!        <USERBLOCK ETHSWT_30_GENERIC_LOCAL_FUNCTION_MACROS
>
 **********************************************************************************/
#define <NEW_DEFINE>        <FUNCTION>
/**********************************************************************************
 * DO NOT CHANGE THIS COMMENT!         </USERBLOCK>
 **********************************************************************************/
```

Example
The function macro <NEW_DEFINE> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_GLOBAL_FUNCTION_MACROS.

```
/**********************************************************************************
 * DO NOT CHANGE THIS COMMENT!        <USERBLOCK ETHSWT_30_GENERIC_GLOBAL_FUNCTION_MACRO
S>
 **********************************************************************************/
#define <NEW_DEFINE>        <FUNCTION>
/**********************************************************************************
 * DO NOT CHANGE THIS COMMENT!         </USERBLOCK>
 **********************************************************************************/
```

4.4.5 Local/Global Data types and structures

Additional types and structures can be added in the UserBlock sections ETHSWT_30_GENERIC_LOCAL_DATA_TYPES_AND_STRUCTURES and

ETHSWT_30_GENERIC_GLOBAL_DATA_TYPES_AND_STRUCTURES

shown in the following example.

Example
The structure <StructName> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_LOCAL_DATA_TYPES_AND_STRUCTURES.

```
/**************************************************************************************************
 *  LOCAL DATA TYPES AND STRUCTURES
 **************************************************************************************************/
#ifndef   STATIC
#  define STATIC static
#endif /* STATIC */
/**********************************************************************************
 * DO NOT CHANGE THIS COMMENT!      <USERBLOCK ETHSWT 30 GENERIC LOCAL DATA TYPES AND STRUCTUR
ES>
 **********************************************************************************/
struct <StructName> {
  VAR(uint8, ETHSWT_<VendorId>_<DriverName>_VAR_NOINIT) <VarName1>;
  VAR(uint8, ETHSWT_<VendorId>_<DriverName>_VAR_NOINIT) <VarName2>;
};
/**********************************************************************************
 * DO NOT CHANGE THIS COMMENT!       </USERBLOCK>
 **********************************************************************************/
```

Example
The structure <StructName> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_GLOBAL_DATA_TYPES_AND_STRUCTURES.

```
/**********************************************************************************************
 *  LOCAL DATA TYPES AND STRUCTURES
 **********************************************************************************************/
#ifndef   STATIC
#  define STATIC static
#endif /* STATIC */
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!      <USERBLOCK ETHSWT_30_GENERIC_GLOBAL_DATA_TYPES_AND_STRUCTU
RES>
 **********************************************************************************************/
struct <StructName> {
  VAR(uint8, ETHSWT_<VendorId>_<DriverName>_VAR_NOINIT) <VarName1>;
  VAR(uint8, ETHSWT_<VendorId>_<DriverName>_VAR_NOINIT) <VarName2>;
};
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!       </USERBLOCK>
 **********************************************************************************************/
```

4.4.6 Local/Global data prototypes

Additional prototypes can be added in the UserBlock sections ETHSWT_30_GENERIC_LOCAL_DATA_PROTOTYPES and

ETHSWT_30_GENERIC_GLOBAL_DATA_PROTOTYPES

shown in the following example.

Example
The data prototype <dataPrototypeName> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_LOCAL_DATA_PROTOTYPES.

```
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!      <USERBLOCK ETHSWT_30_GENERIC_LOCAL_DATA_PROTOTYPES>
 **********************************************************************************************/
#define ETHSWT_<VendorId>_<DriverName>_START_SEC_VAR_NOINIT_8BIT
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */

STATIC VAR(uint8, ETHSWT_<VendorId>_<DriverName>_VAR_NOINIT) <dataPrototypeName>;

#define ETHSWT_<VendorId>_<DriverName>_STOP_SEC_VAR_NOINIT_8BIT
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!       </USERBLOCK>
 **********************************************************************************************/
```

Example
The data prototype <dataPrototypeName> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_GLOBAM_DATA_PROTOTYPES.

```
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!      <USERBLOCK ETHSWT_30_GENERIC_GLOBAL_DATA_PROTOTYPES>
 *********************************************************************************************/
#define ETHSWT_<VendorId>_<DriverName>_START_SEC_VAR_NOINIT_8BIT
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */

STATIC VAR(uint8, ETHSWT_<VendorId>_<DriverName>_VAR_NOINIT) <dataPrototypeName>;

#define ETHSWT_<VendorId>_<DriverName>_STOP_SEC_VAR_NOINIT_8BIT
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!       </USERBLOCK>
 *********************************************************************************************/
```

4.4.7 Local/Global data

Additional data can be added in the UserBlock sections

ETHSWT_30_GENERIC_LOCAL_DATA and

ETHSWT_30_GENERIC_GLOBAL_DATA

shown in the following example.

Example
The variable <VarName> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_LOCAL_DATA.

```
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!     <USERBLOCK ETHSWT_30_GENERIC_LOCAL_DATA>
 *********************************************************************************************/
#define ETHSWT_<VendorId>_<DriverName>_START_SEC_VAR_NOINIT_8BIT
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */

VAR(uint8, ETHSWT_<VendorId>_<DriverName>_VAR_NOINIT) <VarName>;

#define ETHSWT_<VendorId>_<DriverName>_STOP_SEC_VAR_NOINIT_8BIT
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!       </USERBLOCK>
 *********************************************************************************************/
```

Example
The variable <VarName> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_GLOBAL_DATA.

