MCAL User Manual for Eth_17_GEthMacV2

32-bit TriCore™ AURIX™ TC3xx microcontroller

About this document

Scope and purpose

This User Manual is intended to enable users to integrate the Microcontroller Abstraction Layer (MCAL) software for the TriCore™ AURIX™ family of 32-bit microcontrollers.

This document describes responsibilities of integrator in-charge of integrating MCAL software with the basic software (BSW) stack. This document also provides detailed information on safety, configuration and functions along with examples of usage of significant features.

Note: Detailed information about package installation, safety and other generic information that are common across all modules are provided in MCAL User Manual General.

Intended audience

This document is intended for anyone using the Eth_17_GEthMacV2 module of the TC3xx MCAL software.

Document conventions

Table 1 Conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, table and figure captions, screen names, windows, dialog boxes, menus, sub-menus
Italics	Denotes variable(s) and reference(s)
`Courier New`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
>	Indicates that a cascading sub-menu opens when you select a menu item
[cover parentID=<alpha numeric value>]	Used for traceability completeness. Reader should ignore these.

Reference documents

This User Manual should be read in conjunction with the following documents:

- AURIX™ TC3xx MCAL User Manual General
- Specification of Driver, AUTOSAR_SWS_EthernetDriver, AUTOSAR Release 4.4.0

Table of contents

Table of contents

Table of contents

Table of contents

Table of contents

Table of contents

1 Eth_17_GEthMacV2 driver

1.1 User information

1.1.1 Description

The Ethernet (ETH) driver is responsible for providing standard ETH controller services specified by AUTOSAR 4.4.0. This enables the upper layers (for example, ETH interface) to access the underlying bus system in a uniform manner. The ETH driver provides functionality for configuration, initialization, data transmission and reception. The ETH driver provides optional features such as checksum offloading, time stamping, updating the physical source address, read or write interface to ETH transceiver, Ethernet switch management support functionality, ETH statistics and ETH error counter values.

1.1.2 Hardware-software mapping

This section describes the system view of the ETH driver and peripherals administered by it.

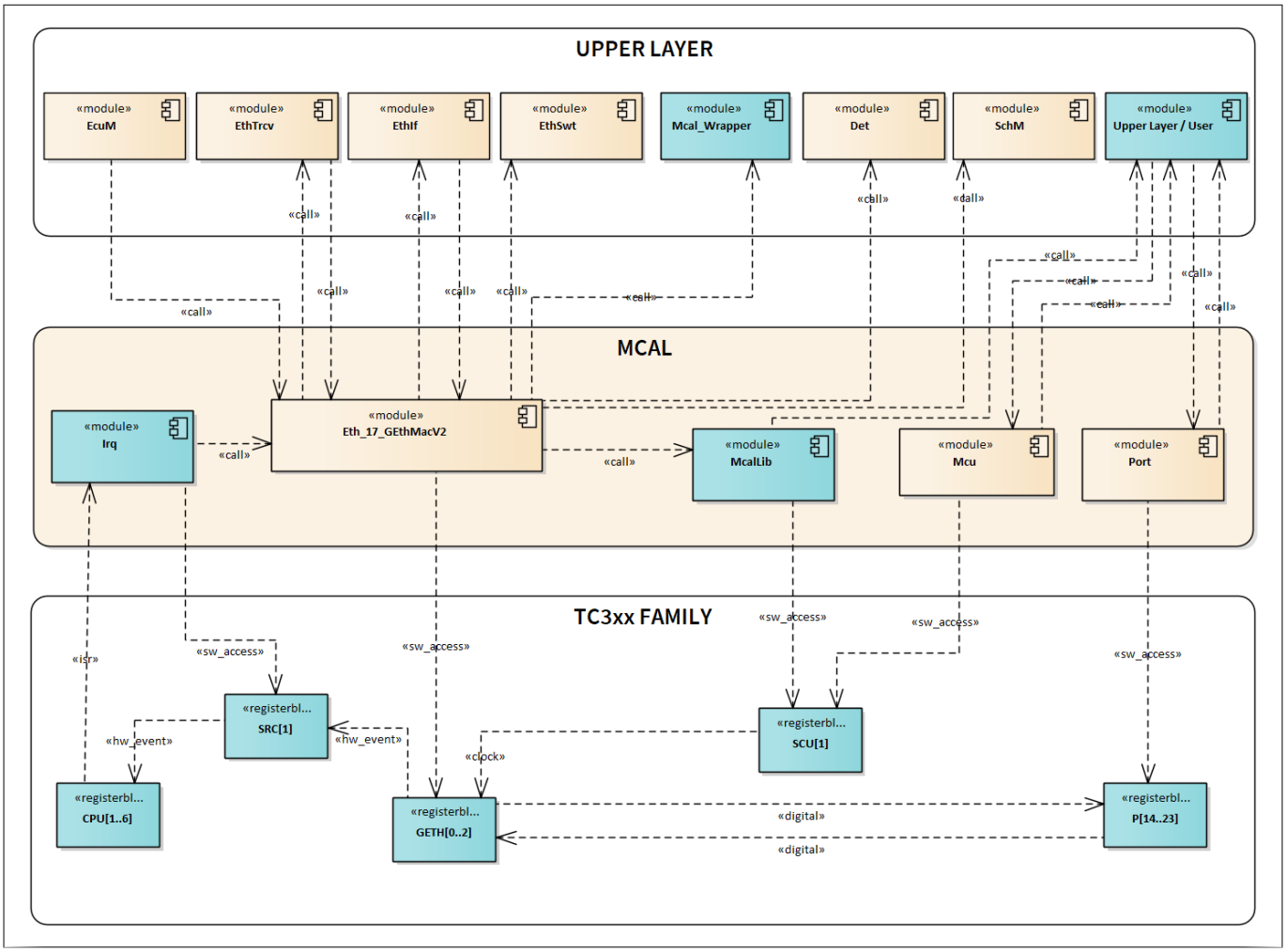


Figure 1 Mapping of hardware-software interfaces

1.1.2.1 GETH: primary hardware peripheral

Hardware functional features

1 Eth_17_GEthMacV2 driver

The ETH driver uses the GETH for initialization, configuration and data transmission.

The key hardware functional features used by the ETH driver are:

- Data transmission speed supported are 10 Mbps, 100 Mbps and 1 Gbps
- PHY interfaces supported are MII, RMII and RGMII
- Full-duplex and half-duplex modes are supported for data transmission
- Preamble and start of packet data (SFD) insertion/deletion
- Automatic CRC and pad generation/stripping options
- Up to 32 layer 2 (MAC) address filtering
- ETH frame time stamp (supports IEEE 1588-2008 for precision network clock synchronization)
- Checksum offloading for IPV4, ICMP, TCP and UDP frames
- Embedded DMA in the ETH controller is used for data exchanges between the ETH controller and the system memory
- Multiple DMA channels and multiple MTL queues from GETH MAC hardware is used
- MTL queue is built from FIFO memory available in GETH MAC for transmission and reception of size 4 Kbyte and 8 Kbyte respectively. Entire FIFO memory can be split to configure upto 4 queues. Each queue can be configured to different sizes in multiples of 256 bytes
- Standard ETH of frame size 1518 bytes (1522 bytes with VLAN support)
- Upto 32 destination address filters
- Broadcast frames are always allowed irrespective of the filter status
- GETH is implemented as a 32 bit peripheral. Nevertheless, it is connected to 64 bit wide bus (SRI)

The unsupported features of the ETH driver are:

- Double VLAN tagged frames
- Jumbo frames
- Source address filters are not supported
- Loopback mode
- Energy efficient ETH (EEE)
- Pulse per second output
- VLAN filtering

Users of the hardware

The ETH driver exclusively utilizes the GETH module.

Hardware diagnostic features

- The SMU alarms configured for the GETH are not monitored by the ETH driver
- ETH packet drop counts and packet statistics are available through the ETH driver API services

Hardware events

The ETH driver uses the following hardware events from the GETH IP:

- Packet transmission complete

- Packet reception complete

1.1.2.2 SRC: dependent hardware peripheral

Hardware functional features

The ETH driver depends on the interrupt router for raising an interrupt to the CPU based on the transmit and receive events, which indicates successful packet transmission and reception respectively.

Users of the hardware

The interrupt router is configured by the user software. The ETH driver does not administer any functional block of the interrupt router.

Hardware diagnostic features

The SMU alarms configured for interrupt router are not monitored by the ETH driver.

Hardware events

The interrupt events raised by the interrupt router are serviced by the CPU. The ETH driver provides interrupt handlers as software interfaces, which must be invoked from the ISR.

1.1.2.3 SCU: dependent hardware peripheral

Hardware functional features

The ETH driver depends on the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB, fSRI and fGETH clock signals for functioning. The fGETH defines the basic frequency for the GETH kernel. The fGETH is independent to fSPB and allows the GETH to operate at a constant baud rate.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the ETH driver.

Hardware events

Not applicable.

1.1.2.4 PORT: dependent hardware peripheral

Hardware functional features

The MII/ RMII/ RGMII and MDIO signals are routed to the transceiver through the port pads. These signals are configured and enabled through the PORT driver.

Users of the hardware

The port pads are configured by the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events from port pads are not used by the ETH driver.

1.1.3 File structure

1.1.3.1 C file structure

This section provides details of the C files of the ETH driver.

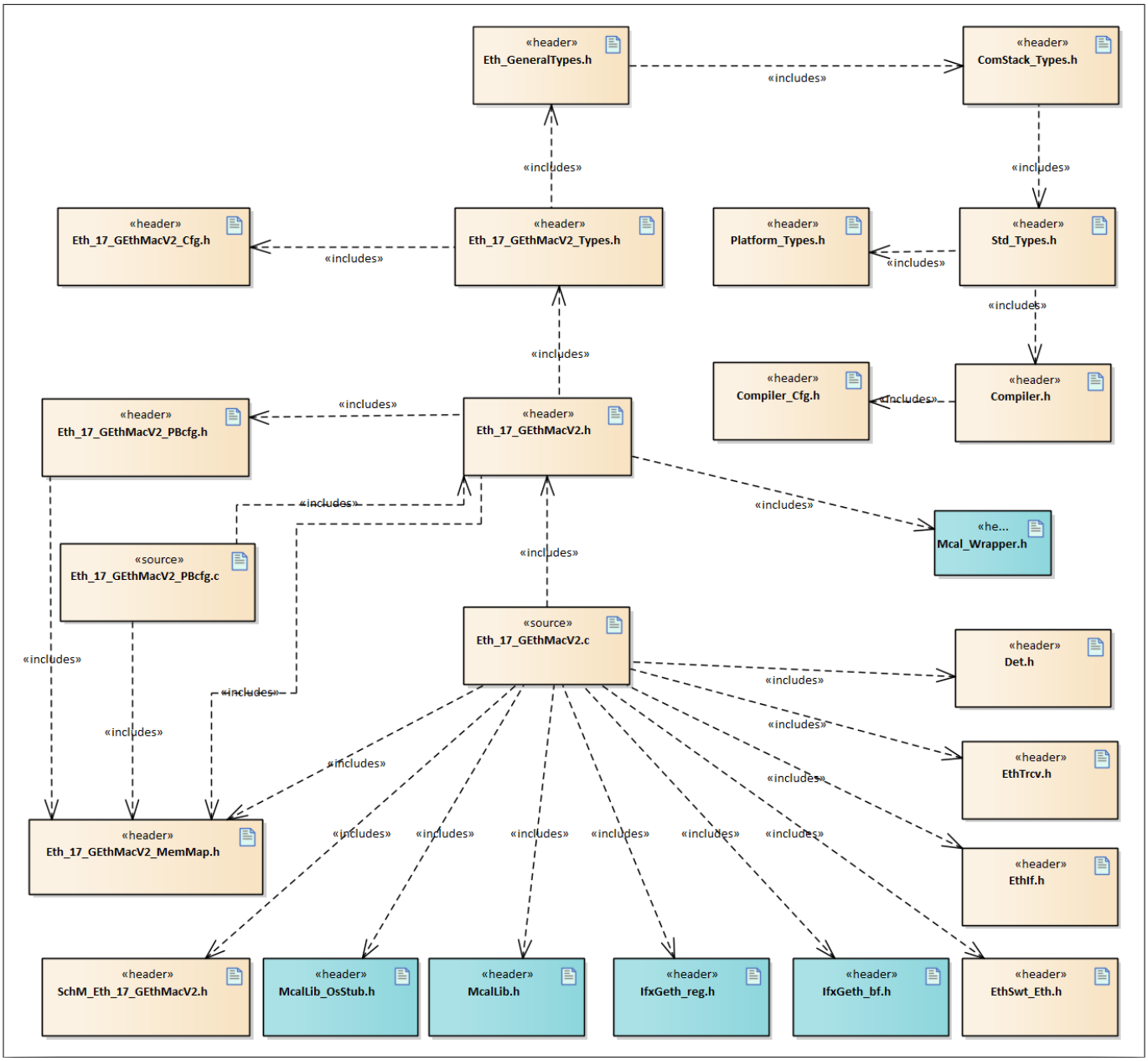


Figure 2 Eth_17_GEthMacV2_C_File_Structure-1.png

Table 2 C file structure

File name	Description
`ComStack_Types.h`	Type Definition for Com stack
`Compiler.h`	Provides abstraction from compiler-specific keywords

(table continues...)

Table 2 (continued) C file structure

File name	Description
Compiler_Cfg.h	Configuration header file for compiler abstraction
Det.h	Provides the exported interfaces of Development Error Tracer
EthIf.h	Provides the exported interfaces of Ethernet Interface (EthIf).
EthSwt_Eth.h	Contains the declarations of the functions of Ethernet Switch (EthSwt) module that are invoked by the ETH module.
EthTrcv.h	Provides the exported interfaces of Ethernet Transceiver (EthTrcv) driver.
Eth_17_GEthMacV2.c	File (Static) containing implementation of APIs
Eth_17_GEthMacV2.h	Header file (Static) defining prototypes of data structures, APIs and interrupt handlers
Eth_17_GEthMacV2_Cfg.h	Header file (Generated) containing constants and pre-processor macros as #defines
Eth_17_GEthMacV2_MemMap.h	File (Static) containing the memory section definitions used by the ETH driver
Eth_17_GEthMacV2_PBcfg.c	File (Generated) containing definition of the configuration data structures
Eth_17_GEthMacV2_PBcfg.h	File (Generated) containing declaration of the post-build configuration data structures of ETH driver
Eth_17_GEthMacV2_Types.h	ETH driver specific type declaration file
Eth_GeneralTypes.h	ETH specific type declaration file as defined by AUTOSAR
IfxGeth_bf.h	SFR header file for GETH
IfxGeth_reg.h	SFR header file for GETH
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB.
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore. This shall be included by other drivers to call OS APIs.
Mcal_Wrapper.h	Provides the exported interfaces for Production Error and Runtime Development Errors. Implemented by default to include functions of Dem.h and Det.h files. This file can be modified by the user but function prototype is not user modifiable.
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Eth_17_GEthMacV2.h	Export header for SchM functions of the ETH
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

1.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the ETH driver.

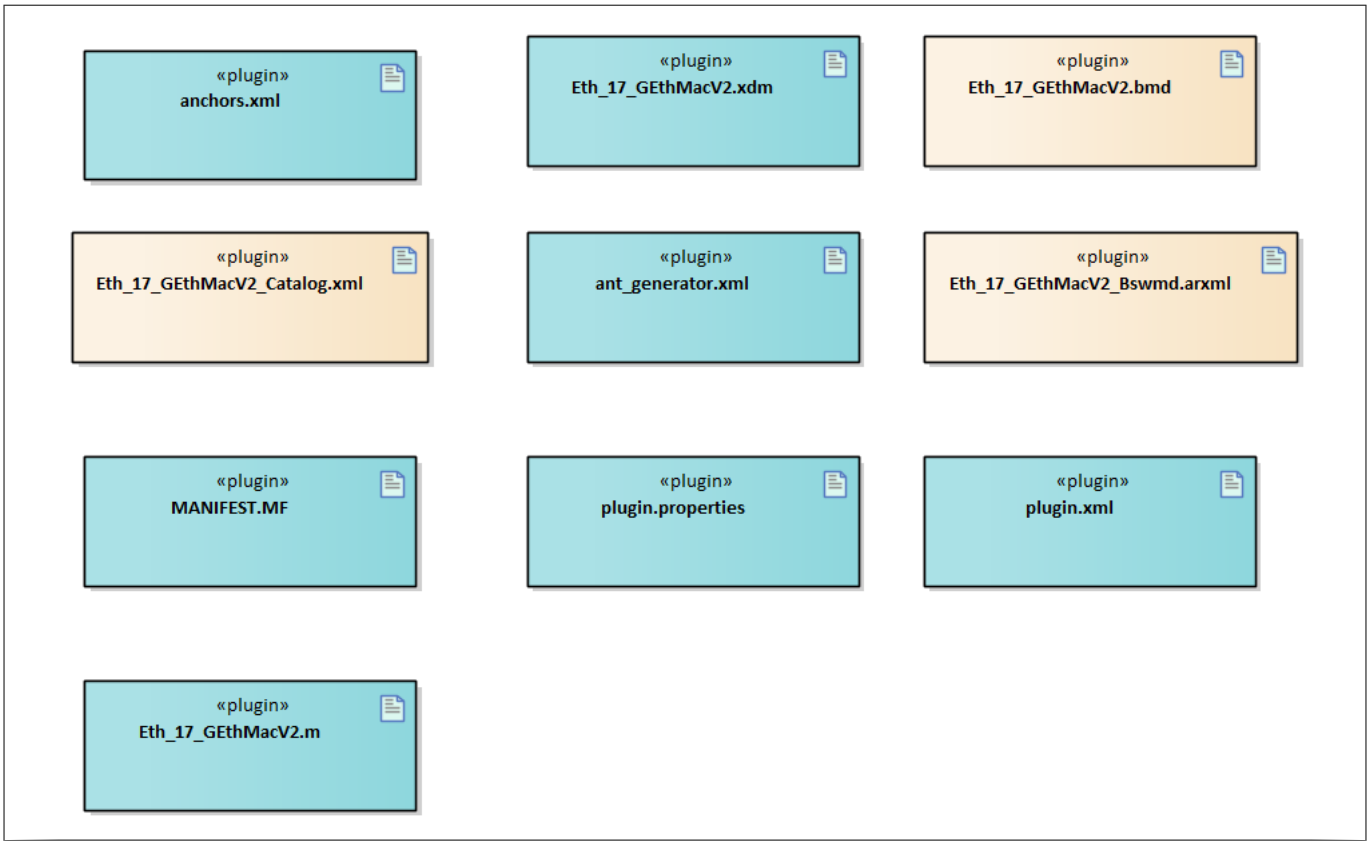


Figure 3 **Eth_17_GEthMacV2_Code_Generator_Plugin_Files-1.png**

Table 3 **Code generator plugin files**

File name	Description
`Eth_17_GEthMacV2.bmd`	AUTOSAR format XML data model schema file
`Eth_17_GEthMacV2.m`	Code template macro file for the Eth_17_GEthMacV2 driver
`Eth_17_GEthMacV2.xdm`	Tresos format XML data model schema file
`Eth_17_GEthMacV2_Bswmd.arxml`	AUTOSAR format module description file
`Eth_17_GEthMacV2_Catalog.xml`	AUTOSAR format catalog file
`MANIFEST.MF`	Tresos plugin support file containing the metadata for the ETH driver
`anchors.xml`	Tresos anchors support file for the ETH driver
`ant_generator.xml`	Tresos support file to generate and rename multiple post-build configuration when using variation point
`plugin.properties`	Tresos plugin support file for the ETH driver
`plugin.xml`	Tresos plugin support file for the ETH driver

1.1.4 Integration hints

This section lists the key points that an integrator or user of the ETH driver must consider.

1.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of MCAL, but are required to integrate the ETH driver.

- **EcuM**

  The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of MCAL, the EcuM is used for initialization and de-initialization of the software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **ETH Interface (EthIf)**

  The EthIf module is part of the AUTOSAR stack that provides the upper layers, a hardware independent interface, to the ETH communication system comprising multiple ETH controllers.

  The ETH driver uses the APIs of EthIf module to provide transmit confirmation, indicate successful reception and indicate successful ETH controller mode change. The files `EthIf.c` and `EthIf.h` are provided as stub code and needs to be replaced with the complete EthIf module during integration phase.

- **ETH Transceiver Driver (EthTrcv)**

  The EthTrcv module is part of the AUTOSAR stack that provides the upper layers, a hardware independent interface comprising multiple equal transceivers.

  The ETH driver uses APIs of EthTrcv to indicate successful Media independent (MII) read and write access. The files `EthTrcv.c` and `EthTrcv.h` are provided as stub code and needs to be replaced with complete EthTrcv module during integration phase.

- **ETH Switch Driver (EthSwt)**

  The EthSwt module is part of the AUTOSAR stack that provides to the upper layers (e.g. Ethernet Interface) , a hardware independent interface comprising a switch with several ports.

  The ETH driver invokes the APIs of EthSwt module to inform the ETH Switch Driver about a required special treatment for switch management purpose, if the switch management support functionality is enabled. If this functionality is disabled, the APIs of EthSwt module are not invoked by the ETH driver and thus, the integration with EthSwt module is optional.

  The files `EthSwt.c` and `EthSwt_Eth.h` are provided as stub code and needs to be replaced with the complete EthSwt module during integration phase.

- **Memory mapping**

  Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. In order to achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the file `Eth_17_GEthMacV2_MemMap.h`.

  The `Eth_17_GEthMacV2_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are re-located to the correct memory region.

  Note: The GETH MAC IP is implemented as a 32 bit peripheral. Nevertheless it is connected to 64 bit wide bus (SRI). To make full use of the possible performance of SRI and its bridges, it is advised to align the Tx and Rx data buffers and the descriptors to 64 bit addresses(8 byte). To support the placement of the data buffers and descriptors to 64-bit addresses, separate Memmap sections are assigned for them, namely `ETH_17_GETHMACV2_TXBUFFER_[START/ STOP]_SEC_VAR_CLEARED_QM_CORE0_32`, `ETH_17_GETHMACV2_TXDESC_[START/ STOP]_SEC_VAR_CLEARED_QM_CORE0_32`, `ETH_17_GETHMACV2_RXBUFFER_[START/ STOP]_SEC_VAR_CLEARED_QM_CORE0_32` and `ETH_17_GETHMACV2_RXDESC_[START/ STOP]_SEC_VAR_CLEARED_QM_CORE0_32`. The regular 32-bit variables are assigned to a separate Memmap section (different from data buffers and descriptors) as these do not need to be aligned to 64-bit addresses.

A sample implementation listing the memory-section macros is shown as follows.

```
/**** GLOBAL DATA SECTION for CORE[x] , x(Core Id)= 0..5 ****/
#if defined ETH_17_GETHMACV2_START_SEC_VAR_CLEARED_QM_CORE0_UNSPECIFIED
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_START_SEC_VAR_CLEARED_QM_CORE0_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_STOP_SEC_VAR_CLEARED_QM_CORE0_UNSPECIFIED
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_STOP_SEC_VAR_CLEARED_QM_CORE0_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_TXBUFFER_START_SEC_VAR_CLEARED_QM_CORE0_32
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_TXBUFFER_START_SEC_VAR_CLEARED_QM_CORE0_32
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_TXBUFFER_STOP_SEC_VAR_CLEARED_QM_CORE0_32
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_TXBUFFER_STOP_SEC_VAR_CLEARED_QM_CORE0_32
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_TXDESC_START_SEC_VAR_CLEARED_QM_CORE0_32
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_TXDESC_START_SEC_VAR_CLEARED_QM_CORE0_32
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_TXDESC_STOP_SEC_VAR_CLEARED_QM_CORE0_32
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_TXDESC_STOP_SEC_VAR_CLEARED_QM_CORE0_32
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_RXBUFFER_START_SEC_VAR_CLEARED_QM_CORE0_32
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_RXBUFFER_START_SEC_VAR_CLEARED_QM_CORE0_32
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_RXBUFFER_STOP_SEC_VAR_CLEARED_QM_CORE0_32
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_RXBUFFER_STOP_SEC_VAR_CLEARED_QM_CORE0_32
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_RXDESC_START_SEC_VAR_CLEARED_QM_CORE0_32
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_RXDESC_START_SEC_VAR_CLEARED_QM_CORE0_32
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_RXDESC_STOP_SEC_VAR_CLEARED_QM_CORE0_32
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_RXDESC_STOP_SEC_VAR_CLEARED_QM_CORE0_32
 #undef MEMMAP_ERROR

/**** ETH MODULE CONFIG DATA ****/
#elif defined ETH_17_GETHMACV2_START_SEC_CONFIG_DATA_QM_GLOBAL_UNSPECIFIED
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_START_SEC_CONFIG_DATA_QM_GLOBAL_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_STOP_SEC_CONFIG_DATA_QM_GLOBAL_UNSPECIFIED
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_STOP_SEC_CONFIG_DATA_QM_GLOBAL_UNSPECIFIED
 #undef MEMMAP_ERROR
```

```
/**** CODE SECTION ****/
#elif defined ETH_17_GETHMACV2_START_SEC_CODE_QM_GLOBAL
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_START_SEC_CODE_QM_GLOBAL
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_STOP_SEC_CODE_QM_GLOBAL
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_STOP_SEC_CODE_QM_GLOBAL
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_START_SEC_CODE_QM_LOCAL
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_START_SEC_CODE_QM_LOCAL
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMACV2_STOP_SEC_CODE_QM_LOCAL
 /* User Pragma here */
 #undef ETH_17_GETHMACV2_STOP_SEC_CODE_QM_LOCAL
 #undef MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Eth_17_GEthMacV2_MemMap.h, wrong pragma command"
#endif
```

- **DET**

  The DET module is a part of the AUTOSAR stack that handles all the development errors reported by the BSW modules. The ETH driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the ETH driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

  The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **Mcal_Wrapper**

  This driver performs reporting of the Production and Runtime errors. The handling of the reported errors shall be done by the user. The `Mcal_Wrapper_Det_ReportRuntimeError()` API, `Mcal_Wrapper_Dem_SetEventStatus()` API and `Mcal_Wrapper_Dem_ReportErrorStatus()` API are provided in the `Mcal_Wrapper.c` and `Mcal_Wrapper.h` files as a stub code, and can be updated by the integrator to handle the reported errors. The files `Mcal_Wrapper.c` and `Mcal_Wrapper.h` are user modifiable but the function prototypes are not user modifiable and by default the Mcal_Wrapper function shall call AUTOSAR DEM and DET Modules.

  The user of the ETH driver must process all the production errors (fail / pass) reported to the Mcal_Wrapper module through the `Mcal_Wrapper_Dem_SetEventStatus()` API. The interface used for reporting production error in AUTOSAR version 4.4.0 is `Mcal_Wrapper_Dem_SetEventStatus()`. The `Mcal_Wrapper.c` and `Mcal_Wrapper.h` files are provided in the MCAL package as a stub code and can be replaced with a user specific production error handling module during the integration phase.

- **SchM**

  The SchM module is a part of the RTE that manages the BSW Scheduler. The ETH driver uses the exclusive areas defined in the `SchM_Eth_17_GEthMacV2.c` file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the ETH driver are:

  (i) TxDescData

(ii) TxBufIdx

(iii) TxDescDataISR (implementation required only when the interrupt type is CAT2, else can be kept blank)

The `SchM_Eth_17_GEthMacV2.h` and `SchM_Eth_17_GEthMacV2.c` files are provided in the MCAL package as an example code and needs to be updated by the integrator. The user must implement the SchM functions defined by the ETH driver as **suspend / resume** of interrupts for the CPU on which the API is invoked.

*Note: The SchM functions (Enter/ Exit) for the exclusive area **TxDescDataISR** must be implemented as **suspend / resume** of interrupts for the CPU only if the interrupt type is CAT2. Otherwise, these functions can be kept blank.*

A sample implementation of the SchM functions is shown as follows:

```
/**** Sample implementation of SchM_Eth_17_GEthMacV2.c ****/
#include "Os.h"

void SchM_Enter_Eth_17_GEthMacV2_TxDescData(void)
{
 SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Eth_17_GEthMacV2_TxDescData(void)
{
 ResumeAllInterrupts(); /* Resume CPU core interrupt */
}

void SchM_Enter_Eth_17_GEthMacV2_TxBufIdx(void)
{
 SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Eth_17_GEthMacV2_TxBufIdx(void)
{
 ResumeAllInterrupts(); /* Resume CPU core interrupt */
}

/* Below SchM sections are required only when interrupt type is CAT2 */
void SchM_Enter_Eth_17_GEthMacV2_TxDescDataISR(void)
{
 SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Eth_17_GEthMacV2_TxDescDataISR(void)
{
 ResumeAllInterrupts(); /* Resume CPU core interrupt */
}
```

- **Safety error**

  The ETH driver does not report any safety errors.
- **Notifications and callbacks**

  The ETH driver itself does not implement any notifications. However, the ETH driver reports transmit confirmation, successful reception indication and controller mode change through notification functions of

the EthIf module and successful Media Independent Interface(MII) read / write access through notification functions of EthTrcv module.

- **Operating system (OS)**

  The OS or application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application. The OS files provided by MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

1.1.4.2 Multicore and Resource Manager

The ETH driver supports execution of its APIs simultaneously from all CPU cores. The user should allocate ETH controllers to the CPU cores at pre-compile time using the Resource Manager module. The following are the key points to be considered with respect to multicore in the driver:

- ETH controller can be allocated to CPU cores at pre-compile time. For example, EthCtrlConfig_0, EthCtrlConfig_1.
- It must be ensured that the ETH controller index passed as parameter while invoking an API, belongs to the same core.
- Development errors are reported in case APIs are invoked with mismatch of core and controller index.
- Locating constants, variables and configuration data to correct memory space should be done by the user. Memory sections are marked GLOBAL (common to all cores) and CORE[x](specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

  Code section:

  The executable code of ETH driver is placed under single MemMap section. It can be relocated to any PFlash region.

  Data section:

  The RAM variable memory sections marked as specific to core, should be re-located to the DSPR/DLMU of the same core. The sections marked as global should be relocated to the non-cached LMU region.

  Configuration data and constants:

  The configuration data section sections marked as specific to core, should be re-located to the PFlash of the same core. The sections marked as global should be relocated to the PFlash of the master core.

  Note: Relocating of code, data and constants to a distant memory space would impact execution timings.

1.1.4.3 MCU support

The ETH driver is dependent on the MCU driver for the generation of fSRI, fSPB and fGETH clocks. The initialization of the ETH driver must be started only after completing the MCU driver initialization. The fGETH defines the application clock frequency for the Gigabit ETH Kernel. The fGETH is independent of fSPB and allows the Gigabit ETH to operate at a constant baud rate (frequency). To configure fSRI, fSPB and fGETH clock frequencies, update the parameters `McuSRIFrequency`, `McuSPBFrequency` and `McuGEthFrequency` within the MCU driver configuration.

1.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure port pins used by the ETH driver through the PORT configuration and initialize the PORT driver prior to invoking of the ETH driver initialization. The following must be considered while configuring PORT driver in EB Tresos:

- Configure all PORT pins that are used in the ETH driver for MII/RMII/RGMII and MDIO interface with PHY. That is, parameters such as `PortPinDirection` (input or output), `PortPinInitialMode` (as GPIO for input pin or corresponding ALT option for output pins) and so on.

- For all output and input pins used by the ETH driver for MII/RMII/RGMII interface, the value of parameter `PortPinOutputPadDriveStrength` shall be configured as PORT_PIN_RGMII_DRIVER.
- For all output pins used by the ETH driver, the parameter `PortPinControllerSelect` shall be selected as ENABLE.

  Refer to the following sample configurations for the PORT driver:

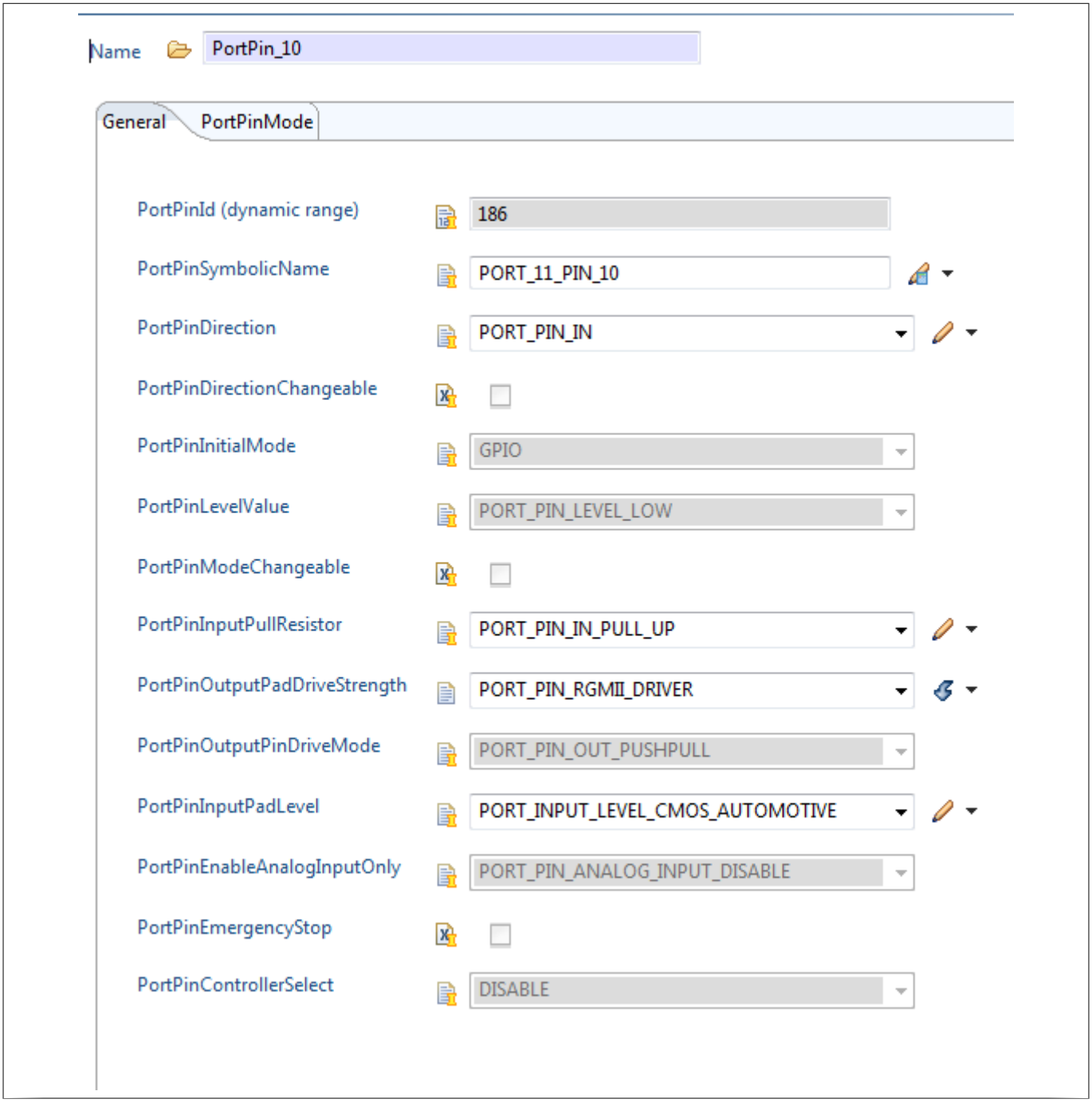


Figure 4 Input pin configuration

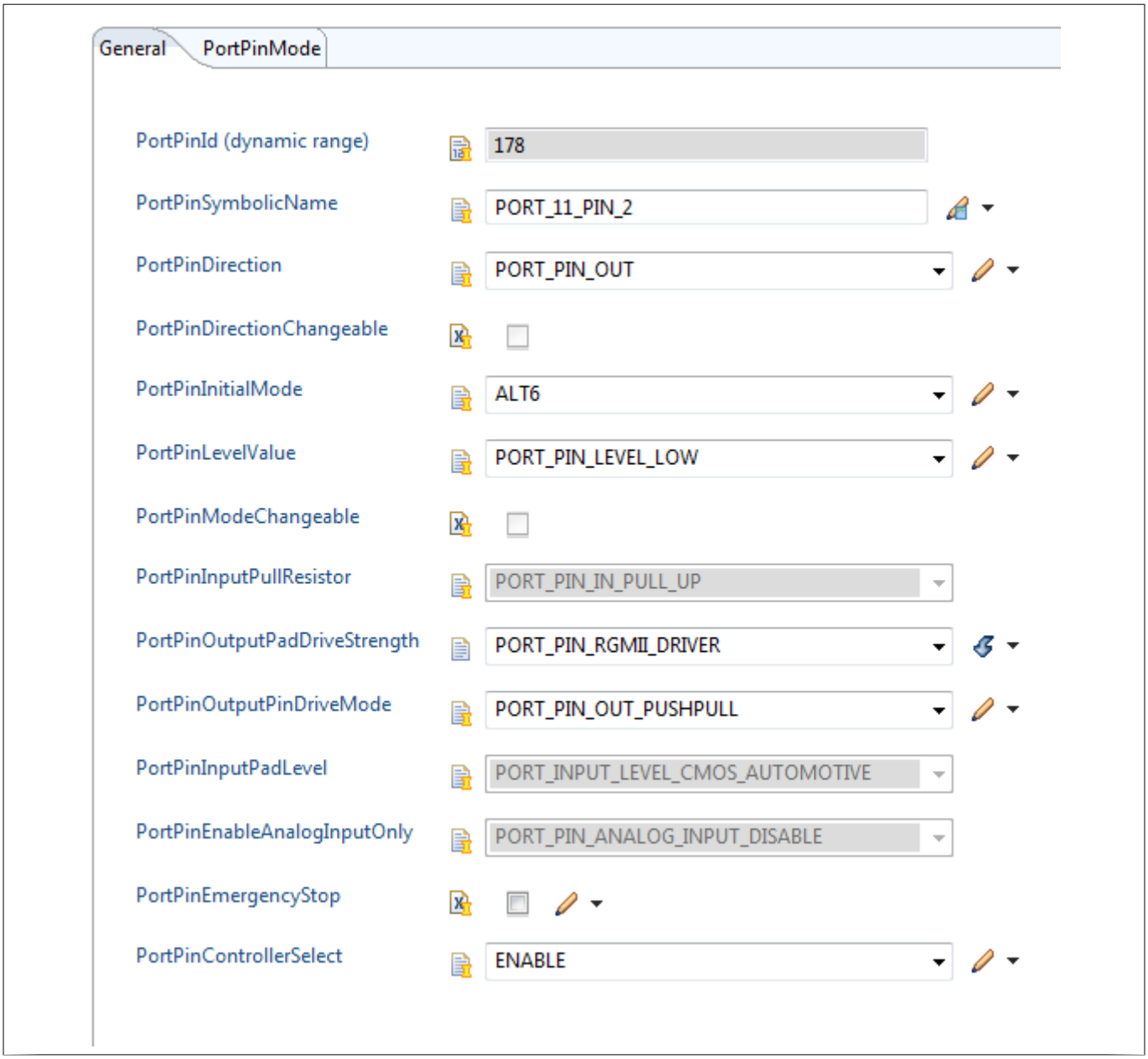


Figure 5 Output pin configuration

1.1.4.5 DMA support

The ETH controller has its own DMA in hardware and the ETH driver implements the necessary code to exercise this DMA. Therefore, the ETH driver does not use any services of the MCAL DMA available in the TC3xx device.