```
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!     <USERBLOCK ETHSWT_30_GENERIC_GLOBAL_DATA>
 *********************************************************************************************/
#define ETHSWT_<VendorId>_<DriverName>_START_SEC_VAR_NOINIT_8BIT
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */

VAR(uint8, ETHSWT_<VendorId>_<DriverName>_VAR_NOINIT) <VarName>;

#define ETHSWT_<VendorId>_<DriverName>_STOP_SEC_VAR_NOINIT_8BIT
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!       </USERBLOCK>
 *********************************************************************************************/
```

4.4.8 Local/Global data prototypes

Additional data prototypes can be added in the UserBlock sections ETHSWT_30_GENERIC_LOCAL_DATA_PROTOTYPES and

ETHSWT_30_GENERIC_LOCAL_DATA_PROTOTYPES

shown in the following example.

Example
The data prototype <dataPrototypeName> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_LOCAL_DATA_PROTOTYPES.

```
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!      <USERBLOCK ETHSWT_30_GENERIC_LOCAL_DATA_PROTOTYPES>
 *********************************************************************************************/
#define ETHSWT_<VendorId>_<DriverName>_START_SEC_VAR_NOINIT_8BIT
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */

STATIC VAR(uint8, ETHSWT_<VendorId>_<DriverName>_VAR_NOINIT) <dataPrototypeName>;

#define ETHSWT_<VendorId>_<DriverName>_STOP_SEC_VAR_NOINIT_8BIT
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!       </USERBLOCK>
 *********************************************************************************************/
```

Example
The data prototype <dataPrototypeName> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_GLOBAL_DATA_PROTOTYPES.

```
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!      <USERBLOCK ETHSWT_30_GENERIC_GLOBAL_DATA_PROTOTYPES>
 *********************************************************************************************/
#define ETHSWT_<VendorId>_<DriverName>_START_SEC_VAR_NOINIT_8BIT
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */

STATIC VAR(uint8, ETHSWT_<VendorId>_<DriverName>_VAR_NOINIT) <dataPrototypeName>;

#define ETHSWT_<VendorId>_<DriverName>_STOP_SEC_VAR_NOINIT_8BIT
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!       </USERBLOCK>
 *********************************************************************************************/
```

4.4.9 Local/Global functions prototypes

Additional function prototypes can be added in the UserBlock sections ETHSWT_30_GENERIC_FUNCTION_PROTOTYPES shown in the following example.

Example
The function < functionName> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_FUNCTION_PROTOTYPES.

```
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!     <USERBLOCK ETHSWT_30_GENERIC_FUNCTION_PROTOTYPES>
 *********************************************************************************************/
#define ETHSWT_<VendorId>_<DriverName>_START_SEC_CODE
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */

FUNC(Std_ReturnType, ETHSWT_<VendorId>_<DriverName>_CODE) <functionName>( void );

#define ETHSWT_<VendorId>_<DriverName>_STOP_SEC_CODE
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */
/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!       </USERBLOCK>
 *********************************************************************************************/
```

4.4.10 Configuration structure

Addition structure elements can be added in the UserBlock section ETHSWT_30_GENERIC_CONFIG_STRUCT shown in the following example.

Example
The structure element < StructMember> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_CONFIG_STRUCT.

```
Typedef struct EthSwt_30_Generic_ConfigStruct{

/**************************************************************************************************
 * DO NOT CHANGE THIS COMMENT!     <USERBLOCK ETHSWT_30_GENERIC_CONFIG_STRUCT>
 *************************************************************************************************/

Uint8 DummyElement,
Uint32 <StructMember>

/**************************************************************************************************
 * DO NOT CHANGE THIS COMMENT!       </USERBLOCK>
 *************************************************************************************************/

} EthSwt_30_Generic_ConfigType;
```

4.4.11 Local functions

Additional local function can be added in the UserBlock sections ETHSWT_30_GENERIC_LOCAL_FUNCTION shown in the following example.

Example
The function < functionName> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_LOCAL_FUNCTION.

```
/**************************************************************************************************
 * DO NOT CHANGE THIS COMMENT!     <USERBLOCK ETHSWT_30_GENERIC_LOCAL_FUNCTION >
 *************************************************************************************************/
#define ETHSWT_<VendorId>_<DriverName>_START_SEC_CODE
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */

FUNC(void, ETHSWT_<VendorId>_<DriverName>_CODE) <functionName>( void ){

(…)

}

#define ETHSWT_<VendorId>_<DriverName>_STOP_SEC_CODE
#include "MemMap.h" /* PRQA S 5087 */ /* MD_MSR_19.1 */
/**************************************************************************************************
 * DO NOT CHANGE THIS COMMENT!       </USERBLOCK>
 *************************************************************************************************/
```

4.4.12 Functions implementation

The function ETHSWT_<VendorID>_<DriverName>_<FunctionName> has one UserBlock section which shall contains the local data declaration, local data definition and the implementation.

Example
The local data declaration of the variable <VarName> will not be deleted by the generator due to the UserBlock sections ETH_<VendorID>_<DriverName>_<FunctionName>

```
FUNC(Std_ReturnType, ETHSSWT_<VendorID>_<DriverName>_CODE) EthSWT_<VendorID>_<DriverName>_
<FunctionName>(void)
{
  /* ----- Local data declaration ------------------------------------- */
  Std_ReturnType RetVal;

/**********************************************************************************************
 * DO NOT CHANGE…  <USERBLOCK EthSwt_<VendorID>_<DriverName>_<FunctionName>_<FunctionName>
 *********************************************************************************************/

  Uint8 myVariable;

  myVariable = 0x2;

  (..)