The following must be ensured by the user for proper functioning of the DMA controller:

- Address space 0xD and 0xC shall not be used for DMA-related usage. MemMap sections allocating memory in the scratch pad RAM shall always generate global address instead of local addresses.
- The memory addresses accessed by DMA, that is, the memory-section macros `ETH_17_GETHMACV2_TXBUFFER_[START/ STOP]_SEC_VAR_CLEARED_QM_CORE0_32`, `ETH_17_GETHMACV2_TXDESC_[START/ STOP]_SEC_VAR_CLEARED_QM_CORE0_32`, `ETH_17_GETHMACV2_RXBUFFER_[START/ STOP]_SEC_VAR_CLEARED_QM_CORE0_32` and `ETH_17_GETHMACV2_RXDESC_[START/ STOP]_SEC_VAR_CLEARED_QM_CORE0_32`, shall be placed in non-cached memory.

Note: The variables defined in the ETH driver for transmit buffer, receive buffer, transmit DMA descriptor list and receive DMA descriptor list will be accessed by the DMA controller.

1.1.4.6 Interrupt connections

The interrupt connections of the ETH driver are described in this section.

- **Packet transmission complete interrupt from Tx DMA channels 0 to 3 for controller Index 0 and 1**

  When ETH transmission is configured in the interrupt mode and a requested packet transmission is completed, interrupt will be generated.

  The service request line numbers SRC_GETH2, SRC_GETH3, SRC_GETH4 and SRC_GETH5 are used for transmission complete interrupt for controller index 0. For controller index 1, the service request line numbers SRC_GETH12, SRC_GETH13, SRC_GETH14 and SRC_GETH15 are used for transmission complete interrupt.

  The number of Tx DMA channels used depends on the number of egress FIFOs configured within the configuration. As an example, if only two egress FIFOs are configured in controller 0, the service request lines SRC_GETH2 and SRC_GETH3 are used.

  User must ensure that the interrupt handler provided by ETH driver is called when packet transmission complete interrupt occurs. A sample invocation for transmission complete interrupt from DMA channel-0 of controller index 0 is as follows:

```
/* Include Eth header file Eth_17_GEthMacV2.h */
#include "Eth_17_GEthMacV2.h"

/********TX Interrupt from DMA Channel-0, Controller Id 0*********/
ISR(ETHSR2_ISR)
{
 /* Enable Global Interrupts */
 ENABLE();
 /* Invoke Irq handler from GETH module file */
 Eth_17_GEthMacV2_TxDmaIrqHdlr(0, 0);
}

/********TX Interrupt from DMA Channel-2, Controller Id 1*********/
ISR(ETHSR14_ISR)
{
 /* Enable Global Interrupts */
 ENABLE();
 /* Invoke Irq handler from GETH module file */
 Eth_17_GEthMacV2_TxDmaIrqHdlr(1, 2);
}
```

- **Packet receive complete interrupt from Rx DMA channel 0 to 3 for controller index 0 and 1**

  When ETH reception is configured in the interrupt mode and a packet is received by ETH controller, interrupt will be generated.

  The service request line numbers SRC_GETH6, SRC_GETH7, SRC_GETH8 and SRC_GETH9 are used for receive complete interrupt for controller index 0. For controller index 1, the service request line numbers SRC_GETH16, SRC_GETH17, SRC_GETH18 and SRC_GETH19 are used for receive complete interrupt.

  The number of Rx DMA channels used depends on the number of ingress FIFOs configured within the configuration. As an example, if only two ingress FIFOs are configured in controller 0, the service request lines SRC_GETH6 and SRC_GETH7 are used.

User must ensure that the interrupt handler provided by the ETH driver is called when packet receive complete interrupt occurs. A sample invocation for receive complete interrupt from DMA channel-0 of controller index 0 is as follows:

```
/* Include Eth header file Eth_17_GEthMacV2.h */
#include "Eth_17_GEthMacV2.h"

/********RX Interrupt from DMA Channel-0, Controller Id 0*********/
ISR(ETHSR6_ISR)
{
 /* Enable Global Interrupts */
 ENABLE();
 /* Invoke Irq handler from GETH module file */
 Eth_17_GEthMacV2_RxDmaIrqHdlr(0, 0);
}

/********RX Interrupt from DMA Channel-3, Controller Id 1*********/
ISR(ETHSR19_ISR)
{
 /* Enable Global Interrupts */
 ENABLE();
 /* Invoke Irq handler from GETH module file */
 Eth_17_GEthMacV2_RxDmaIrqHdlr(1, 3);
}
```

The below table provides information about the service request line numbers used for ETH controllers 0 and 1.

Table 4 Interrupt Service Request Lines for ETH controllers 0 and 1

ETH Controller	DMA Channel	Service Request Line
Controller 0	Tx DMA Channel 0	SRC_GETH2
	Tx DMA Channel 1	SRC_GETH3
	Tx DMA Channel 2	SRC_GETH4
	Tx DMA Channel 3	SRC_GETH5
	Rx DMA Channel 0	SRC_GETH6
	Rx DMA Channel 1	SRC_GETH7
	Rx DMA Channel 2	SRC_GETH8
	Rx DMA Channel 3	SRC_GETH9
Controller 1	Tx DMA Channel 0	SRC_GETH12
	Tx DMA Channel 1	SRC_GETH13
	Tx DMA Channel 2	SRC_GETH14
	Tx DMA Channel 3	SRC_GETH15
	Rx DMA Channel 0	SRC_GETH16
	Rx DMA Channel 1	SRC_GETH17
	Rx DMA Channel 2	SRC_GETH18

(table continues...)

Table 4 (continued) Interrupt Service Request Lines for ETH controllers 0 and 1

ETH Controller	DMA Channel	Service Request Line
	Rx DMA Channel 3	SRC_GETH19

1.1.4.7 Example usage

Configuration

ETH driver must be configured before usage and configuration files are generated and made available during the software build process.

To configure ETH driver, the following guidelines shall be followed properly.

Note: User of the ETH driver must ensure that the `EthCtrlConfigIngressFifoBufLenByte` parameter for an ingress FIFO is configured as expected in the ETH bus (including the broadcast frames if any). If received packet size is more than what is configured, then such packets will be ignored by the ETH driver and receive notification will not be called.

Step1: In the MCU driver, configure the following system clocks: fSRI, fSPB and fGETH.

Step2: In the PORT driver, for all the port pins that are used in the ETH driver as interface (MII/RMII/RGMII and MDIO) with PHY, configure the same in the PORT driver.

Step3: If the ETH driver is configured in the interrupt mode (for transmit and receive), configure the interrupt priority, type of service and interrupt type in IRQ driver. ETH driver uses interrupt line SRC_GETH2, SRC_GETH3, SRC_GETH4, SRC_GETH5, SRC_GETH12, SRC_GETH13, SRC_GETH14 and SRC_GETH15 for transmission and SRC_GETH6, SRC_GETH7, SRC_GETH8, SRC_GETH9, SRC_GETH16, SRC_GETH17, SRC_GETH18 and SRC_GETH19 for receive.

Note: Ensure MAC address is unique if more than one controller is configured.

Step4: In the ETH driver, select the required API configuration such as PHY interface (MII/RMII/RGMII), ETH speed (10/100/1000 Mbps), ETH operation mode (Half/Full duplex), alternate input selection and so on.

Step5: In the Resource Manager, allocate controller with Id 0 / 1 to the required Core (example, core 0). (If this is not configured, by default the controller is assumed to be allocated to the master core as per the Resource Manager module).

Configuration: MTL Scheduling Algorithm and EthCtrlConfigSchedulerPredecessor

The GETH MAC hardware supports two algorithms, namely Strict Priority and Weighted Round Robin (WRR), for transmit scheduling of the queues at the MTL layer. The `EthCtrlConfigSchedulerAlgorithm` parameter is used to select one of the supported Tx scheduling algorithms.

- **Configuration: MTL Scheduling Algorithm - Strict Priority**

  To choose strict priority, configure the `EthCtrlConfigSchedulerAlgorithm` parameter to `ETH_SCHEDULER_STRICT_PRIORITY`. In case of strict priority, the parameter `EthCtrlConfigSchedulerPredecessorOrder` represents the order of the scheduler entities.

  When credit shaper is not configured, the egress FIFO which is referenced with the lowest value of `EthCtrlConfigSchedulerPredecessorOrder` is the FIFO that is treated as having the lowest transmission priority order by the HW. On the other hand, the FIFO referenced with the highest value of `EthCtrlConfigSchedulerPredecessorOrder` is considered by the HW as having the highest transmission priority order.

  With credit shapers configured, the `EthCtrlConfigSchedulerPredecessorOrder` for the FIFOs without shapers should have lower values than the FIFOs with shapers configured.

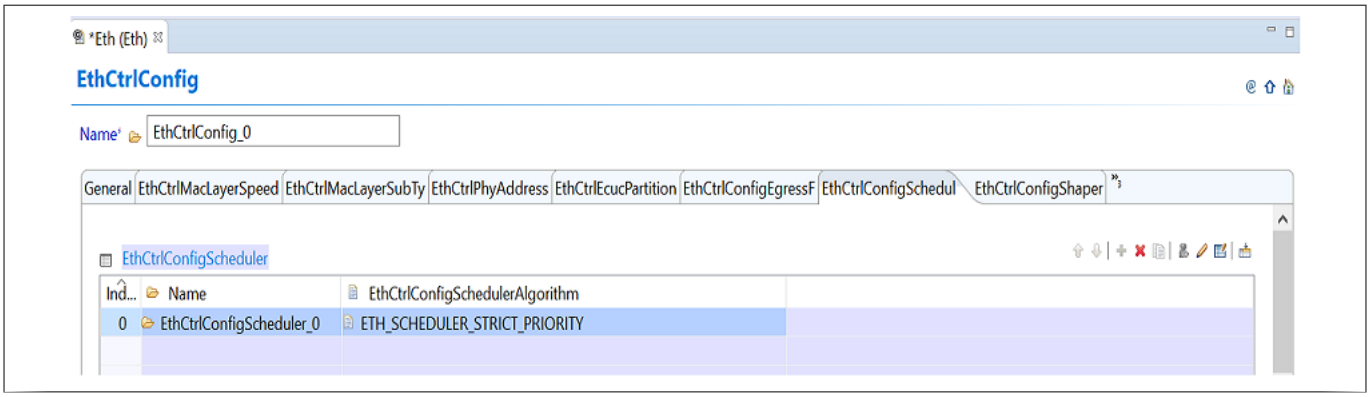


Figure 6 MTL Scheduling Algorithm - Strict Priority

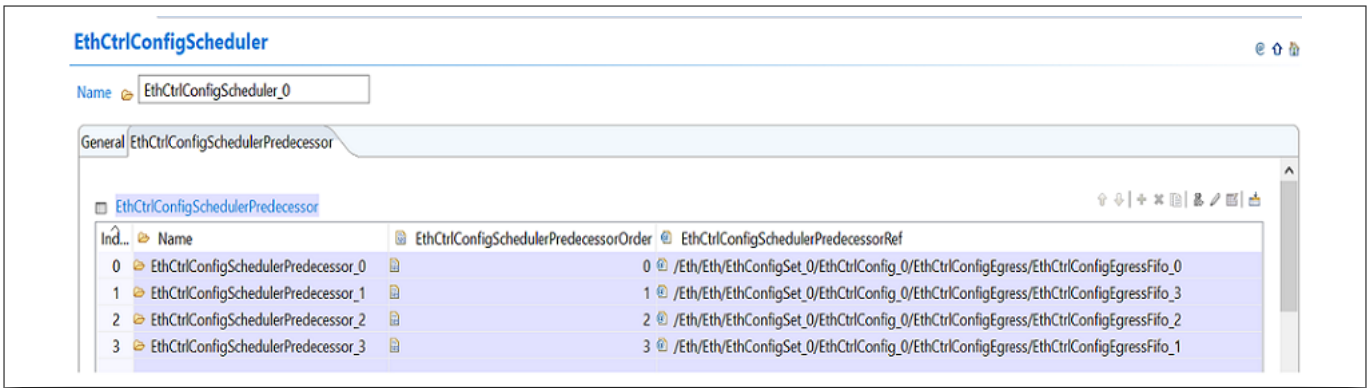


Figure 7 Strict Priority with FIFO scheduled without shaper

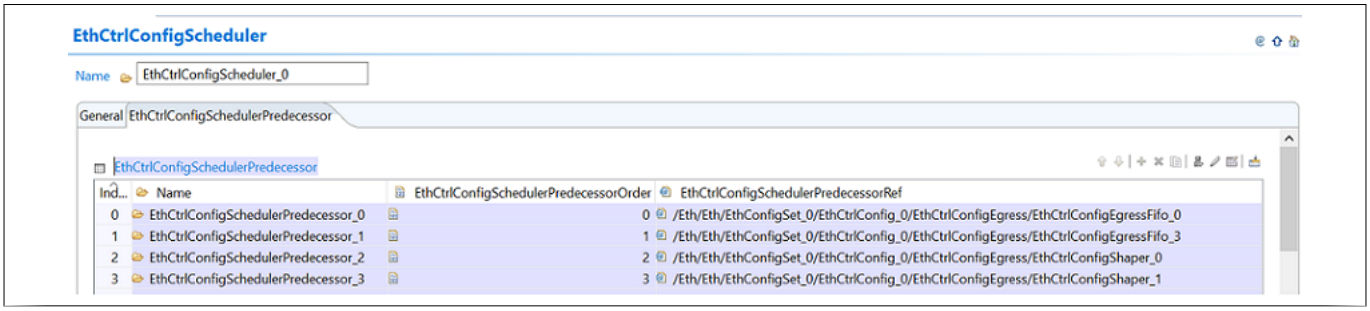


Figure 8 Strict Priority with FIFO scheduled with shaper

- **Configuration: MTL Scheduling Algorithm - Weighted Round Robin**

  In case the selected algorithm is Weighted Round Robin (WRR), the parameter `EthCtrlConfigSchedulerPredecessorOrder` is used to define the weight for the queue associated to a FIFO.

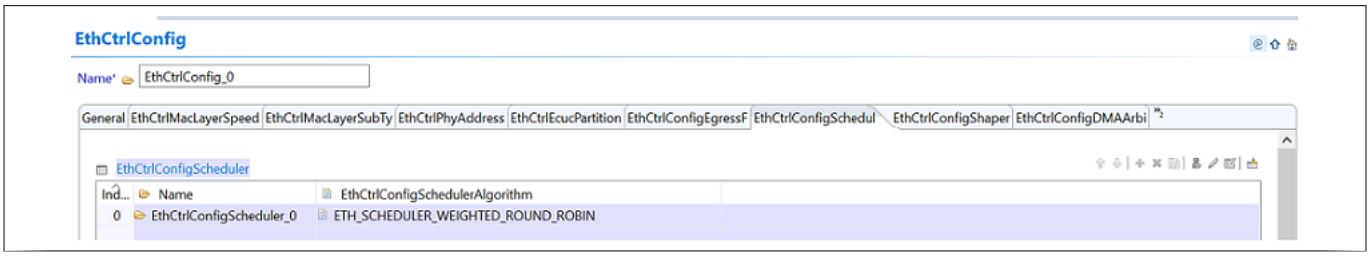


Figure 9 MTL Scheduling Algorithm - Weighted Round Robin

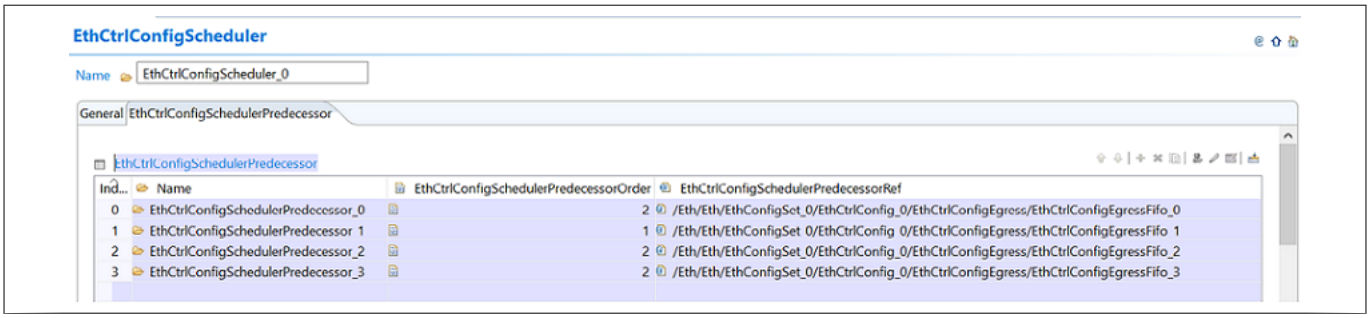


Figure 10 Weighted Round Robin with FIFOs scheduled and assigned with weights

Configuration: DMA Arbitration Algorithm

The GETH MAC hardware supports up to 4 transmit DMA channels. The DMA arbiter helps in the arbitration of all the paths on the transmit channels.