  RetVal = E_OK;


/**********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!           </USERBLOCK>
 *********************************************************************************************/

  return RetVal;
}
```

4.4.13 DEM events

Additional DEM events utility can be added in the UserBlock sections ETHSWT_30_GENERIC_DEM_EVENTS shown in the following example.

Example

The structure element <StructDemSwtMember> will not be deleted by the generator due to the UserBlock sections ETHSWT_30_GENERIC_DEM_EVENTS.

```
/*********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!     <USERBLOCK ETHSWT_30_GENERIC_DEM_EVENTS>
 *********************************************************************************************/


Typedef struct DemEventsMyUserDefinition{


Uint8 Swtindex,
(…)

} EthSwt_30_Generic_ConfigType;

/*********************************************************************************************
 * DO NOT CHANGE THIS COMMENT!       </USERBLOCK>
 *********************************************************************************************/
```

5 API Description

5.1 Type Definitions

EthSwt_30_Generic_ConfigType

This structure contains/holds the element necessary for the configuration of the Ethernet Switch driver device.

Note
The users/integrators have to implement his own data structure

Struct Element Name	C-Type	Description	Value Range
Dummy	Uint8	Dummy value shown as example	0x00
			0xFF

Table 5-1 EthSwt_30_Generic_ConfigType

5.2 API Table

5.2.1 EthSwt_<VendorID>_<DriverName>_InitMemory

Prototype	
`void` **`EthSwt_<VendorID>_<DriverName>_InitMemory`** `(void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Function for *_INIT_*-variable initialization. Service to initialize module global variables at power up. This function initializes the variables in *_INIT_* sections. Used in case they are not initialized by the startup code.	
Particularities and Limitations	
Module is uninitialized.	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-2 EthSwt_<VendorID>_<DriverName>_InitMemory

5.2.2 EthSwt_<VendorID>_<DriverName>_Init

Prototype	
`void` **`EthSwt_<VendorID>_<DriverName>_Init`** `(const EthSwt_30_Generic_ConfigType *ConfigPtr)`	
Parameter	
ConfigPtr [in]	Configuration structure for initializing the module
Return code	
void	none
Functional Description	
Initialization function of the component. This function initializes the module EthSwt_<VendorID>_<DriverName>. It initializes all variables and sets the module state to initialized.	
Particularities and Limitations	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-3 EthSwt_<VendorID>_<DriverName>_Init

5.2.3 EthSwt_<VendorID>_<DriverName>_SwitchInit

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_SwitchInit`** `(uint8 SwitchIdx)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch
Return code	
Std_ReturnType	E_NOT_OK - Initialization failed due to e.g. being unable to access hardware
Std_ReturnType	E_OK - Ethernet switch initialized
Functional Description	
Initializes an Ethernet switch. This function initializes the Ethernet switch hardware by downloading the static configuration	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-4 EthSwt_<VendorID>_<DriverName>_SwitchInit

5.2.4 EthSwt_<VendorID>_<DriverName>_VSwitchInit

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_VSwitchInit`** `(uint8 SwitchIdx)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch
Return code	
Std_ReturnType	E_NOT_OK - Initialization failed due to e.g. being unable to access hardware
Std_ReturnType	E_OK - Ethernet switch initialized
Functional Description	
Initializes an Ethernet switch. This function initializes the Ethernet switch hardware by downloading the static configuration This function may cost runtime !	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-5 EthSwt_<VendorID>_<DriverName>_VSwitchInit

5.2.5 EthSwt_<VendorID>_<DriverName>_SetSwitchPortMode

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_SetSwitchPortMode`** `(uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_ModeType PortMode)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
PortMode [in]	Mode that shall be applied: ETHTRCV_MODE_DOWN - Turns the port off ETHTRCV_MODE_ACTIVE - Turns the port on
Return code	
Std_ReturnType	E_NOT_OK - Mode of port couldn't be altered
Std_ReturnType	E_OK - Mode of port was altered
Functional Description	
Alters the mode of an Ethernet switch port. This function changes the mode of an Ethernet switch port.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-6 EthSwt_<VendorID>_<DriverName>_SetSwitchPortMode

5.2.6 EthSwt_<VendorID>_<DriverName>_GetSwitchPortMode

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_GetSwitchPortMode`** `(uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_ModeType *PortModePtr)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
PortModePtr [out]	Mode currently applied to the port: ETHTRCV_MODE_DOWN - Port turned off ETHTRCV_MODE_ACTIVE - Port turned on
Return code	
Std_ReturnType	E_NOT_OK - Mode of port couldn't be retrieved
Std_ReturnType	E_OK - Mode of port retrieved
Functional Description	
Retrieves the current mode of an Ethernet switch port. This function retrieves the current mode of an Ethernet switch port.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-7 EthSwt_<VendorID>_<DriverName>_GetSwitchPortMode