The `EthCtrlConfigDMAArbitrationAlgorithm` parameter is used to select of one of the algorithms for the transmit DMA to perform the arbitration. The supported algorithms are Fixed Priority, Weighted Strict Priority (WSP) or Weighted Round Robin (WRR).

- **DMA Arbitration Algorithm - Fixed Priority**

  In fixed priority mode, Tx DMA Channel 0 has the lowest priority and Tx DMA Channel 3 has the highest priority. Each configured egress FIFO is mapped to one of the DMA channels based on the scheduler order configured in the parameter `EthCtrlConfigSchedulerPredecessorOrder`. The Tx DMA channel 3 is assigned to the FIFO with the highest `EthCtrlConfigSchedulerPredecessorOrder` value, Tx DMA channel 2 to the FIFO with the next highest EthCtrlConfigSchedulerPredecessorOrder value and so on.

  There is no need to configure the weights in fixed priority algorithm.

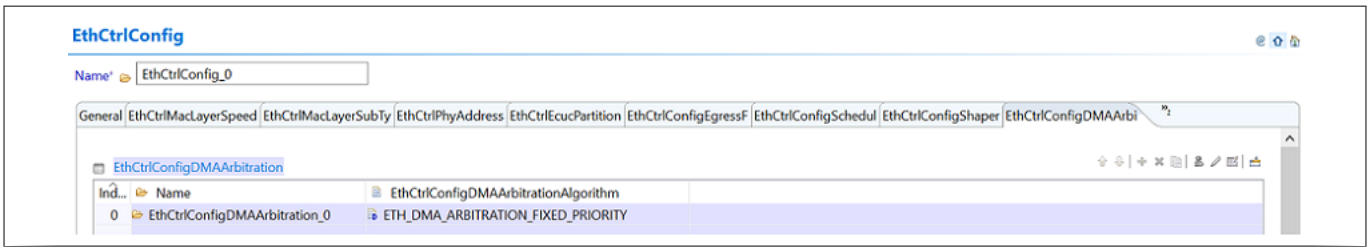


Figure 11 DMA Arbitration Algorithm - Fixed Priority

- **DMA Arbitration Algorithm - Weighted Round Robin**

  If Weighted Round Robin mode is selected, the weight needs to configured using the parameter `EthCtrlConfigDMAArbitrationWeight` for the corresponding egress FIFO referenced by the parameter `EthCtrlConfigDMAEgressFifoRef`. This egress FIFO is mapped internally to a DMA channel. In this mode, the DMA arbiter first selects the egress FIFO (mapped internally to Tx DMA channel) with the highest weight programmed, and then the FIFO with next highest weight, and so on. If any channel does not have a frame to transmit, the weight of that channel gets equally distributed to all channels that have frames to transmit.

  Note: The configured weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel.

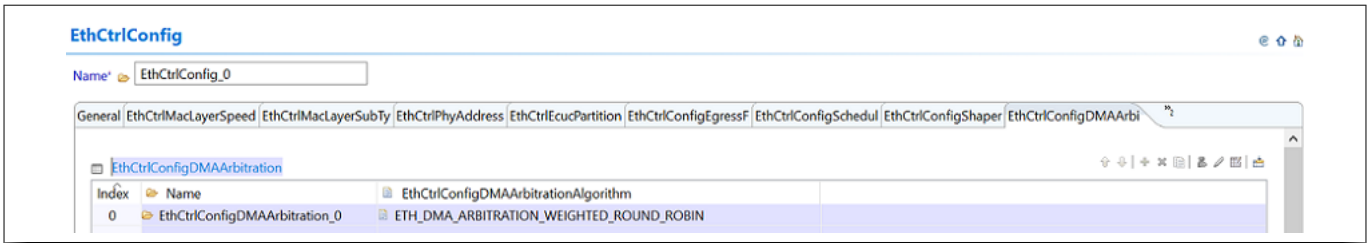


Figure 12 DMA Arbitration Algorithm - Weighted Round Robin

- **DMA Arbitration Algorithm - Weighted Strict Priority**

  If Weighted Strict Priority mode is selected, the weight needs to configured using the parameter `EthCtrlConfigDMAArbitrationWeight` for the corresponding egress FIFO referenced by the parameter `EthCtrlConfigDMAEgressFifoRef`. This egress FIFO is mapped internally to a DMA channel based on the highest priority value assigned to the FIFO. In this mode, the arbiter first processes Channel 3 and then Channel 2, Channel 1 and Channel 0. If any channel does not have a frame to transmit, the weight of that channel gets reassigned to Channel3. If Channel 3 has no frames to transmit, the remaining weight is assigned to Channel 2 and so on.

  Note: The configured weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel.

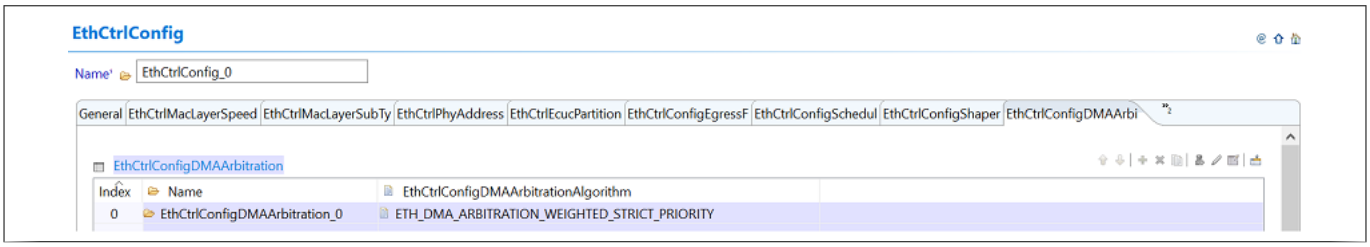


Figure 13 DMA Arbitration Algorithm - Weighted Strict Priority

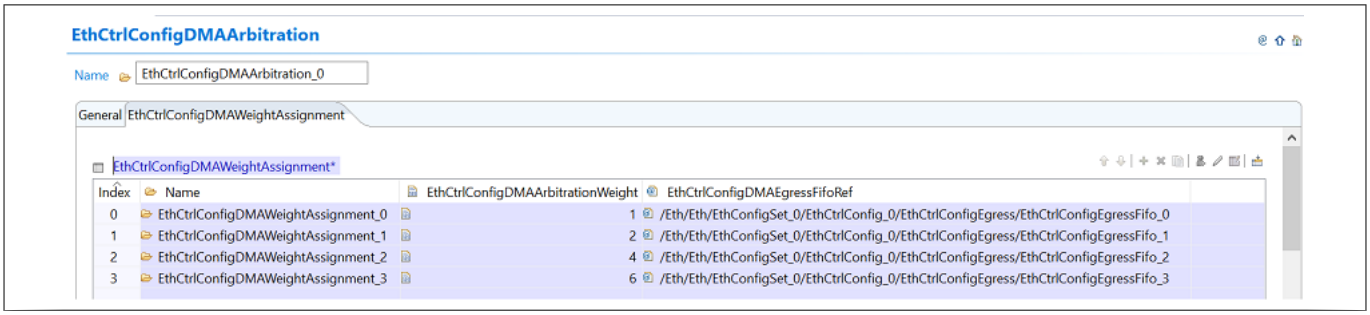


Figure 14 DMA Arbitration - Configuration of Weights to FIFOs

Initialization

The code sequence for initializing the ETH driver is as follows from the core (core 0) in which the controller is configured:

```
#include "Eth_17_GEthMacV2.h"
#include “Port.h”
#include “Mcu.h”
#include "McalLib.h"
#include “Irq.h”

uint32 CoreId;

/* MCU Initialization */
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock();

/* Port Initialization */
Port_Init(&Port_Config);

/* Ethernet driver Initialization */
Eth_17_GEthMacV2_Init(&Eth_17_GEthMacV2_Config);

/*Obtain the current core Id*/
CoreId = Mcal_GetCpuIndex();

/* Perform Ethernet transceiver initialization */
```

Setting controller mode

The code sequence to change the mode of the ETH controller 0 from ETH_MODE_DOWN state to ETH_MODE_ACTIVE is as follows:

```
Eth_17_GEthMacV2_SetControllerMode(0, ETH_MODE_ACTIVE);
```

Set physical address (MAC address)

The code sequence to set the MAC address of the ETH controller 0 is as follows:

```
Eth_17_GEthMacV2_SetPhysAddr(0, &MacAddress[0]);
```

Get physical address (MAC address)

The code sequence to get the MAC address of the ETH controller 0 is as follows:

```
Eth_17_GEthMacV2_GetPhysAddr(0, &MacAddressRead[0]);
```

Configure ETH transceiver (PHY) device

The code sequence to configure ETH Transceiver (PHY) by using `Eth_17_GEthMacV2_ReadMii` and `Eth_17_GEthMacV2_WriteMii` APIs is as follows:

```
uint16 phy_id1 = 0;
uint16 phy_id2 = 0;
uint16 reg_value = 0;

/* Read Device ID from Ethernet transceiver(PHY) */
Eth_17_GEthMacV2_ReadMii( 0, 1, MII_PHYSID1, &phy_id1);
Eth_17_GEthMacV2_ReadMii( 0, 1, MII_PHYSID2, &phy_id2);

/* Configure Link Speed and Duplex mode in Ethernet transceiver (PHY) */
reg_value = BMCR_SPEED100 | BMCR_FULLDPLX;
reset_status = Eth_17_GEthMacV2_WriteMii(0, 1, MII_BMCR, reg_value);
```

Transmit frame

The code sequence to transmit an ETH frame having Priority 0 in the polling mode is as follows:

```
/* Request the data buffer for frame Transmission */
Eth_17_GEthMacV2_ProvideTxBuffer(0, 0, BufIdxPtr, BufPtr, LenBytePtr);

/* Application Layer fills the buffer with frame data*/

/* Transmit already filled buffer using the BuffIdx */
Eth_17_GEthMacV2_Transmit(0, BufIdx, FrameType, TxConfirmation, LenByte, PhysAddrPtr);
 ..........................

/* Call TX confirmation API to check if the packet transmission is completed */
Eth_17_GEthMacV2_TxConfirmation(0);
```

Receive frame

The received ETH frames are transferred to the driver buffer automatically. In polling mode, the application should call `Eth_17_GEthMacV2_Receive` API to get the indication of successful reception of all the filled buffers. In the interrupt mode, the ETH driver will indicate successful reception of frames from receive ISR context. The code sequence to receive ETH frames for FIFO 0 of controller 0 in polling mode is as follows:

```
/* Poll for received frames */
Eth_17_GEthMacV2_Receive(0, 0, RxStatusPtr);
```

1.1.5 Key architectural considerations

1.1.5.1 ETH controller initialization sequence

The AUTOSAR specifications for the ETH and the ETH transceiver(PHY) drivers are conceived in a way that both shall work independently. As per AUTOSAR specification, the ETH controller and PHY initialization sequencing shall be as follows:

- Invoke `Eth_17_GEthMacV2_Init()`. This will initialize MAC controller and select PHY interface (MDIO and MII).
- Invoke `EthTrcv_Init()`. This will communicate to PHY using PHY interface (MDIO) and prepare the PHY to operate in the required mode.

Note: Since the PHY interface (MDIO) is managed by the ETH driver, `EthTrcv_Init()` will use `Eth_17_GEthMacV2_WriteMii()` and `Eth_17_GEthMacV2_ReadMii()` API services provided by ETH driver.

However, the said sequence does not fit with the Gigabit ETH MAC IP used in the TC3xx controller. Until the clock signal from the PHY is stable at MAC input, the software reset cannot be completed. Otherwise, this will cause unpredictable behavior. Hence the ETH controller initialization by ETH driver is split into two parts.

Part 1: The `Eth_17_GEthMacV2_Init()` API enables the module and prepares the MDIO interface.

Note: The `Eth_17_GEthMacV2_Init()` API will not initialize the controller because it is probable that the clock from the PHY is not yet available at the MAC input pin. The `Eth_17_GEthMacV2_Init()` API enables the clock to all the controllers allocated to the core from which it is being invoked and core initialization status is set to success. If there is a failure in initializing any one of the controllers, `ETH_E_ACCESS` production error is reported as PREFAILED for the failed controller and core initialization status is set to unsuccessful.

Part 2: The `Eth_17_GEthMacV2_SetControllerMode()` API performs the following actions:

- Flush the transmit and receive queues.
- Disables/ masks the module interrupts.
- If the configured mode is RGMII, set the mode to MII (`GETH_GPCTL.EPR` = 000b) and initialize the skew timing to 0. If the configured mode is not RGMII, select the PHY interface to either RMII or MII based on the configuration.
- Perform a kernel reset (expecting that `EthTrcv_Init()` is already called and the clock signal from the PHY is already available at the MAC input pin) and then wait for the necessary fSPB cycles before proceeding further.
- Set the PHY interface to the configured mode and also select the alternate inputs. In RGMII mode, set the Tx / Rx timing skew as per the configuration.
- Apply software reset to DMA and then complete the ETH controller initialization for the requested controller only (that is, only for the controller index which is passed as a parameter to the `Eth_17_GEthMacV2_SetControllerMode()` API).

  This action is performed only once when the `Eth_17_GEthMacV2_SetControllerMode()` API is called for the first time after `Eth_17_GEthMacV2_Init()` API.

The workaround mentioned in the HW errata is also considered in the initialization sequence followed in the ETH driver :

- GETH_TC.002 - Initialization of RGMII interface
- GETH_AI.H001 - Preparation for Software Reset

Note:

- *Since `Eth_17_GEthMacV2_SetControllerMode()` should be invoked before initiating any data transmit or receive operation, this change in the ETH controller initialization sequence does not affect the user application. Therefore, the calling sequences mentioned in AUTOSAR for the ETH driver remains the same.*
- *`Eth_17_GEthMacV2_SetControllerMode()` API performs the second stage of initialization only for the controller index passed as the input parameter. If the second stage of initialization for the controller is successful, then the controller initialization status is set to success.*
- *Runtime APIs reports the development error `ETH_17_GETHMACV2_E_UNINIT`, if indexed controller initialization or core initialization is unsuccessful.*

1.1.5.2 `Eth_17_GEthMacV2_SetControllerMode()` API implemented as synchronous

AUTOSAR specification mentions the `Eth_17_GEthMacV2_SetControllerMode()` API as asynchronous function. Since the ETH controller in TC3xx supports changing the controller mode instantaneously, the `Eth_17_GEthMacV2_SetControllerMode()` API is implemented as synchronous.

1.1.5.3 Development error checks added to avoid undefined behavior

The ETH driver performs the following error checks which are not explicitly mentioned in the product requirement. These error checks are introduced to avoid any undefined behavior from the ETH driver. If development error detection is enabled:

- The `Eth_17_GEthMacV2_SetControllerMode()` API checks the `CtrlMode` parameter for being valid. If the check fails, the API reports `ETH_17_GETHMACV2_E_INV_PARAM` development error and return `E_NOT_OK`.
- The `Eth_17_GEthMacV2_UpdatePhysAddrFilter()` API checks the `Action` parameter for being valid. If the check fails, the API reports `ETH_17_GETHMACV2_E_INV_PARAM` development error and return `E_NOT_OK`.
- The `Eth_17_GEthMacV2_Transmit()` API checks the `LenByte` parameter for being valid. If the value of `LenByte` is more than the data length that is granted by the `Eth_17_GEthMacV2_ProvideTxBuffer()`, the `Eth_17_GEthMacV2_Transmit()` API reports `ETH_17_GETHMACV2_E_INV_PARAM` development error and returns `E_NOT_OK`. If the value of `LenByte` is less than the data length that is granted by the `Eth_17_GEthMacV2_ProvideTxBuffer()` API, the `Eth_17_GEthMacV2_Transmit()` API proceeds with transmission of ETH packet of length `LenByte`.
- The APIs `Eth_17_GEthMacV2_EnableEgressTimeStamp()` and `Eth_17_GEthMacV2_GetEgressTimeStamp()` check the `BufIdx` parameter for being valid. This check is carried out to ensure that the value of `BufIdx` is within the total number of buffers configured for the egress FIFO and the value of `BufIdx` is same as the one allocated to application by ETH driver from the `Eth_17_GEthMacV2_ProvideTxBuffer()` API. If the aforementioned checks fail, the API reports the `ETH_17_GETHMACV2_E_INV_PARAM` development error.
- The `Eth_17_GEthMacV2_ProvideTxBuffer()` API checks the `Priority` parameter for being valid. If the check fails, the API reports `ETH_17_GETHMACV2_E_INV_PARAM` development error and return `E_NOT_OK`.
- The `Eth_17_GEthMacV2_Receive()` API checks the `FifoIdx` parameter for being valid. If the check fails, the API reports `ETH_17_GETHMACV2_E_INV_PARAM` development error.
- The transmit and receive interrupt handlers `Eth_17_GEthMacV2_TxDmaIrqHdlr()` and `Eth_17_GEthMacV2_RxDmaIrqHdlr()` report the `ETH_17_GETHMACV2_E_UNINIT` development error, if the indexed controller initialization or core initialization was unsuccessful. These interrupt handlers report the `ETH_17_GETHMACV2_E_INV_CTRL_IDX` development error, if the controller index is invalid or if the controller is not allocated to the current core. Also, these handlers check for the validity of the `DmaChnlIdx` parameter and if the check fails, `ETH_17_GETHMACV2_E_INV_PARAM` development error is reported.

1.1.5.4 Multicore support for ETH driver

ETH controllers are allowed to be configured to any of the cores as per the hardware availability. The APIs of the ETH driver can be classified as concurrent safe for different controllers.

Note: For certain TC3xx devices, more than one ETH controller can be present. In a multicore environment, each controller can be allocated to different cores or the same core based on the application requirement. If the `Eth_17_GEthMacV2_Init()` API is invoked from a core to which no controller is allocated then `ETH_17_GETHMACV2_E_CORE_NOT_CONFIGURED` development error is reported. If runtime APIs with controller index as input parameter are invoked from a core to which no controller or the requested controller is not allocated, then `ETH_17_GETHMACV2_E_INV_CTRL_IDX` development error is reported.

1.1.5.5 Reporting of ETH_E_ACCESS production error during timer update operation

In the ETH controller used in the TC3xx devices, any timer update operation requires a read back of the corresponding feedback bit to ensure that the requested operation has completed successfully. If the requested operation is not completed until a configured time, the `Eth_17_GEthMacV2_SetControllerMode()` API reports the `ETH_E_ACCESS` production error:

The reporting of `ETH_E_ACCESS` production error is performed to handle any hardware error while operating the available timer in ETH controller.

1.1.5.6 Releasing transmit buffers in `Eth_17_GEthMacV2_ProvideTxBuffer()` API

The `Eth_17_GEthMacV2_ProvideTxBuffer()` API performs a check whether any previously allocated buffer (buffer already provided to application in the previous invocation of the `Eth_17_GEthMacV2_ProvideTxBuffer()` API) has completed the transmission and the transmit confirmation is not requested by the application for that buffer index in the `Eth_17_GEthMacV2_Transmit()` API. In such a case, the `Eth_17_GEthMacV2_ProvideTxBuffer()` API will release those buffers.

1.1.5.7 Priority assignment for egress FIFO

The ETH controller supports only 4 DMA channels and hence it is allowed to configure upto 4 egress FIFOs as each FIFO is associated with a DMA channel. Each egress FIFO can have multiple priorities configured.

If only one egress FIFO is required by the application (upper layer), a single FIFO needs to be configured with either the `EthCtrlConfigEgressFifoPriorityAssignment` parameter not present (multiplicity of 0) or with the `EthCtrlConfigEgressFifoPriorityAssignment` parameter configured with all the priorities 0 to 7. The configuration of a single FIFO without `EthCtrlConfigEgressFifoPriorityAssignment parameter` is allowed, as the lower multiplicity of this parameter is 0 according to the AUTOSAR 4.4.0 SWS.

If multiple egress FIFOs are configured (upto 4), the priority assignment to these FIFOs should be mutually exclusive between them, that is, two or more FIFOs cannot have the same priority configured.

1.1.5.8 Granting of buffer based on requested priority

When more than one priority is configured for an egress FIFO, the ETH driver treats the multiple priorities assigned to the FIFO at the same level. There is no differentiation made among the priorities assigned to the same FIFO when `Eth_17_GEthMacV2_ProvideTxBuffer()` API is invoked. This implies, when `Eth_17_GEthMacV2_ProvideTxBuffer()` API is invoked for providing the Tx buffers for these priorities, ETH driver provides the Tx buffer index from the same Tx FIFO in sequence. It is not possible to handle the priorities within a FIFO at runtime.

As an example, consider that egress FIFO_A is assigned with multiple priorities 0, 1 and 2. The back to back requests to provide the Tx buffers with these priorities 0, 1 and 2 via the `Eth_17_GEthMacV2_ProvideTxBuffer()` API results in providing the transmit buffer resource in sequence from the same FIFO_A without having any differentiation among the priorities assigned to the same FIFO.

1.1.5.9 Index of the granted buffer provided by `Eth_17_GEthMacV2_ProvideTxBuffer()` API comprises of the buffer index and an internal index to refer FIFO

In case of multiple egress FIFOs configured, when a request for providing the access to a transmit buffer via `Eth_17_GEthMacV2_ProvideTxBuffer()` API is placed, the index to the granted buffer (output parameter `BufIdxPtr`) also contains the internal index used by the SW to refer to the egress FIFO. Thus, this granted buffer index can be a large value (within the 16 bit range) provided by the ETH driver. This buffer index is used in the subsequent request in `Eth_17_GEthMacV2_Transmit()` API for triggering the transmission of the filled buffer.

The `Eth_17_GEthMacV2_Transmit()` API uses this internal FIFO index reference present within the buffer index to identify the FIFO from which the buffer index was provided earlier to trigger the transmission.

To support this internal FIFO index reference, the upper 3 bits out of the 16 bits (the valid range of total buffer count) are used. So, the range of total number of buffers that can be configured within an egress FIFO (parameter `EthCtrlConfigEgressFifoBufTotal`) is limited to 13 bits, and therefore, the range is reduced to 0 - 8191.

1.1.5.10 Specific hardware features used for nominal operation of ETH driver

To implement the ETH driver as per the product requirements, the following hardware-specific features are used in ETH driver.

- Alternative pin selection:

  For the selected input pins of the ETH controller, the ETH controller hardware provides alternative hardware port pins and the ETH driver software should select one pin (which is used in hardware design) from available alternative input pin options. To meet this requirement, the ETH driver provides the following pre-compile configuration parameters under `EthCtrlConfig` container.

  `EthMdioAlternateInput`, `EthRxclkInput`, `EthRxErrMIIInput`, `EthCarrierSenseMIIInput`, `EthRecDataValidMIIInput`, `EthTxClockMIIInput`, `EthCollisionMII`, `EthRefClkRMIIInput`, `EthCRSDVRMIIInput`, `EthReceiveData0Input`, `EthReceiveData1Input`, `EthReceiveData2Input` and `EthReceiveData3Input`.
- Value of gigabit ETH MAC Kernel frequency:

  The ETH controller uses the timer which is part of the ETH controller for implementing the requirement of global time support APIs. To configure this internal timer, the ETH driver needs to read the value of Gigabit ETH MAC Kernel frequency which is configured in the MCU driver. To extract this value, the ETH driver provides a pre-compile configuration parameter `EthOperationFrequency` in the `EthGeneral` container.
- Value of system peripheral bus frequency:

  To configure MDIO interface clock, the ETH driver needs to read the value of the system peripheral bus frequency which is configured in the MCU driver. To extract this value, the ETH driver provides a pre-compile configuration parameter `EthPeripheralBusClock` in the `EthGeneral` container.
- FIFO space and DMA channel:

  The ETH controller supports 4 transmit DMA channels and 4 receive DMA channels.

  Due to this limitation of maximum 4 DMA channels support, the number of FIFOs that can be configured on the egress and ingress side is limited to 4. Each egress/ ingress FIFO is associated with a DMA channel.

  On the MAC transaction layer, the ETH controller supports Tx FIFO size of 4 kB and Rx FIFO size of 8 kB. This FIFO space is used to construct multiple queues (upto 4 Tx and 4 Rx queues). The configured egress/ ingress FIFOs are mapped to one of these queues.

  (i) The Rx FIFO size of 8 kB is allocated among the Rx queues based on the configured ingress FIFO buffer length (`EthCtrlConfigIngressFifoBufLenByte`) and also on the ingress FIFO buffer count (`EthCtrlConfigIngressFifoBufTotal`).

  (ii) The Tx FIFO size of 4 kB is allocated among the Tx queues based on the configured egress FIFO buffer length (`EthCtrlConfigEgressFifoBufLenByte`) and also on the egress FIFO buffer count (`EthCtrlConfigEgressFifoBufTotal`).

  (iii) The egress FIFOs should be configured such that the sum of the individual buffer lengths (converted to the multiple of 256 bytes) should not exceed 4 kB. For example, if 3 egress FIFOs are configured with the parameter `EthCtrlConfigEgressFifoBufLenByte` configured to a value of 1500 bytes, then the total size exceeds 4 kB and this cannot be supported. When the sum of the configured individual buffer lengths

(converted to the multiple of 256 bytes) exceeds 4 kB, the code generation tool detects this condition and reports this as an error.

- Configuration of transmit and receive timing skew in RGMII mode:

  `EthSkewTxClockDelay` and `EthSkewRxClockDelay` configuration parameters under the `EthCtrlConfig` container are added to configure the transmit and receive clock delay for skew timing. This is applicable only in the RGMII mode. In the MII and RMII modes, both parameters are not active.
- Configuration of MDIO clock frequency:

  The ETH driver provides the `EthMDCClockFrequency` configuration parameter to configure MDC clock value for the Ethernet controller. The minimum value of the `EthMDCClockFrequency` parameter is 2.5 MHz. If the configured value is 2.5 MHz, the actual MDC clock generated will be between 1.0 to 2.5 MHz based on the value of system peripheral bus frequency (fSPB) configured in the MCU driver. The minimum value for this parameter is fixed at 2.5 MHz considering that, any IEEE standard PHY device will support MDC clock value up to 2.5 MHz.

1.1.5.11 Additional configuration parameters provided for Credit Based Shaper

In the AUTOSAR specification, for the Credit Based Shaper (CBS) feature, only a single configuration parameter `EthCtrlConfigShaperIdleSlope` is mentioned which is used to configure the idle slope. However, it is required to configure other necessary parameters for CBS feature at the egress side, namely, HiCredit and LoCredit parameters. In order to provide this support, the additional configuration parameters `EthCtrlConfigShaperHiCredit` and `EthCtrlConfigShaperLoCredit` are added.

1.1.5.12 Configuration parameter for MTL scheduling algorithm selection

The GETH MAC hardware supports two algorithms, namely Strict Priority and Weighted Round Robin (WRR), for transmit scheduling among the queues at the MTL layer. Hence, a configuration parameter `EthCtrlConfigSchedulerAlgorithm` is added which is used to select one of the supported transmit scheduling algorithms.

In case the selected algorithm is WRR, the parameter `EthCtrlConfigSchedulerPredecessorOrder` is used to define the weight for the queue associated to a FIFO.

In case of strict priority, the parameter `EthCtrlConfigSchedulerPredecessorOrder` represents the order of the scheduler entities. When credit shaper is not configured, the egress FIFO which is referenced with the lowest value of `EthCtrlConfigSchedulerPredecessorOrder` is the FIFO that is treated as having the lowest transmission priority order by the HW. On the other hand, the FIFO referenced with the highest value of `EthCtrlConfigSchedulerPredecessorOrder` is considered by the HW as having the highest transmission priority order.

With credit shapers configured, the `EthCtrlConfigSchedulerPredecessorOrder` for the FIFOs without shapers should have lower values than the FIFOs with shapers configured.

1.1.5.13 Configuration parameter to choose the FIFO to which untagged Rx packets are routed

When multiple ingress FIFOs are configured with different priorities, the routing of the received frames to these FIFOs is done depending on the PCP field value of VLAN tag. However, there is no parameter provided in AUTOSAR to choose the ingress FIFO to which the received frames that do not have the VLAN tag (untagged frames) should be routed to. Therefore, the parameter `EthCtrlConfigIngressUntaggedPktsFifoRef` is added to enable the routing of the untagged frames to the chosen FIFO. One of the configured ingress FIFO needs to be referenced by this parameter during configuration.

All the untagged packets (that is, packets without a VLAN tag), including PTP untagged packets and AV untagged control packets, are routed to the chosen ingress FIFO.

1.1.5.14 Configuration parameters added for DMA arbitration algorithm

The GETH MAC hardware supports up to 4 transmit DMA channels. The DMA arbiter helps in the arbitration of all the paths on the transmit channels. This arbitration is performed based on one of the algorithms: Fixed Priority, Weighted Strict Priority (WSP) or Weighted Round Robin (WRR).

The configuration parameter `EthCtrlConfigDMAArbitrationAlgorithm` is added to select one of the DMA algorithms for arbitration among the Tx DMA channels. In case, either WSP or WRR algorithm is selected, the weight needs to configured using the parameter `EthCtrlConfigDMAArbitrationWeight` for the corresponding egress FIFO referenced by the parameter `EthCtrlConfigDMAEgressFifoRef`. This egress FIFO is mapped internally to a DMA channel. The configured weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel.

Among the four supported Tx DMA channels, Tx DMA channel 3 has the highest priority, whereas channel 0 has the lowest priority. Each configured egress FIFO is mapped to one of the DMA channels based on the scheduler order configured in the parameter `EthCtrlConfigSchedulerPredecessorOrder` (or based on the FIFO priorities, in case weights are configured in `EthCtrlConfigSchedulerPredecessorOrder`). In Fixed Priority, the Tx DMA channel 3 is assigned to the FIFO with the highest `EthCtrlConfigSchedulerPredecessorOrder` value, Tx DMA channel 2 to the FIFO with the next highest `EthCtrlConfigSchedulerPredecessorOrder` value and so on.

1.1.5.15 Supported counter values are provided irrespective of either half-duplex or full-duplex mode

Irrespective of the configured duplex mode, either half-duplex or full-duplex mode, the counter values that are supported by the GETH MAC hardware are always reported by the appropriate counter/ statistics APIs.

In the AUTOSAR specification, it is mentioned that the APIs that are used to extract the counter values need to return the maximum possible value (0xFFFFFFFF) for the counters which are not supported by the hardware. Out of the supported counters, there are few counters that are incremented only in half-duplex mode. However, since these counters are supported by the hardware, the counter values that are returned by the APIs will not be the maximum possible value (0xFFFFFFFF) in half-duplex mode, but instead the API will read the register values and return them. So the maximum possible value (0xFFFFFFFF) returned for a counter implies that the counter is not supported by the hardware.

1.1.5.16 Supported counter values are provided irrespective of either half-duplex or full-duplex mode

Irrespective of the configured duplex mode, either half-duplex or full-duplex mode, the counter values that are supported by the GETH MAC hardware are always reported by the appropriate counter/ statistics APIs.

In the AUTOSAR specification, it is mentioned that the APIs that are used to extract the counter values need to return the maximum possible value (0xFFFFFFFF) for the counters which are not supported by the hardware. Out of the supported counters, there are few counters that are incremented only in half-duplex mode. However, since these counters are supported by the hardware, the counter values that are returned by the APIs will not be the maximum possible value (0xFFFFFFFF) in half-duplex mode, but instead the API will read the register values and return them. So the maximum possible value (0xFFFFFFFF) returned for a counter implies that the counter is not supported by the hardware.

1.1.5.17 Limitation on the support of only one ingress FIFO when switch management support functionality is enabled

When the switch management support functionality is enabled (configuration parameter `EthSwtManagementSupportApi` is true), only one ingress FIFO is supported by the ETH driver.

The GETH MAC hardware supports routing of the received packets to upto 4 ingress FIFOs based on the priority(PCP) field of the VLAN tag. This HW feature of routing of received packets to the ingress FIFOs is used to support the AUTOSAR requirements on FIFO priority assignment, wherein the ingress FIFOs configured with one or more priorities receive the packets having these priority values in the PCP field.

If the switch management support functionality is enabled, the ETH driver invokes the Eth Switch (EthSwt) driver APIs during transmission and reception of the packets. With this functionality enabled, it is expected that the switch management related information is present within the Ethernet packet; and the position of this information resides after the address field (MAC source address) and before the EthType field within the Ethernet packet. Due to this, the VLAN tag position along with the data field is shifted by the number of bytes of switch management information. As the VLAN field is shifted, the routing of packets to the ingress FIFOs cannot be performed by the hardware based on the priority field.

Thus, multiple ingress FIFOs cannot be supported together with switch management support feature.

1.1.5.18 Switch management related checks performed by `Eth_17_GEthMacV2_ProvideTxBuffer()` API

With the switch management support enabled (configuration parameter `EthSwtManagementSupportApi` is true), when the `Eth_17_GEthMacV2_ProvideTxBuffer()` API is called, this API invokes two functions of the Ethernet switch driver (EthSwt) `EthSwt_EthTxAdaptBufferLength()` and `EthSwt_EthTxPrepareFrame()`. These EthSwt APIs modify the parameters passed to it. Hence, it is necessary to check that these parameters are correctly modified by these EthSwt APIs.

- The `Eth_17_GEthMacV2_ProvideTxBuffer()` API performs validity check on the modified length (output parameter `LengthPtr`) to determine whether the modified length is less than or equal to the configured buffer length. If the modified length is greater than the configured length, `Eth_17_GEthMacV2_ProvideTxBuffer()` API notifies this condition by returning `BUFREQ_E_OVFL`.
- The `Eth_17_GEthMacV2_ProvideTxBuffer()` API performs validity check on the modified data pointer (output parameter `DataPtr`) to determine whether the pointer is updated correctly. If the modified pointer is not moved exactly by the modified length, the `Eth_17_GEthMacV2_ProvideTxBuffer()` API notifies this condition by returning `BUFREQ_E_NOT_OK`.

1.1.5.19 Switch management related checks performed by `Eth_17_GEthMacV2_Transmit()` API

With the switch management support enabled (configuration parameter `EthSwtManagementSupportApi` is true), when the `Eth_17_GEthMacV2_Transmit()` API is called, this API invokes two functions of the Ethernet switch driver (EthSwt) `EthSwt_EthTxProcessFrame()` and `EthSwt_EthTxFinishedIndication()`. The `EthSwt_EthTxProcessFrame()` API modifies the parameters passed to it. Hence, it is necessary to check that these parameters are correctly modified by the EthSwt APIs.

- The `Eth_17_GEthMacV2_Transmit()` API performs validity check on the modified length (output parameter `LengthPtr`) to determine whether the modified length is less than or equal to the configured buffer length. If the modified length is greater than the configured buffer length, the `Eth_17_GEthMacV2_Transmit()` API returns `E_NOT_OK`.
- The `Eth_17_GEthMacV2_Transmit()` API performs validity check on the modified data pointer (output parameter `DataPtr`) to determine whether the pointer is updated correctly. If the modified pointer is not moved exactly by the modified length, the `Eth_17_GEthMacV2_Transmit()` API returns `E_NOT_OK`.

1.1.5.20 Switch management related checks performed by `Eth_17_GEthMacV2_Receive()` API / receive interrupt handler

With the switch management support enabled (configuration parameter `EthSwtManagementSupportApi` is true), when the `Eth_17_GEthMacV2_Receive()` API is called in polling mode, this API invokes two functions of the Ethernet switch driver (EthSwt), `EthSwt_EthRxProcessFrame()` and `EthSwt_EthRxFinishedIndication()`, in case of successful reception of Ethernet frames. In case of interrupt mode, these APIs are invoked within the receive interrupt handler after successful reception of an ethernet frame. The `EthSwt_EthRxProcessFrame()` API modifies the parameters passed to it. Hence, it is necessary to check that these parameters are correctly modified by the EthSwt APIs.

- The `Eth_17_GEthMacV2_Receive()` API / Rx interrupt handler performs validity check on the modified length (output parameter `LengthPtr`) to determine whether the modified length is less than or equal to the configured buffer length. If the modified length is greater than the configured length, the `Eth_17_GEthMacV2_Receive()` API/ Rx interrupt handler detects this condition, aborts further processing and `ETH_NOT_RECEIVED` is returned in `RxStatusPtr`.
- The `Eth_17_GEthMacV2_Receive()` API/ RX interrupt handler performs validity check on the modified data pointer (output parameter `DataPtr`) to determine whether the pointer is updated correctly. If the modified pointer is not moved exactly by the modified length, the `Eth_17_GEthMacV2_Receive()` API detects this condition, aborts further processing and `ETH_NOT_RECEIVED` is returned in `RxStatusPtr`.