5.2.7 EthSwt_<VendorID>_<DriverName>_StartSwitchPortAutoNegotiation

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_StartSwitchPortAutoNegotiation`** `(uint8 SwitchIdx, uint8 SwitchPortIdx)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
Return code	
Std_ReturnType	E_NOT_OK - Auto negotiation process couldn't be triggered
Std_ReturnType	E_OK - Auto negotiation process triggered
Functional Description	
Triggers the autonegotiation process of an Ethernet switch port. This function triggers the auto negotiation process. This will result in losing the link on the port and subsequently acquiring it again after the two counter parts have re-negotiate the link parameters.	
Particularities and Limitations	
Ethernet switch port is in mode ETHTRCV_MODE_ACTIVE	
Call context	
> TASK > This function is Non-Reentrant	

Table 5-8 EthSwt_<VendorID>_<DriverName>_StartSwitchPortAutoNegotiation

5.2.8 EthSwt_<VendorID>_<DriverName>_GetLinkState

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_GetLinkState`** `(uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_LinkStateType *LinkStatePtr)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
LinkStatePtr [out]	Link state of the port: ETHTRCV_LINK_STATE_DOWN - Port hasn't acquired a link ETHTRCV_LINK_STATE_ACTIVE - Port acquired link
Return code	
Std_ReturnType	E_NOT_OK - Link state of port couldn't be retrieved
Std_ReturnType	E_OK - Link state of port retrieved
Functional Description	
Retrieves the current link state of an Ethernet switch port. This function retrieves the current link state of an Ethernet switch port.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-9 EthSwt_<VendorID>_<DriverName>_GetLinkState

5.2.9 EthSwt_<VendorID>_<DriverName>_GetBaudRate

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_GetBaudRate`** `(uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_BaudRateType *BaudRatePtr)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
BaudRatePtr [out]	Baud rate of the port: ETHTRCV_BAUD_RATE_10MBIT - 10 MBit/s connection ETHTRCV_BAUD_RATE_100MBIT - 100 MBit/s connection ETHTRCV_BAUD_RATE_1000MBIT - 1000 MBit/s connection
Return code	
Std_ReturnType	E_NOT_OK - Baud rate of port couldn't be retrieved
Std_ReturnType	E_OK - Baud rate of port retrieved
Functional Description	
Retrieves the current baud rate of an Ethernet switch port. This function retrieves the current baud rate of an Ethernet switch port.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-10 EthSwt_<VendorID>_<DriverName>_GetBaudRate

5.2.10 EthSwt_<VendorID>_<DriverName>_GetDuplexMode

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_GetDuplexMode`** `(uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_DuplexModeType *DuplexModePtr)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
DuplexModePtr [out]	Duplex mode of the port: ETHTRCV_DUPLEX_MODE_HALF - Half duplex connection ETHTRCV_DUPLEX_MODE_FULL - Full duplex connection
Return code	
Std_ReturnType	E_NOT_OK - Duplex mode of port couldn't be retrieved
Std_ReturnType	E_OK - Duplex mode of port retrieved
Functional Description	
Retrieves the current duplex mode of an Ethernet switch port. This function retrieves the current duplex mode of an Ethernet switch port.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-11 EthSwt_<VendorID>_<DriverName>_GetDuplexMode

5.2.11 EthSwt_<VendorID>_<DriverName>_GetPortMacAddr

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_GetPortMacAddr`** `(const uint8 *MacAddrPtr, const uint8 *SwitchIdxPtr, uint8 *PortIdxPtr)`	
Parameter	
MacAddrPtr [in]	MAC-address of the node the Ethernet switch port shall be retrieved for (unicast)
SwitchPortIdxPtr [out]	Identifier of the Ethernet switch
DuplexModePtr [out]	Identifier of the Ethernet switch port
Return code	
Std_ReturnType	E_NOT_OK - Information couldn't be retrieved (MAC-address not known)
Std_ReturnType	E_OK - Information retrieved
Functional Description	
Retrieves the port and the corresponding switch a node is connected to. This function retrieves the Ethernet switch port and the corresponding Ethernet switch a node with the given MAC-address is connected to. In a cascaded setup it also respects uplinks so the actual port the node is connected to is retrieved.	
Particularities and Limitations	
Node must have sent frames with the respective MAC-address to have a valid entry in the address resolution table Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_GET_PORT_MAC_ADDR_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-12 EthSwt_<VendorID>_<DriverName>_GetPortMacAddr

5.2.12 EthSwt_<VendorID>_<DriverName>_GetArlTable

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_GetArlTable`** `(uint8 SwitchIdx, uint32 *LenPtr, EthSwt_MacVlanType *ArlTablePtr)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch
LenPtr [in,out]	Size of the buffer: [in] - maximum entries the provided buffer is able to hold [out] - entries written to the buffer
ArlTablePtr [out]	Buffer to hold the retrieved entries of the address resolution table
Return code	
Std_ReturnType	E_NOT_OK - Address resolution table couldn't be retrieved
Std_ReturnType	E_OK - Address resolution table retrieved
Functional Description	
Retrieves the address resolution table of the Ethernet switch. This function retrieves the valid entries of the address resolution table for the given Ethernet switch.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_GET_ARL_TABLE_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-13 EthSwt_<VendorID>_<DriverName>_GetArlTable

5.2.13 EthSwt_<VendorID>_<DriverName>_GetBufferLevel

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_GetBufferLevel`** `(uint8 SwitchIdx, EthSwt_BufferLevelType *BufferLevelPtr)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch
BufferLevelPtr [out]	Buffer level retrieved
Return code	
Std_ReturnType	E_NOT_OK - Buffer level couldn't be retrieved
Std_ReturnType	E_OK - Buffer level retrieved
Functional Description	
Retrieves the current buffer occupation of an Ethernet switch. This function retrieves current level of buffer occupation of an Ethernet switch. Please refere to the respective technical reference of the used EthSwt driver for information about how to interpret the value returned.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_GET_BUFFER_LEVEL_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-14 EthSwt_<VendorID>_<DriverName>_GetBufferLevel

5.2.14 EthSwt_<VendorID>_<DriverName>_GetDropCount

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_GetDropCount`** `(uint8 SwitchIdx, uint16 *LenPtr, uint32 *DropCountPtr)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch
LenPtr [in,out]	Size of the buffer: [in] - maximum entries the provided buffer is able to hold [out] - entries written to the buffer
DropCountPtr [out]	Buffer to hold the retrieved drop counts
Return code	
Std_ReturnType	E_NOT_OK - Drop counts couldn't be retrieved
Std_ReturnType	E_OK - Drop counts retrieved
Functional Description	
Retrieves the drop counts of the Ethernet switch. This function retrieves the current drop counts for the given Ethernet switch. Please refer to the respective technical reference of the used EthSwt driver for information about which drop counts are available.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_GET_DROP_COUNT_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-15 EthSwt_<VendorID>_<DriverName>_GetDropCount

5.2.15 EthSwt_<VendorID>_<DriverName>_GetEtherStats

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_GetEtherStats`** `(uint8 SwitchIdx, uint8 SwitchPortIdx, uint32 *etherStats)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
etherStats [out]	Ethernet statistics accroding to AUTOSAR
Return code	
Std_ReturnType	E_NOT_OK - Ethernet statistics couldn't be retrieved
Std_ReturnType	E_OK - Ethernet statistics retrieved
Functional Description	
Retrieves the Ethernet statistics of an Ethernet switch port. This function retrieves the current Ethernet statistics of an Ethernet switch port. Please refer to the respective technical reference of the used EthSwt driver for information about which statistics are available.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_GET_ETHER_STATS_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-16 EthSwt_<VendorID>_<DriverName>_GetEtherStats

5.2.16 EthSwt_<VendorID>_<DriverName>_GetSwitchReg

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_GetSwitchReg`** `(uint8 SwitchIdx, uint32 page, uint32 registerAddr, uint32 *registerContent)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch
page [in]	Addressed memory page
registerAddr [in]	Addressed register
registerContent [out]	Content of the register read
Return code	
Std_ReturnType	E_NOT_OK - Register couldn't be retrieved
Std_ReturnType	E_OK - Register retrieved
Functional Description	
Retrieves the value of an Ethernet switch register. This function allows retrieving the value of an Ethernet switch register. Please note that this is API is highly hardware dependent and the addressing of registers may differ from hardware to hardware. Please refer to the technical reference of the respective EthSwt driver used on how the API must be used. CAUTION: In common use-case there shouldn't be the need for using this API.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_GET_SWITCH_REG_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-17 EthSwt_<VendorID>_<DriverName>_GetSwitchReg

5.2.17 EthSwt_<VendorID>_<DriverName>_SetSwitchReg

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_SetSwitchReg`** `(uint8 SwitchIdx, uint32 page, uint32 registerAddr, uint32 registerContent)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch
page [in]	Addressed memory page
registerAddr [in]	Addressed register
registerContent [in]	Value to be written
Return code	
Std_ReturnType	E_NOT_OK - Register couldn't be written
Std_ReturnType	E_OK - Register written
Functional Description	
Sets the value of an Ethernet switch register. This function allows setting the value of an Ethernet switch register. Please note that this is API is highly hardware dependent and the addressing of registers may differ from hardware to hardware. Please refer to the technical reference of the respective EthSwt driver used on how the API must be used. CAUTION: In common use-case there shouldn't be the need for using this API. If using it be aware that registers the driver relies on for proper operation could be changed, which results in unpredictable behavior!	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_SET_SWITCH_REG_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-18 EthSwt_<VendorID>_<DriverName>_SetSwitchReg