1.1.5.21 Hardware dependent counter values supported by `Eth_17_GEthMacV2_GetCounterValues()` API

The hardware dependent counter values `HwDepCtr0`, `HwDepCtr1`, `HwDepCtr2` and `HwDepCtr3` that are part of the structure `Eth_CounterType` are returned by the `Eth_17_GEthMacV2_GetCounterValues()` API with the below supported counter values:

- `HwDepCtr0` - Number of good VLAN packets transmitted
- `HwDepCtr1` - Number of good IPv4 packets received
- `HwDepCtr2` - Number of good IP datagrams received with a good TCP payload
- `HwDepCtr3` - Number of good IP datagrams received with a good UDP payload

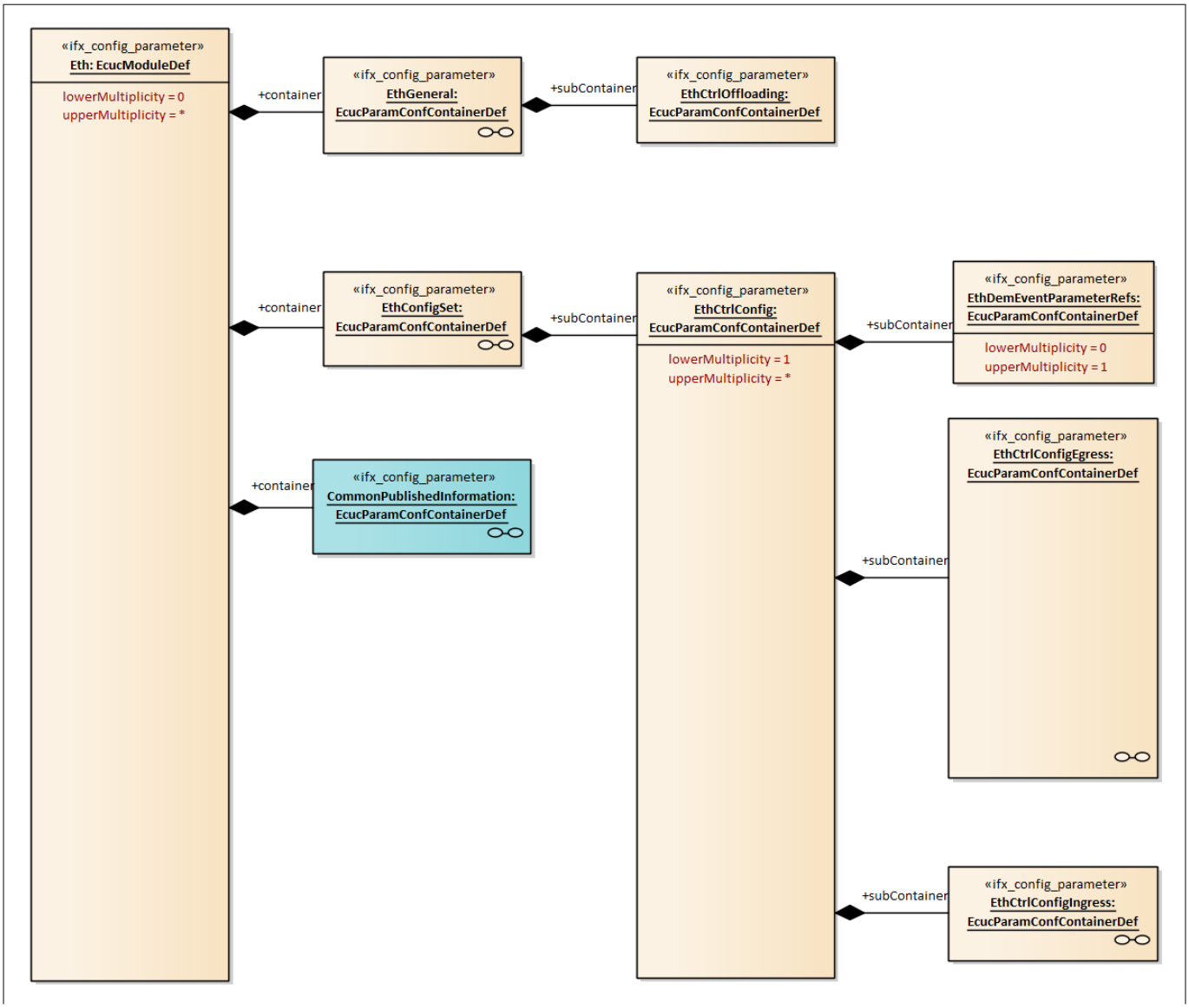
1.2 Assumptions of Use (AoU)

There are no AoUs for the ETH driver.

1.3 Reference information

1.3.1 Configuration interfaces

This section details the configuration container hierarchy along with their configuration parameters.
Supported configuration variant: Post-Build


«ifx_config_parameter»
Eth: EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = *
+container
«ifx_config_parameter»
EthGeneral:
EcucParamConfContainerDef
+subContainer
«ifx_config_parameter»
EthCtrlOffloading:
EcucParamConfContainerDef
+container
«ifx_config_parameter»
EthConfigSet:
EcucParamConfContainerDef
+subContainer
«ifx_config_parameter»
EthCtrlConfig:
EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+subContainer
«ifx_config_parameter»
EthDemEventParameterRefs:
EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
«ifx_config_parameter»
CommonPublishedInformation:
EcucParamConfContainerDef
+subContainer
«ifx_config_parameter»
EthCtrlConfigEgress:
EcucParamConfContainerDef
+subContainer
«ifx_config_parameter»
EthCtrlConfigIngress:
EcucParamConfContainerDef

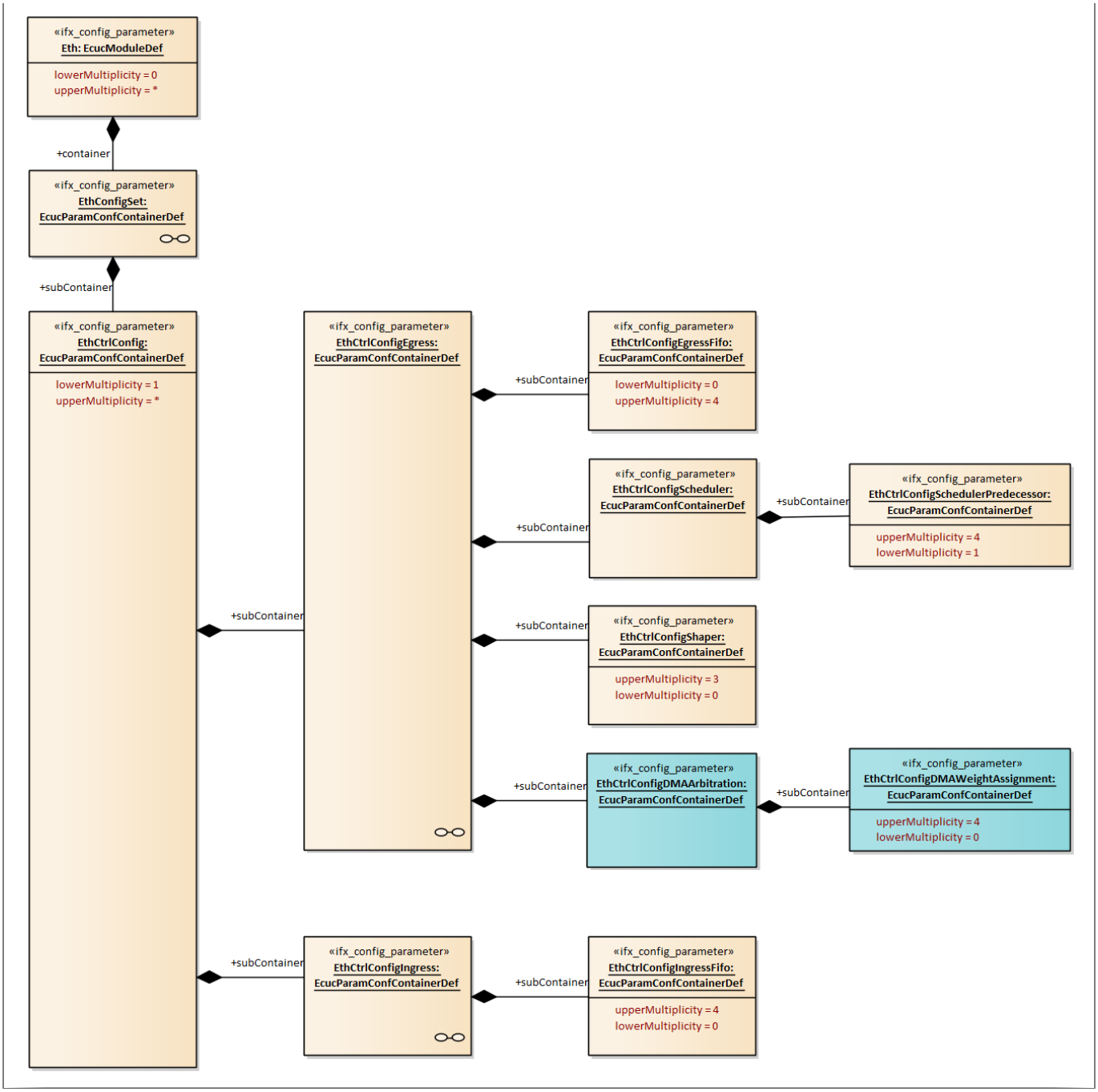


Figure 15 Container hierarchy along with their configuration parameters

1.3.1.1 Container: CommonPublishedInformation

Container contains the common published information of the ETH driver

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.1.1 ArMajorVersion

Table 5 Specification for ArMajorVersion

Name	`ArMajorVersion`		
Description	Provides the major version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.1.2 ArMinorVersion

Table 6 Specification for ArMinorVersion

Name	`ArMinorVersion`		
Description	Provides the minor version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.1.3 ArPatchVersion

Table 7 Specification for ArPatchVersion

Name	`ArPatchVersion`		
Description	Provides the patch version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef

(table continues...)

Table 7 (continued) Specification for ArPatchVersion			
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.1.4 ModuleId

Table 8 Specification for ModuleId			
Name	`ModuleId`		
Description	Provides the module Id.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	88		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.1.5 Release

Table 9 Specification for Release			
Name	`Release`		
Description	Indicates the TC3xx device derivative used for the implementation.		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	As per the hardware derivative		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 9 (continued) Specification for Release

Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.1.6 SwMajorVersion

Table 10 Specification for SwMajorVersion

Name	`SwMajorVersion`		
Description	Provides the major version of the software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the software version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.1.7 SwMinorVersion

Table 11 Specification for SwMinorVersion

Name	`SwMinorVersion`		
Description	Provides the minor version of the software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the software version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

(table continues...)

Table 11 (continued) Specification for SwMinorVersion

Autosar Version	Applicable for Autosar version 4.4.0.

1.3.1.1.8 SwPatchVersion

Table 12 Specification for SwPatchVersion

Name	`SwPatchVersion`		
Description	Provides the patch version of the software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the software version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.1.9 VendorApiInfix

Table 13 Specification for VendorApiInfix

Name	`VendorApiInfix`		
Description	Provides the VendorApiInfix.		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	GEthMacV2		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.1.10 VendorId

Table 14 Specification for VendorId

Name	VendorId		
Description	Provides the vendor Id		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.2 Container: Eth

Configuration of the individual ETH controller.
Post-Build Variant Multiplicity: -
Multiplicity Configuration Class: -

1.3.1.3 Container: EthConfigSet

This container contains the configuration parameters and sub-containers of the AUTOSAR ETH module.

Note: The multiplicity of this container is 1..1

Post-Build Variant Multiplicity: -
Multiplicity Configuration Class: -

1.3.1.4 Container: EthCtrlConfig

Configuration of the individual ETH controller.

Note: The multiplicity of EthCtrlConfig is device dependent. It is 1 to maximum number of controllers available.
Post-Build Variant Multiplicity: FALSE
Multiplicity Configuration Class: Pre-Compile

1.3.1.4.1 EthCRSDVRMIIInput

Table 15 Specification for EthCRSDVRMIIInput

Name	EthCRSDVRMIIInput

(table continues...)

Table 15 (continued) Specification for EthCRSDVRMIIInput

Description	Selects one of the four supported pins for the ETH carrier sense/data valid combi-signal for RMII. The availability of the port pins is dependent on the micro-controller package. *Note 1: This parameter is valid only if the EthPhyInterface parameter is selected as RMII.* *Note 2: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* *User must choose a suitable alternate port pin available for the device.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.2 EthCarrierSenseMIIInput

Table 16 Specification for EthCarrierSenseMIIInput

Name	`EthCarrierSenseMIIInput`		
Description	Selects one of the two supported pins for the ETH carrier sense MII. The availability of the port pins is dependent on the micro-controller package. *Note 1: This parameter is valid only if the EthPhyInterface parameter is selected as MII.* *Note 2: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* User must choose a suitable alternate port pin available for the device.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

(table continues...)

Table 16 (continued) Specification for EthCarrierSenseMIIInput

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.3 EthCollisionMII

Table 17 Specification for EthCollisionMII

Name	`EthCollisionMII`		
Description	Selects one of the four supported pins for collision for MII. The availability of the port pins is dependent on the micro-controller package. *Note 1: This parameter is valid only if the EthPhyInterface parameter is selected as MII.* *Note 2: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* User must choose a suitable alternate port pin available for the device.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.4 EthCtrlEcucPartitionRef

Table 18 Specification for EthCtrlEcucPartitionRef

Name	`EthCtrlEcucPartitionRef`

(table continues...)

Table 18 (continued) Specification for EthCtrlEcucPartitionRef			
Description	Maps the Ethernet controller to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the Ethernet driver is mapped to. *Note: Parameter support is added only for AUTOSAR schema compliance. This parameter is not used in code generation logic and hence, is made non-editable.*		
Multiplicity	0..1	**Type**	EcucReferenceDef
Range	Reference to Node: EcucPartition		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.5 EthCtrlEnableCrcStripping

Table 19 Specification for EthCtrlEnableCrcStripping			
Name	`EthCtrlEnableCrcStripping`		
Description	Includes or excludes the length of the checksum in the received frame length reported to the upper layer. *Note 1: If this parameter is enabled (that is set to TRUE), the ETH driver excludes the length of the checksum in the received frame length.* *Note 2: The default value of this parameter is kept as FALSE. Therefore, the application will receive complete ETH frame including checksum field.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.6 EthCtrlEnableMii

Table 20 Specification for EthCtrlEnableMii

Name	`EthCtrlEnableMii`		
Description	Enables/disables MII/ RMII/ RGMII-based APIs for transceiver access. *Note: The optional APIs are disabled by default to minimize the executable code size. If there are more than one controller for the selected device then the value of this parameter must be same across the controllers.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.7 EthCtrlEnableRxInterrupt

Table 21 Specification for EthCtrlEnableRxInterrupt

Name	`EthCtrlEnableRxInterrupt`		
Description	Enables/disables the receive interrupt. If the receive interrupt is disabled, the reception will work in the polling mode. *Note: The default value of this parameter is kept as false and, therefore, the ETH driver can be functional without configuring the interrupt module by default.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		

(table continues...)

Table 21	(continued) Specification for EthCtrlEnableRxInterrupt
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.1.4.8 EthCtrlEnableTxInterrupt

Table 22 Specification for EthCtrlEnableTxInterrupt

Name	`EthCtrlEnableTxInterrupt`		
Description	Enables / Disables transmit interrupt. If it is disabled, transmission will work in polling mode. *Note: The default value of this parameter kept as false and hence the ETH driver can be functional without configuring the interrupt module in default.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.9 EthCtrlIdx

Table 23 Specification for EthCtrlIdx

Name	`EthCtrlIdx`		
Description	Specifies the instance ID of the configured controller. This value is assigned to the symbolic name derived from the short name of the EthCtrlConfig. *Note: EthCtrlIdx varies from 0 to maximum number of controllers available.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	ECU

(table continues...)

Table 23 (continued) Specification for EthCtrlIdx	
Dependency	-
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.1.4.10 EthCtrlMacLayerSpeed

Table 24 Specification for EthCtrlMacLayerSpeed

Name	`EthCtrlMacLayerSpeed`		
Description	Defines the baud rate of the MAC layer. *Note: The default value of this parameter is set to 100 Mbps which is available across different interfaces - MII, RMII, RGMII (10 Mbps speed is rarely used).*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ETH_MAC_LAYER_SPEED_100M: MAC layer speed is 100 Mbps ETH_MAC_LAYER_SPEED_10M: MAC layer speed is 10 Mbps ETH_MAC_LAYER_SPEED_1G: MAC layer speed is 1 Gbps		
Default value	ETH_MAC_LAYER_SPEED_100M		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.11 EthCtrlMacLayerSubType

Table 25 Specification for EthCtrlMacLayerSubType

Name	`EthCtrlMacLayerSubType`		
Description	Defines the MAC layer subtype of the interface. Only the REDUCED and STANDARD subtypes are supported by the HW. *Note 1: The default subtype value is set to REDUCED as this subtype is available across all HW variants supporting Ethernet.* *Note 2: In the AUTOSAR specification, the description of this parameter is incorrectly mentioned as the MAC layer subtype of the switch port.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	REDUCED: REDUCED subtype is to be selected for RMII / RGMII interface STANDARD: STANDARD subtype is to be selected for MII interface.		
Default value	REDUCED		

(table continues...)

Table 25 (continued) Specification for EthCtrlMacLayerSubType

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.12 EthCtrlMacLayerType

Table 26 Specification for EthCtrlMacLayerType

Name	`EthCtrlMacLayerType`		
Description	This parameter defines the MAC layer type of the ETH controller. *Note: The default type is set to ETH_MAC_LAYER_TYPE_XMII which supports either RMII or MII. The sublayer type is set to a default value of REDUCED which together with this parameter makes the default interface as RMII that is available across all HW variants supporting Ethernet.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ETH_MAC_LAYER_TYPE_XGMII: MAC layer interface (data) bandwidth class - 1 Gbps *Note: Only RGMII interface is supported by HW in this bandwidth class (Supported speeds: 10 Mbps, 100 Mbps, 1 Gbps)* ETH_MAC_LAYER_TYPE_XMII: MAC layer interface (data) bandwidth class - 100 Mbps *Note: Only MII and RMII interfaces are supported by HW in this bandwidth class (Supported speeds: 10 Mbps, 100 Mbps)*		
Default value	ETH_MAC_LAYER_TYPE_XMII		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.13 EthCtrlPhyAddress

Table 27 Specification for EthCtrlPhyAddress

Name	`EthCtrlPhyAddress`

(table continues...)