5.2.18 EthSwt_<VendorID>_<DriverName>_ReadTrcvRegister

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_ReadTrcvRegister`** `(uint8 SwitchIdx, uint8 SwitchPortIdx, uint8 RegIdx, uint16 *RegValPtr)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
RegIdx [in]	Identifier of the register
RegValPtr [out]	Retrieved register value
Return code	
Std_ReturnType	E_NOT_OK - Register couldn't be retrieved
Std_ReturnType	E_OK - Register retrieved
Functional Description	
Retrieves a register value of an Ethernet switch port. This function abstracts a read access to a register of an Ethernet switch port by utilizing the underlying hardware access mechanisms like MDIO, SPI etc.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_READ_TRCV_REG_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-19 EthSwt_<VendorID>_<DriverName>_ReadTrcvRegister

5.2.19 EthSwt_<VendorID>_<DriverName>_WriteTrcvRegister

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_WriteTrcvRegister`** `(uint8 SwitchIdx, uint8 SwitchPortIdx, uint8 RegIdx, uint16 RegVal)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
RegIdx [in]	Identifier of the register
RegVal [in]	Register value to be written
Return code	
Std_ReturnType	E_NOT_OK - Register couldn't be written
Std_ReturnType	E_OK - Register written
Functional Description	
Writes a register of an Ethernet switch port. This function abstracts a write access to a register of an Ethernet switch port by utilizing the underlying hardware access mechanisms like MDIO, SPI etc.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_WRITE_TRCV_REG_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-20 EthSwt_<VendorID>_<DriverName>_WriteTrcvRegister

5.2.20 EthSwt_<VendorID>_<DriverName>_EnableVlan

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_EnableVlan`** `(uint8 SwitchIdx, uint8 SwitchPortIdx, uint16 VlanId, boolean Enable)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
VlanId [in]	VID of the VLAN the forwarding shall be changed for
Enable [in]	Action that shall be applied: FALSE - VLAN shall be disabled TRUE - VLAN shall be enabled
Return code	
Std_ReturnType	E_NOT_OK - VLAN forwarding behavior couldn't be changed
Std_ReturnType	E_OK - VLAN forwarding behavior changed
Functional Description	
Enables/disables forwarding of the given VLAN on an Ethernet switch port. This function allows enable/disable the forwarding of the given VLAN on an Ethernet switch port.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_ENABLE_VLAN_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-21 EthSwt_<VendorID>_<DriverName>_EnableVlan

5.2.21 EthSwt_<VendorID>_<DriverName>_StoreConfiguration

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_StoreConfiguration`** `(uint8 SwitchIdx)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch
Return code	
Std_ReturnType	E_NOT_OK - Configuration couldn't be stored in NV-RAM
Std_ReturnType	E_OK - Configuration stored in NV-RAM
Functional Description	
Stores the currently valid entries of the address resolution table in the NV-RAM. This function stores the currently valid entries of the address resolution table in the NV-RAM, which in consequence is restored during the next Ethernet switch initialization so the switch instantly is able to redirect the Ethernet frames to the correct port without the need for learning them again.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_STORE_CONFIG_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-22 EthSwt_<VendorID>_<DriverName>_StoreConfiguration

5.2.22 EthSwt_<VendorID>_<DriverName>_ResetConfiguration

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_ResetConfiguration`** `(uint8 SwitchIdx)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch
Return code	
Std_ReturnType	E_NOT_OK - Configuration couldn't be invalidated in NV-RAM
Std_ReturnType	E_OK - Configuration invalidated in NV-RAM
Functional Description	
Invalidates the NV-RAM the valid entries of the address resolution table was stored in. This function invalidates the NV-RAM the valid entries of the address resolution table were stored in previously by a call to EthSwt_<VendorID>_<DriverName>_StoreConfiguration(). This in consequence disables the restore of the address resolution table during the next Ethernet switch initialization and will force the switch to learn the MAC-addresses again to redirect Ethernet frames without flooding them.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_RESET_CONFIG_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-23 EthSwt_<VendorID>_<DriverName>_ResetConfiguration

5.2.23 EthSwt_<VendorID>_<DriverName>_SetMacLearningMode

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_SetMacLearningMode`** `(uint8 SwitchIdx, uint8 SwitchPortIdx, EthSwt_MacLearningType MacLearningMode)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
MacLearningMode [in]	Mode that shall be applied: ETHSWT_MACLEARNING_HWDISABLED - Disables hardware learning of MAC-addresses on the port ETHSWT_MACLEARNING_HWENABLED - Enables hardware learning of MAC-addresses on the port ETHSWT_MACLEARNING_SWENABLED - not supported (software learning must involve the Host-CPU but AUTOSAR doesn't specify such functionality)