Table 27 (continued) Specification for EthCtrlPhyAddress

Description	Specifies the unique 48-bit physical address (MAC address) of the ETH controller in network byte order. Regular Expression: [0-9a-fA-F]{2}[[:-][0-9a-fA-F]{2}]{5} Allowed characters are [a-f, A-F, 0-9] and each pair should be separated by symbols: or -. For example, 00:A0:C9:14:C8:29 *Note 1: As per AUTOSAR, the multiplicity is 0-1. However, in the ETH driver the multiplicity is implemented as 1-1. Therefore, Post-Build variant multiplicity is also false.* *Note 2: The default value of this parameter is kept as Infineon's ID to match the Vendor ID of MAC. If there are more than one ETH controller, then the same default value is retained. It is the responsibility of the user to provide a unique MAC address as per the application needs (if the MAC address is not unique then an error is prompted to the user to provide a unique MAC address).*		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	00:03:19:00:00:01		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.14 EthMDCClockFrequency

Table 28 Specification for EthMDCClockFrequency

Name	`EthMDCClockFrequency`

(table continues...)

Table 28 (continued) Specification for EthMDCClockFrequency

Description	MDC clock frequency in Hz. As specified in IEEE 802.3, the maximum value of MDC clock is 2.5 MHz. However, there are PHY devices, which support higher than 2.5 MHz. In AURIX controller, the MDC clock is derived from fSPB and the different values of MDC clock frequency is achieved by programming the clock divider value in the Ethernet controller. The exact MDC clock value, that can be generated is dependent on fSPB value and the clock divider values programmable in Ethernet controller. For example, when fSPB clock is of 100 MHz frequency and the parameter EthMDCClockFrequency is configured as 12500000 Hz (12.5 MHz), then MDC clock of 12.5 MHz is achieved by programming the divider value 8 in the Ethernet controller. *Note 1: Based on the fSPB value and possible clock divider values, it may not be possible to generate exact MDC clock value as configured in the parameter EthMDCClockFrequency. If so, the next lowest possible value is chosen. For example, if fSPB clock is of 100 MHz and the parameter EthMDCClockFrequency is configured as 20000000 Hz (20 MHz), MDC clock of 16.66 MHz is generated by programming the divider value as 6 in the Ethernet controller.* *Note 2: To generate the MDC clock frequency, the possible divider values available (for the supported fSPB values) in the Ethernet controller are 4, 6, 8, 10, 12, 14, 16, 18, 26 and 42.* *Note 3: To allow higher frequencies (more than 2.5 MHz), maximum possible value of this parameter is the quotient of the configured SPB frequency and minimum divider value in the Ethernet Controller.* *Note 4: The minimum (default) value for the parameter EthMDCClockFrequency is 2.5 MHz. If the configured value is 2.5 MHz, the actual MDC clock generated will be between 1.0 to 2.5 MHz based on the value of fSPB clock configured. The minimum value for this parameter is fixed at 2.5 MHz considering that, any IEEE standard PHY device shall support MDC clock value up to 2.5 MHz.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	2500000 - 25000000		
Default value	2500000		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.15 EthMdioAlternateInput

Table 29 Specification for EthMdioAlternateInput

Name	`EthMdioAlternateInput`

(table continues...)

Table 29 (continued) Specification for EthMdioAlternateInput			
Description	Selects one of the four supported pins for the MDIO signal. The availability of port pins is dependent on the micro-controller package. *Note: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* User must choose a suitable alternate port pin available for the device.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.16 EthOpMode

Table 30 Specification for EthOpMode			
Name	`EthOpMode`		
Description	Specifies the mode of operation (FULLDUPLEX/HALFDUPLEX).		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	FULLDUPLEX: Full-duplex mode HALFDUPLEX: Half-duplex mode		
Default value	FULLDUPLEX		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.17 EthRecDataValidMIIInput

Table 31 Specification for EthRecDataValidMIIInput

Name	`EthRecDataValidMIIInput`		
Description	Selects one of the four supported pins for the ETH Receive Data Valid MII. The availability of the port pins is dependent on the micro-controller package. *Note 1: This parameter is valid only if the EthPhyInterface parameter is selected as MII.* *Note 2: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* User must choose a suitable alternate port pin available for the device.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.18 EthReceiveData0Input

Table 32 Specification for EthReceiveData0Input

Name	`EthReceiveData0Input`		
Description	Selects one of the four supported pins for receive data 0 for MII, RMII and RGMII (RGMII can use RXD0A only). The availability of the port pins is dependent on the micro-controller package. *Note: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* User must choose a suitable alternate port pin available for the device.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		

(table continues...)

Table 32 (continued) Specification for EthReceiveData0Input

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.19 EthReceiveData1Input

Table 33 Specification for EthReceiveData1Input

Name	`EthReceiveData1Input`		
Description	Selects one of the four supported pins for receive data 1 for MII, RMII and RGMII (RGMII can use RXD1A only). The availability of the port pins is dependent on the micro-controller package. *Note: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* User must choose a suitable alternate port pin available for the device.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.20 EthReceiveData2Input

Table 34 Specification for EthReceiveData2Input

Name	`EthReceiveData2Input`

(table continues...)

Table 34 (continued) Specification for EthReceiveData2Input

Description	Selects one of the four supported pins for receive data 2 for MII and RGMII (RGMII can use RXD2A only). The availability of the port pins is dependent on the micro-controller package. *Note 1: This parameter is invalid only if the EthPhyInterface parameter is selected as RMII.* *Note 2: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* User must choose a suitable alternate port pin available for the device.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.21 EthReceiveData3Input

Table 35 Specification for EthReceiveData3Input

Name	`EthReceiveData3Input`		
Description	Selects one of the four supported pins for receive data 3 for MII and RGMII (RGMII can use RXD3A only). The availability of the port pins is dependent on the micro-controller package. *Note 1: This parameter is invalid only if the EthPhyInterface parameter is selected as RMII.* *Note 2: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* User must choose a suitable alternate port pin available for the device.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		

(table continues...)

Table 35 (continued) Specification for EthReceiveData3Input

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.22 EthRefClkRMIIInput

Table 36 Specification for EthRefClkRMIIInput

Name	`EthRefClkRMIIInput`		
Description	Selects one of the four supported pins for reference clock input for RMII. The availability of the port pins is dependent on the micro-controller package. *Note 1: This parameter is valid only if the EthPhyInterface parameter is selected as RMII.* *Note 2: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* User must choose a suitable alternate port pin available for the device.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.23 EthRxClkInput

Table 37 Specification for EthRxClkInput

Name	`EthRxClkInput`

(table continues...)

Table 37 (continued) Specification for EthRxClkInput

Description	Selects one of the four supported pins for the ETH receive clock for MII and RGMII (RGMII can use RXCLKA only). The availability of the port pins is dependent on the micro-controller package *Note 1: This parameter is valid only if the EthPhyInterface parameter is selected as MII.* *Note 2: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* User must choose a suitable alternate port pin available for the device.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.24 EthRxErrMIIInput

Table 38 Specification for EthRxErrMIIInput

Name	`EthRxErrMIIInput`		
Description	Selects one of the four supported pins for the ETH Receive Error MII. The availability of the port pins is dependent on the micro-controller package. *Note 1: This parameter is valid only if the EthPhyInterface parameter is selected as MII.* *Note 2: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* User must choose a suitable alternate port pin available for the device.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		

(table continues...)

Table 38 (continued) Specification for EthRxErrMIIInput

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.25 EthSkewRxClockDelay

Table 39 Specification for EthSkewRxClockDelay

Name	`EthSkewRxClockDelay`		
Description	Specifies the receive clock delay in the RGMII mode for Transmit Skew Timing. *Note: The minimum value is kept as the default value for this parameter and, therefore, by default the clock delay is 0.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 15		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.26 EthSkewTxClockDelay

Table 40 Specification for EthSkewTxClockDelay

Name	`EthSkewTxClockDelay`		
Description	Specifies the transmit clock delay in RGMII mode for transmit skew timing. *Note: Minimum value is kept as default value for this parameter and hence in default the clock delay is zero.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 15		
Default value	0		

(table continues...)

Table 40 (continued) Specification for EthSkewTxClockDelay

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.4.27 EthTxClockMIIInput

Table 41 Specification for EthTxClockMIIInput

Name	`EthTxClockMIIInput`		
Description	Selects one of the four supported pins for the transmit clock input for MII. The availability of the port pins is dependent on the micro-controller package. *Note 1: This parameter is valid only if the EthPhyInterface parameter is selected as MII.* *Note 2: The default option is ALTx_SELECT_NONE where, x indicates the alternate input select. ALTx_SELECT_NONE indicates that no port pin is selected.* User must choose a suitable alternate port pin available for the device.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ALTx_SELECT_NONE: Indicates that no port pin is selected (x in ALTx_SELECT_NONE indicates the alternate input select) ALTx_SELECT_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlMacLayerSubType, EthCtrlMacLayerType		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.5 Container: EthCtrlConfigDMAArbitration

Contains the configuration of the arbitration for the transmit DMA channels.

Note: The multiplicity of this container is 1..1

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.5.1 EthCtrlConfigDMAArbitrationAlgorithm

Table 42 Specification for EthCtrlConfigDMAArbitrationAlgorithm

Name	`EthCtrlConfigDMAArbitrationAlgorithm`		
Description	This parameter configures the arbitration algorithm for the Tx DMA channels. The GETH MAC hardware supports up to 4 transmit DMA channels and each configured egress FIFO is mapped to one of the DMA channels (based on the scheduler configuration). The Tx DMA channel 3 has the highest priority and channel 0 has the lowest priority. The DMA arbiter helps in the arbitration of all the paths on the transmit channels based on one of the configured algorithms: Fixed Priority, Weighted Strict Priority (WSP) or Weighted Round Robin (WRR). In addition, when Weighted Strict Priority or Weighted Round Robin algorithm is chosen, the necessary weights needs to be configured using the EthCtrlConfigDMAArbitrationWeight parameter for the corresponding configured egress FIFOs. *Note: The default value of this parameter is set to ETH_DMA_ARBITRATION_FIXED_PRIORITY, so that, the ethernet packets from the egress FIFO configured with highest scheduler order are selected for transmission on priority.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ETH_DMA_ARBITRATION_FIXED_PRIORITY: Transmit DMA arbitration algorithm is Fixed Priority. In this mode, channel 0 has the lowest priority and channel 3 has the highest priority. ETH_DMA_ARBITRATION_WEIGHTED_ROUND_ROBIN: Transmit DMA arbitration algorithm is Weighted Round Robin. In this mode, the channel with the highest programmed weight is selected first, and then the channel with the next highest weight, and so on. ETH_DMA_ARBITRATION_WEIGHTED_STRICT_PRIORITY: Transmit DMA arbitrationalgorithm is Weighted Strict Priority. In this mode, the Tx DMA arbiter first processes Channel 3 and then Channel 2, Channel 1, and Channel 0. If any channel does not have a frame to transmit, the weight of that channel gets reassigned to Channel 3. If Channel 3 has no frames to transmit, the remaining weight is assigned to Channel 2 and so on.		
Default value	ETH_DMA_ARBITRATION_FIXED_PRIORITY		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.6 Container: EthCtrlConfigDMAWeightAssignment

Configuration of the Tx DMA channel weights mapped to the egress FIFOs.

This container needs to be added and configured only if either weighted round robin or weighted strict priority arbitration algorithm is selected in the parameter EthCtrlConfigDMAArbitrationAlgorithm.

Note 1: The channel weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel.

Note 2: The multiplicity of this container is 0..4

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

1.3.1.6.1 EthCtrlConfigDMAArbitrationWeight

Table 43 Specification for EthCtrlConfigDMAArbitrationWeight

Name	`EthCtrlConfigDMAArbitrationWeight`		
Description	This parameter specifies the weight configured for a DMA channel that is mapped to an egress FIFO.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - 8		
Default value	1		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlConfigDMAArbitrationAlgorithm		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.6.2 EthCtrlConfigDMAEgressFifoRef

Table 44 Specification for EthCtrlConfigDMAEgressFifoRef

Name	`EthCtrlConfigDMAEgressFifoRef`		
Description	Reference to the configured egress FIFO.		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: EthCtrlConfigEgressFifo		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthCtrlConfigDMAArbitrationAlgorithm		

(table continues...)

Table 44 (continued) Specification for EthCtrlConfigDMAEgressFifoRef

Autosar Version	Applicable for Autosar version 4.4.0.

1.3.1.7 Container: EthCtrlConfigEgress

Configuration of egress behavior of Ethernet controller

Note: The multiplicity of this container is 1..1

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.7.1 EthCtrlConfigEgressLastSchedulerRef

Table 45 Specification for EthCtrlConfigEgressLastSchedulerRef

Name	`EthCtrlConfigEgressLastSchedulerRef`		
Description	Reference to the scheduler which is the last in the egress structure. As the GETH MAC hardware provides only one transmit scheduler at the MTL layer, this parameter refers to the only configured EthCtrlConfigScheduler container.		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: EthCtrlConfigScheduler		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.8 Container: EthCtrlConfigEgressFifo

Represents a FIFO at the egress side

Note: The multiplicity of this container is 0..4

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.8.1 EthCtrlConfigEgressFifoBufLenByte

Table 46 Specification for EthCtrlConfigEgressFifoBufLenByte

Name	`EthCtrlConfigEgressFifoBufLenByte`

(table continues...)

Table 46 (continued) Specification for EthCtrlConfigEgressFifoBufLenByte

Description	Length of FIFO elements in bytes. This parameter determines the length of each buffer within the egress FIFO. This parameter limits the maximum transmit buffer length (frame length) in bytes for the corresponding egress FIFO. This configured length includes ETH frame header and frame checksum (Total 22 bytes including 4 bytes of optional VLAN Tag). *Note 1: During configuration, the buffer length should be: Header + Payload data + CRC.* *Note 2: The buffer length configured should be greater than 18 bytes as the length of 18 bytes are consumed by Header and CRC fields of an Ethernet packet.* *Note 3: Maximum length of one ETH frame packet is 1522 bytes (1500 (Payload) + 4 (VLAN Tag - Optional) + 14 (Header) + 4 (CRC) = 1522).* *Note 4: The FIFO buffer length needs to be configured such that the sum of individual egress FIFO buffer lengths (converted to a multiple of 256 bytes) should not exceed 4096 bytes, as the HW supports a total Tx queue size of 4 kB. As an example, if 3 egress FIFOs are configured with the parameter EthCtrlConfigEgressFifoBufLenByte configured to a value of 1500 bytes, then the total size exceeds 4 kB and thus, this cannot be supported.* *Note 5: The minimum valid size of one ETH frame is 64 bytes, the default value for this parameter is kept as 64.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 1522		
Default value	64		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.8.2 EthCtrlConfigEgressFifoBufTotal

Table 47 Specification for EthCtrlConfigEgressFifoBufTotal

Name	`EthCtrlConfigEgressFifoBufTotal`

(table continues...)

Table 47 (continued) Specification for EthCtrlConfigEgressFifoBufTotal

Description	Count of the number of buffers within the egress FIFO. *Note 1: The total buffer size in the RAM that is reserved by the ETH driver for the corresponding FIFO is calculated as EthCtrlConfigEgressFifoBufLenByte * EthCtrlConfigEgressFifoBufTotal.* *Note 2: By default, the number of buffers reserved within a FIFO is set to 4 as per the recommended minimum DMA descriptor ring length.* *Note 3: The range of this parameter is changed to 0 - 8191. The range is reduced to 13 bits, because the upper 3 bits (out of the 16 bits for total buffer range) is utilized by the software to store the internal index to refer the FIFO from which the buffer is allocated by Eth_17_GEthMacV2_ProvideTxBuffer() API.* *Note 4: The number of buffers should not be configured to the value 0. If a FIFO is not required for transmission, the FIFO container can be removed from configuration.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 8191		
Default value	4		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.8.3 EthCtrlConfigEgressFifoIdx

Table 48 Specification for EthCtrlConfigEgressFifoIdx

Name	`EthCtrlConfigEgressFifoIdx`		
Description	Index of the egress FIFO. The maximum limit on the number of the egress FIFOs is limited to 4, as only 4 queues (which are mapped to FIFOs) are supported by the GETH MAC hardware. *Note 1: The default index value is set to 0, as the indexing starts from the value 0.* *Note 2: The FIFO index starts from 0 and should be configured sequentially.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 3		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-

(table continues...)

Table 48 (continued) Specification for EthCtrlConfigEgressFifoIdx			
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.8.4 EthCtrlConfigEgressFifoPriorityAssignment

Table 49 Specification for EthCtrlConfigEgressFifoPriorityAssignment			
Name	`EthCtrlConfigEgressFifoPriorityAssignment`		
Description	Message priority assignment for the egress FIFO. *Note 1: The default priority is assigned a value of 0 (lowest priority value). The priority value of 0 indicates lowest priority and a value of 7 indicates highest priority.* *Note 2: The API Eth_17_GEthMacV2_ProvideTxBuffer() provides a transmit buffer from the egress FIFO configured with the priority assigned in this field. Hence, two or more FIFOs should not be assigned with the same priority.*		
Multiplicity	0..8	**Type**	EcucIntegerParamDef
Range	0 - 7		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.9 Container: EthCtrlConfigIngress

Configuration of ingress behavior of Ethernet controller.

Note: The multiplicity of this container is 1..1

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.9.1 EthCtrlConfigIngressUntaggedPktsFifoRef

Table 50 Specification for EthCtrlConfigIngressUntaggedPktsFifoRef	
Name	`EthCtrlConfigIngressUntaggedPktsFifoRef`
Description	Reference to the ingress FIFO to which all the untagged Rx packets are to be routed. *Note: All the untagged packets (that is, packets without a VLAN tag), including PTP untagged packets and AV untagged control packets, are routed to the chosen ingress FIFO.*

(table continues...)

Table 50 (continued) Specification for EthCtrlConfigIngressUntaggedPktsFifoRef

Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: EthCtrlConfigIngressFifo		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.10 Container: EthCtrlConfigIngressFifo

Represents a FIFO at the ingress side.

Note: The multiplicity of this container is 0..4

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.10.1 EthCtrlConfigIngressFifoBufLenByte

Table 51 Specification for EthCtrlConfigIngressFifoBufLenByte

Name	`EthCtrlConfigIngressFifoBufLenByte`		
Description	Length of FIFO elements in bytes. This parameter determines the length of each buffer within the ingress FIFO. *Note 1: The minimum valid size of an ETH frame is 64 bytes, the default value for this parameter is kept as 64.* *Note 2: The buffer length configured should be greater than 18 bytes as the length of 18 bytes are consumed by Header and CRC fields of an Ethernet packet.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 1522		
Default value	64		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.10.2 EthCtrlConfigIngressFifoBufTotal

Table 52 Specification for EthCtrlConfigIngressFifoBufTotal

Name	`EthCtrlConfigIngressFifoBufTotal`		
Description	Configures the number of buffers within the ingress FIFO. *Note 1: The total buffer size in the RAM that is reserved by the ETH driver for the corresponding ingress FIFO is calculated as EthCtrlConfigIngressFifoBufTotal * EthCtrlConfigIngressFifoBufLenByte.* *Note 2: By default, the number of buffers reserved within a FIFO is set to 4 as per the recommended minimum DMA descriptor ring length.* *Note 3: The number of buffers should not be configured to the value 0. If a FIFO is not required for reception, the FIFO container can be removed from configuration.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	4		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.10.3 EthCtrlConfigIngressFifoIdx

Table 53 Specification for EthCtrlConfigIngressFifoIdx

Name	`EthCtrlConfigIngressFifoIdx`		
Description	Index of the ingress FIFO. The maximum limit on the number of the ingress FIFOs is limited to 4, as only 4 queues (which are mapped to FIFOs) are supported by the GETH MAC hardware. *Note 1: The default index value is set to 0, as the indexing starts from the value 0.* *Note 2: The FIFO index starts from 0 and should be configured sequentially.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 3		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL

(table continues...)