Return code	
Std_ReturnType	E_NOT_OK - Learning mode couldn't be modified
Std_ReturnType	E_OK - Learning mode modified
Functional Description	
Alters the MAC-address learning mode of an Ethernet switch port. This function alters the MAC-address learning mode of an Ethernet switch port.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_SET_MAC_LEARNING_MODE_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-24 EthSwt_<VendorID>_<DriverName>_SetMacLearningMode

5.2.24 EthSwt_<VendorID>_<DriverName>_GetMacLearningMode

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_GetMacLearningMode`** `(uint8 SwitchIdx, uint8 SwitchPortIdx, EthSwt_MacLearningType *MacLearningModePtr)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
MacLearningModePtr [out]	Mode that shall be applied: ETHSWT_MACLEARNING_HWDISABLED - Hardware learning of MAC-addresses on the port disabled ETHSWT_MACLEARNING_HWENABLED - Hardware learning of MAC-addresses on the port enabled
Return code	
Std_ReturnType	E_NOT_OK - Learning mode couldn't be retrieved
Std_ReturnType	E_OK - Learning mode retrieved
Functional Description	
Retrieves the MAC-address learning mode of an Ethernet switch port. This function retrieves the MAC-address learning mode of an Ethernet switch port.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_GET_MAC_LEARNING_MODE_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-25 EthSwt_<VendorID>_<DriverName>_GetMacLearningMode

5.2.25 EthSwt_<VendorID>_<DriverName>_GetVersionInfo

Prototype	
`void EthSwt_<VendorID>_<DriverName>_GetVersionInfo (Std_VersionInfoType *versioninfo)`	
Parameter	
versioninfo [out]	Pointer to where to store the version information. Parameter must not be NULL.
Return code	
void	none
Functional Description	
Returns the version information. EthSwt_<VendorID>_<DriverName>_GetVersionInfo() returns version information, vendor ID and AUTOSAR module ID of the component.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_VERSION_INFO_API	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 5-26 EthSwt_<VendorID>_<DriverName>_GetVersionInfo

5.2.26 EthSwt_<VendorID>_<DriverName>_MainFunction

Prototype	
`void` **`EthSwt_<VendorID>_<DriverName>_MainFunction`** `(void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Processes drop count retrieval, drop count monitoring. -	
Particularities and Limitations	
> --	
Call context	
> TASK > TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-27 EthSwt_<VendorID>_<DriverName>_MainFunction

5.2.27 EthSwt_<VendorID>_<DriverName>_UpdateMCastPortAssignment

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_UpdateMCastPortAssignment`** `(uint8 SwitchIdx, uint8 SwitchPortIdx, const uint8 *MCastAddr, EthSwt_MCastPortAssignActionType Action)`	
Parameter	
SwitchIdx [in]	Identifier of the Ethernet switch the port is related to
SwitchPortIdx [in]	Identifier of the Ethernet switch port
MCastAddr [in]	Pointer to the MAC Multicast Address
Action [in]	Action that shall be applied: ETHSWT_MCAST_PORT_ASSIGN_ACTION_ADD - Request passing of multicast on the port ETHSWT_MCAST_PORT_ASSIGN_ACTION_REMOVE - Request removal of multicast on the port
Return code	
Std_ReturnType	E_NOT_OK - Assignment could not be adapted
Std_ReturnType	E_OK - Assignment adapted successfully
Functional Description	
Modifes the MAC-Multicast to port assignment to allow/restrict communication to specific ports. This function allows modifying the forwarding of a MAC-multicast for the given port. Uplinks between switches are implicitly considered so no explicit action is needed for them. If the multicast isn't restricted to any specific port it will be flooded like common multicasts.	
Particularities and Limitations	
- Configuration Variant(s): ETHSWT_<VendorID>_<DriverName>_UPDATE_MCAST_PORT_ASSIGN_API	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-28 EthSwt_<VendorID>_<DriverName>_UpdateMCastPortAssignment

5.2.28 EthSwt_<VendorID>_<DriverName>_MirrorInit

Prototype	
`void` **`EthSwt_<VendorID>_<DriverName>_MirrorInit`** `(uint8 swtIdx)`	
Parameter	
swtIdx [in]	Switch identifier index
Return code	
void	none
Functional Description	
mode, Set the port mirroring to default and configure the switch	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-29 EthSwt_<VendorID>_<DriverName>_MirrorInit

5.2.29 EthSwt_<VendorID>_<DriverName>_WritePortMirrorConfiguration

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_WritePortMirrorConfiguration`** `(uint8 MirroredSwitchIdx, const EthSwt_PortMirrorCfgType *PortMirrorConfigurationPtr)`	
Parameter	
MirroredSwitchIdx [in]	Identifier of the switch which shall get the new configuration
PortMirrorConfigurationPtr [in]	Pointer to the new configuration
Return code	
Std_ReturnType	ETHSWT_PORT_MIRRORING_CONFIGURATION_NOT_SUPPORTED - Port mirroring configuration is invalid
	E_NOT_OK - Error writing configuration
	E_OK - Port mirroring configuration is valid and has been saved to buffer
Functional Description	
Write port mirroring configuration into internal buffer. This function validates the given port mirroring configuration and writes it into an internal buffer in the microcontroller. The stored configuration will be written into the switch later when the function EthSwt_<VendorID>_<DriverName>_SetPortMirrorState() is called to set the mirroring state to active.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-30 EthSwt_<VendorID>_<DriverName>_WritePortMirrorConfiguration

5.2.30 EthSwt_<VendorID>_<DriverName>_ReadPortMirrorConfiguration

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_ReadPortMirrorConfiguration`** `(uint8 MirroredSwitchIdx, EthSwt_PortMirrorCfgType *PortMirrorConfigurationPtr)`	
Parameter	
MirroredSwitchIdx [in]	Identifier of switch which configuration shall be read
PortMirrorConfigurationPtr [out]	Pointer to buffer where configuration shall be stored
Return code	
Std_ReturnType	E_NOT_OK - No port mirroring configuration has been found
Std_ReturnType	E_OK - Retrieval of port mirroring configuration has been successful
Functional Description	
Read port mirroring configuration from internal buffer. This function reads the current mirroring configuration of the specified switch from the internal buffer in the microcontroller.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-31 EthSwt_<VendorID>_<DriverName>_ReadPortMirrorConfiguration

5.2.31 EthSwt_<VendorID>_<DriverName>_GetPortMirrorState

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_GetPortMirrorState`** `(uint8 MirroredSwitchIdx, EthSwt_PortMirrorStateType *PortMirrorStatePtr)`	
Parameter	
MirroredSwitchIdx [in]	Identifier of switch which state shall be read
PortMirrorStatePtr [out]	Pointer to variable where state shall be stored
Return code	
Std_ReturnType	E_NOT_OK - Error getting port mirroring state
Std_ReturnType	E_OK - Retrieval of port mirroring state has been successful
Functional Description	
Get current port mirroring state. This function returns the current port mirroring state of the specified switch.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-32 EthSwt_<VendorID>_<DriverName>_GetPortMirrorState

5.2.32 EthSwt_<VendorID>_<DriverName>_SetPortMirrorState

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_SetPortMirrorState`** `(uint8 MirroredSwitchIdx, EthSwt_PortMirrorStateType PortMirrorState)`	
Parameter	
MirroredSwitchIdx [in]	Identifier of switch which state shall be written
PortMirrorState [in]	New port mirroring state
Return code	
Std_ReturnType	E_NOT_OK - Error setting port mirroring state
Std_ReturnType	E_OK - Setting of port mirroring state has been successful
Functional Description	
Set new port mirroring state. This function sets the new port mirroring state of the specified switch.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-33 EthSwt_<VendorID>_<DriverName>_SetPortMirrorState

5.2.33 EthSwt_<VendorID>_<DriverName>_DeletePortMirrorConfiguration

Prototype	
`Std_ReturnType` **`EthSwt_<VendorID>_<DriverName>_DeletePortMirrorConfiguration`** `(uint8 MirroredSwitchIdx)`	
Parameter	
MirroredSwitchIdx [in]	Identifier of switch which configuration shall be deleted
Return code	
Std_ReturnType	E_NOT_OK - Error deleting port mirroring configuration
Std_ReturnType	E_OK - Deleting of port mirroring configuration has been successful
Functional Description	
Delete stored mirroring configuration. This function deletes the port mirroring configuration of the specified switch. This is only only possible if the port mirroring state is PORT_MIRRORING_DISABLED.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-34 EthSwt_<VendorID>_<DriverName>_DeletePortMirrorConfiguration

5.3 Services used by EthSwt

In the following table services provided by other components, which are used by the EthSwt are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DEM	Dem_ReportErrorStatus() *Need to be implemented by the user*

Table 5-35 Services used by the EthSwt

5.4 Callback Functions

No callback functions are available in the Generic Ethernet Switch Driver.

6 Configuration

Create a new Generic Ethernet Switch driver by right-clicking on the EthSwtGenericDrivers and choose ‘Create EthSwtGenericDriver container’.

Name the Ethernet Switch driver with a unique driver name and Vendor ID,

as shown in Figure 6-1.

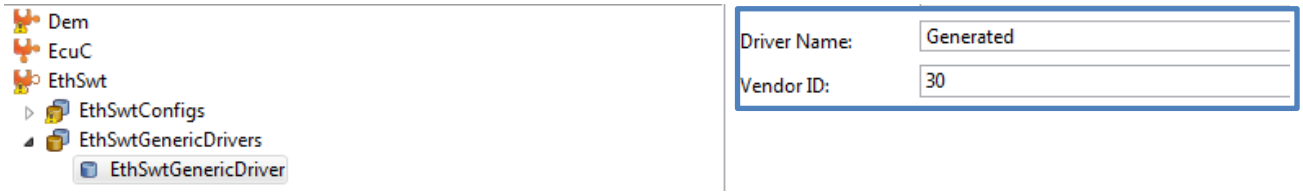


Figure 6-1 Ethernet Switch Driver name configuration

Caution
The Driver Name “Generic” and the Vendor ID “30” are not allowed.

The new created Ethernet Switch driver needs an Ethernet Switch configuration (EthSwtConfigs).

Create a new Ethernet controller by right-clicking on the ‘EthSwtConfigs’ and choose ‘Create EthSwtConfig container’.

Choose in the field ‘Generic Driver’ the new created Ethernet Switch driver shown in Figure 9.

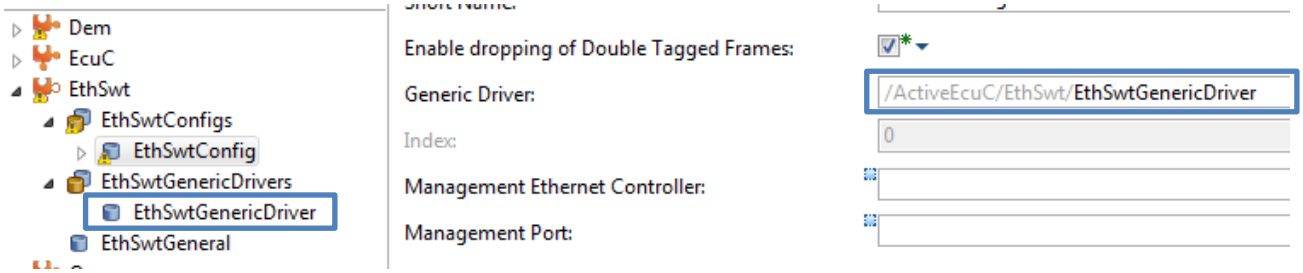


Figure 6-2 Ethernet Switch configuration

6.1 Configuration Variants

The EthSwt supports the configuration variants

> VARIANT-PRE-COMPILE

The configuration classes of the EthSwt parameters depend on the supported configuration variants. For their definitions please see the EthSwt_Generic_bswmd.arxml file.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
DaVinci Configurator PRO	Generation tool for the MICROSAR components

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
EAD	Embedded Architecture Designer
ECU	Electronic Control Unit
HIS	Hersteller Initiative Software
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
PPORT	Provide Port
RPORT	Require Port
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com