Table 53	(continued) Specification for EthCtrlConfigIngressFifoIdx
Dependency	-
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.1.10.4 EthCtrlConfigIngressFifoPriorityAssignment

Table 54 Specification for EthCtrlConfigIngressFifoPriorityAssignment

Name	`EthCtrlConfigIngressFifoPriorityAssignment`		
Description	Message priority assignment for the ingress FIFO. *Note 1: The default priority is assigned a value of 0 (lowest priority value). The priority value of 0 indicates lowest priority and a value of 7 indicates highest priority.* *Note 2: The GETH MAC hardware routes the received tagged packets to the ingress FIFO based on the priority assigned in this field. Hence, two or more FIFOs should not be assigned with the same priority.* *Note 3: When configuring priorities for the ingress FIFOs, all the priorities within the range 0 to 7 should be configured across the FIFOs. This is required as the GETH MAC hardware does not drop the packets with non-configured priorities.*		
Multiplicity	0..8	**Type**	EcucIntegerParamDef
Range	0 - 7		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.11 Container: EthCtrlConfigScheduler

Represents a Scheduler on the egress side.

Note: The multiplicity of this container is 1..1

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.11.1 EthCtrlConfigSchedulerAlgorithm

Table 55 Specification for EthCtrlConfigSchedulerAlgorithm

Name	`EthCtrlConfigSchedulerAlgorithm`

(table continues...)

Table 55 (continued) Specification for EthCtrlConfigSchedulerAlgorithm

Description	Defines the algorithm for Transmit scheduling at the queue level. *Note 1: The default value of this parameter is set to ETH_SCHEDULER_STRICT_PRIORITY so that the ethernet frames from the higher priority queues are selected for transmission on priority.* *Note 2: This parameter configures the Tx scheduling algorithm at the MTL layer of the GETH MAC hardware.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ETH_SCHEDULER_STRICT_PRIORITY: Transmit scheduling algorithm is strict priority. In strict priority, queue 3 has highest priority and queue 0 has the lowest priority. ETH_SCHEDULER_WEIGHTED_ROUND_ROBIN: Transmit scheduling algorithm is weighted round robin.		
Default value	ETH_SCHEDULER_STRICT_PRIORITY		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.12 Container: EthCtrlConfigSchedulerPredecessor

Defines an ordered list of schedule entities for this scheduler.

Note 1: The multiplicity of this container is 1..4

Note 2: The upper multiplicity of this container is limited to 4 as only four FIFOs are supported.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

1.3.1.12.1 EthCtrlConfigSchedulerPredecessorOrder

Table 56 Specification for EthCtrlConfigSchedulerPredecessorOrder

Name	`EthCtrlConfigSchedulerPredecessorOrder`

(table continues...)

Table 56 (continued) Specification for EthCtrlConfigSchedulerPredecessorOrder

Description	Defines the order of the scheduler entities, in case the parameter EthCtrlConfigSchedulerAlgorithm is selected as ETH_SCHEDULER_STRICT_PRIORITY. The FIFO that is referenced (via Shaper or directly) by the scheduler with the highest value of EthCtrlConfigSchedulerPredecessorOrder is the FIFO with the highest priority. If the parameter EthCtrlConfigSchedulerAlgorithm is chosen as ETH_SCHEDULER_WEIGHTED_ROUND_ROBIN, this parameter defines the weight to be configured in terms of the number of frames. *Note 1: The default value is set to 0 as the scheduling order starts from 0 in case of strict priority which is set as the default scheduling algorithm.* *Note 2: For strict priority scheduling, this parameter ranges from 0 - 3, and for WRR scheduling, this parameter ranges from 1 - 100. The range is limited to value 100, as the maximum weight that can be configured within the HW register is 100.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 100		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	EthCtrlConfigSchedulerAlgorithm		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.12.2 EthCtrlConfigSchedulerPredecessorRef

Table 57 Specification for EthCtrlConfigSchedulerPredecessorRef

Name	`EthCtrlConfigSchedulerPredecessorRef`		
Description	Choice reference to the scheduler entities. *Note: As per the AUTOSAR Ethernet driver SWS, this parameter can be configured to have a reference to either EthCtrlConfigEgressFifo , EthCtrlConfigScheduler or EthCtrlConfigShaper. However, since multiple schedulers are not supported within the GETH MAC hardware, the reference to EthCtrlConfigScheduler should not be configured.*		
Multiplicity	1..1	**Type**	EcucChoiceReference Def
Range	Reference to Node: EthCtrlConfigEgressFifo, EthCtrlConfigShaper, EthCtrlConfigScheduler		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-

(table continues...)

Table 57 (continued) Specification for EthCtrlConfigSchedulerPredecessorRef

Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.13 Container: EthCtrlConfigShaper

Represents a shaper on the egress side.

It is required to have this container only when credit based shaper needs to be configured on the egress side.

Note: The multiplicity of this container is 0..3

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

1.3.1.13.1 EthCtrlConfigShaperHiCredit

Table 58 Specification for EthCtrlConfigShaperHiCredit

Name	`EthCtrlConfigShaperHiCredit`		
Description	The maximum value in bits that can be accumulated in the credit value for the credit-based shaper algorithm.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 131072		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.13.2 EthCtrlConfigShaperIdleSlope

Table 59 Specification for EthCtrlConfigShaperIdleSlope

Name	`EthCtrlConfigShaperIdleSlope`		
Description	Defines the increase of credit in bits per second for the AVB shaper. *Note: This parameter should always be configured when the EthCtrlConfigShaper container is configured.*		
Multiplicity	0..1	**Type**	EcucIntegerParamDef
Range	0 - 1000000000		

(table continues...)

Table 59 (continued) Specification for EthCtrlConfigShaperIdleSlope			
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	FALSE
Value configuration class	Post-Build	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.13.3 EthCtrlConfigShaperLoCredit

Table 60 Specification for EthCtrlConfigShaperLoCredit			
Name	`EthCtrlConfigShaperLoCredit`		
Description	The minimum value in bits that can be accumulated in the credit value for the credit-based shaper algorithm.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 131072		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.13.4 EthCtrlConfigShaperPredecessorFifoRef

Table 61 Specification for EthCtrlConfigShaperPredecessorFifoRef			
Name	`EthCtrlConfigShaperPredecessorFifoRef`		
Description	Reference to the FIFO to which the configured config shaper parameter applies.		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: EthCtrlConfigEgressFifo		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

(table continues...)

Table 61 (continued) Specification for EthCtrlConfigShaperPredecessorFifoRef

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.14 Container: EthCtrlOffloading

Configuration of hardware checksum offloading features.

Note 1: Individual enabling of hardware offload functionality for CRC checksum (for IPv4, UDP, TCP, ICMP frames) is not possible due to hardware limitation. Enabling of any one of the below configuration parameters EthCtrlEnableOffloadChecksumIPV4, EthCtrlEnableOffloadChecksumUDP, EthCtrlEnableOffloadChecksumTCP, EthCtrlEnableOffloadChecksumICMP would enable the checksum offload functionality for IPv4, UDP, TCP and ICMP. This is a deviation from the AUTOSAR requirements.

Note 2: The multiplicity of this container is 1..1

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.14.1 EthCtrlEnableOffloadChecksumICMP

Table 62 Specification for EthCtrlEnableOffloadChecksumICMP

Name	`EthCtrlEnableOffloadChecksumICMP`		
Description	Enables/disables checksum offloading of IPv4, TCP, UDP and ICMP frames for both transmission (that is, calculating and inserting checksum in the transmitted frames at the hardware level) and reception (that is, checking for checksum mismatch at the hardware level for the received frames). *Note: This is a deviation from the AUTOSAR requirement. The GETH MAC hardware supports either enabling or disabling of the checksum offload feature for all types of packets, but not for individual packet types.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL

(table continues...)

Table 62	(continued) Specification for EthCtrlEnableOffloadChecksumICMP
Dependency	EthCtrlEnableOffloadChecksumTCP, EthCtrlEnableOffloadChecksumIPv4, EthCtrlEnableOffloadChecksumUDP
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.1.14.2 EthCtrlEnableOffloadChecksumIPv4

Table 63 Specification for EthCtrlEnableOffloadChecksumIPv4

Name	`EthCtrlEnableOffloadChecksumIPv4`		
Description	Enables/ disables checksum offloading of IPv4, TCP, UDP, ICMP frames for both transmission (that is, calculating and inserting checksum in the transmitted frames at the hardware level) and reception (that is, checking for checksum mismatch at the hardware level for the received frames). *Note: This is a deviation from the AUTOSAR requirement. The GETH MAC hardware supports either enabling or disabling of the checksum offload feature for all types of packets, but not for individual packet types.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	EthCtrlEnableOffloadChecksumICMP, EthCtrlEnableOffloadChecksumTCP, EthCtrlEnableOffloadChecksumUDP		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.14.3 EthCtrlEnableOffloadChecksumTCP

Table 64 Specification for EthCtrlEnableOffloadChecksumTCP

Name	`EthCtrlEnableOffloadChecksumTCP`
Description	Enables/ disables checksum offloading of IPv4, TCP, UDP and ICMP frames for both transmission (that is, calculating and inserting checksum in the transmitted frames at the hardware level) and reception (that is, checking for checksum mismatch at the hardware level for the received frames). *Note: This is a deviation from the AUTOSAR requirement. The GETH MAC hardware supports either enabling or disabling of the checksum offload feature for all types of packets, but not for individual packet types.*

(table continues...)

Table 64 (continued) Specification for EthCtrlEnableOffloadChecksumTCP

Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	EthCtrlEnableOffloadChecksumICMP, EthCtrlEnableOffloadChecksumIPv4, EthCtrlEnableOffloadChecksumUDP		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.14.4 EthCtrlEnableOffloadChecksumUDP

Table 65 Specification for EthCtrlEnableOffloadChecksumUDP

Name	`EthCtrlEnableOffloadChecksumUDP`		
Description	Enables/ disables checksum offloading of IPv4, TCP, UDP, ICMP frames for both transmission (that is, calculating and inserting checksum in the transmitted frames at the hardware level) and reception (that is, checking for checksum mismatch at the hardware level for the received frames). *Note: This is a deviation from the AUTOSAR requirement. The GETH MAC hardware supports either enabling or disabling of the checksum offload feature for all types of packets, but not for individual packet types.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	EthCtrlEnableOffloadChecksumICMP, EthCtrlEnableOffloadChecksumIPv4, EthCtrlEnableOffloadChecksumTCP		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.15 Container: EthDemEventParameterRefs

This is a container for the references to the DemEventParameter elements, which are invoked using the Mcal_Wrapper_Dem_SetEventStatus() API in case the corresponding errors occur. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor-specific error references.

Note: The multiplicity of this container is 0..1

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

1.3.1.15.1 ETH_E_ACCESS

Table 66 Specification for ETH_E_ACCESS

Name	ETH_E_ACCESS		
Description	Provides reference to the DemEventParameter, which is issued when the controller access failed error occurs.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.15.2 ETH_E_ALIGNMENT

Table 67 Specification for ETH_E_ALIGNMENT

Name	ETH_E_ALIGNMENT		
Description	Provides reference to the DemEventParameter, which is issued when the alignment error occurs.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE

(table continues...)

Table 67 (continued) Specification for ETH_E_ALIGNMENT

Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.15.3 ETH_E_CRC

Table 68 Specification for ETH_E_CRC

Name	ETH_E_CRC		
Description	Provides reference to the DemEventParameter, which is issued when the CRC failure error occurs.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.15.4 ETH_E_LATECOLLISION

Table 69 Specification for ETH_E_LATECOLLISION

Name	ETH_E_LATECOLLISION		
Description	Provides reference to the DemEventParameter, which is issued when the late frame collision event occurs. *Note: This is applicable only in the half-duplex mode.*		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE

(table continues...)

Table 69 (continued) Specification for ETH_E_LATECOLLISION

Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.15.5 ETH_E_MULTIPLECOLLISION

Table 70 Specification for ETH_E_MULTIPLECOLLISION

Name	`ETH_E_MULTIPLECOLLISION`		
Description	Provides reference to the DemEventParameter, which is issued when the multiple frame collision event occurs. *Note: This is applicable only in the half-duplex mode.*		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.15.6 ETH_E_OVERSIZEFRAME

Table 71 Specification for ETH_E_OVERSIZEFRAME

Name	`ETH_E_OVERSIZEFRAME`		
Description	Provides reference to the DemEventParameter, which is issued when the over-sized frame error occurs.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE

(table continues...)

Table 71 (continued) Specification for ETH_E_OVERSIZEFRAME

Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.15.7 ETH_E_RX_FRAMES_LOST

Table 72 Specification for ETH_E_RX_FRAMES_LOST

Name	ETH_E_RX_FRAMES_LOST		
Description	Provides reference to the DemEventParameter, which is issued when the receive frames lost error occurs.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.15.8 ETH_E_SINGLECOLLISION

Table 73 Specification for ETH_E_SINGLECOLLISION

Name	ETH_E_SINGLECOLLISION		
Description	Provides reference to the DemEventParameter, which is issued when the single frame collision event occurs. *Note: This is applicable only in the half-duplex mode.*		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE

(table continues...)

Table 73 (continued) Specification for ETH_E_SINGLECOLLISION

Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.15.9 ETH_E_UNDERSIZEFRAME

Table 74 Specification for ETH_E_UNDERSIZEFRAME

Name	`ETH_E_UNDERSIZEFRAME`		
Description	Provides reference to the DemEventParameter, which is issued when the under-sized frame error occurs.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16 Container: EthGeneral

General configuration of the ETH driver module.

Note: The multiplicity of this container is 1..1

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.16.1 EthDevErrorDetect

Table 75 Specification for EthDevErrorDetect

Name	`EthDevErrorDetect`
Description	Switches the development error detection and notification on or off. true : detection and notification is enabled false : detection and notification is disabled

(table continues...)

Table 75 (continued) Specification for EthDevErrorDetect

Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.2 EthDmaSwResetWaitCycle

Table 76 Specification for EthDmaSwResetWaitCycle

Name	`EthDmaSwResetWaitCycle`		
Description	This parameter specifies the number of fSPB wait cycles to wait after the DMA software reset. *Note: The HW manual specifies that wait time should be at least 4 fSPB cycles, hence the default value of this parameter is set to 4.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	4 - 255		
Default value	4		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.3 EthEcucPartitionRef

Table 77 Specification for EthEcucPartitionRef

Name	`EthEcucPartitionRef`

(table continues...)

Table 77 (continued) Specification for EthEcucPartitionRef

Description	Maps the Ethernet driver to zero or multiple ECUC partitions to make the modules API available in this partition. The Ethernet driver will operate as an independent instance in each of the partitions. *Note: Parameter support is added only for AUTOSAR schema compliance. This parameter is not used in code generation logic and hence, is made non-editable.*		
Multiplicity	0..*	**Type**	EcucReferenceDef
Range	Reference to Node: EcucPartition		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.4 EthGetDropCountApi

Table 78 Specification for EthGetDropCountApi

Name	`EthGetDropCountApi`		
Description	Enables or disables the Eth_17_GEthMacV2_GetCounterValues() API. *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.5 EthGetEtherStatsApi

Table 79 Specification for EthGetEtherStatsApi

Name	`EthGetEtherStatsApi`		
Description	Enables or disables the Eth_17_GEthMacV2_GetRxStats() API. *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.6 EthGetTxErrorCounterValuesApi

Table 80 Specification for EthGetTxErrorCounterValuesApi

Name	`EthGetTxErrorCounterValuesApi`		
Description	Enables or disables the Eth_17_GEthMacV2_GetTxErrorCounterValues() API. *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.7 EthGetTxStatsApi

Table 81 Specification for EthGetTxStatsApi

Name	`EthGetTxStatsApi`		
Description	Enables or disables the Eth_17_GEthMacV2_GetTxStats() API. *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.8 EthGlobalTimeSupport

Table 82 Specification for EthGlobalTimeSupport

Name	`EthGlobalTimeSupport`		
Description	Enables or disables the following GlobalTime APIs: Eth_17_GEthMacV2_GetCurrentTime() Eth_17_GEthMacV2_EnableEgressTimeStamp() Eth_17_GEthMacV2_GetEgressTimeStamp() Eth_17_GEthMacV2_GetIngressTimeStamp() *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL

(table continues...)

Table 82	(continued) Specification for EthGlobalTimeSupport
Dependency	-
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.1.16.9 EthIndex

Table 83	Specification for EthIndex		
Name	`EthIndex`		
Description	Specifies the ID of this module instance. If only one instance is present it should have an ID value of 0. *Note: Since there is only one instance present in the most of the TC3xx controllers (except TC377_EX_ED device), the default value is kept as 0.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.10 EthInitApiMode

Table 84	Specification for EthInitApiMode		
Name	`EthInitApiMode`		
Description	Defines the mode in which the Eth_17_GEthMacV2_Init() API is used. *Note: The ETH driver accesses the SFRs, therefore, it is more efficient to operate the ETH driver in the Supervisor mode. Hence, the default mode of operation is Supervisor.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ETH_MCAL_SUPERVISOR: Operating mode is Supervisory ETH_MCAL_USER1: Operating mode is User-1		
Default value	ETH_MCAL_SUPERVISOR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 84 (continued) Specification for EthInitApiMode

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthRuntimeApiMode		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.11 EthMainFunctionPeriod

Table 85 Specification for EthMainFunctionPeriod

Name	`EthMainFunctionPeriod`		
Description	Specifies the period of main function Eth_17_GEthMacV2_MainFunction in seconds. ETH driver does not require this information but the BSW scheduler uses this information. *Note 1: The range of this parameter is limited to a value 10.0 which is a deviation from the AUTOSAR SWS, in which the upper limit is specified as infinite.* *Note 2: The value of 0 should not be configured for this parameter as the AUTOSAR SWS excludes the value 0 from the configurable range.*		
Multiplicity	1..1	**Type**	EcucFloatParamDef
Range	0 - 10		
Default value	0.005		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.12 EthMaxCtrlsSupported

Table 86 Specification for EthMaxCtrlsSupported

Name	`EthMaxCtrlsSupported`		
Description	Limits the total number of supported controllers. This parameter is disabled for configuration because the ETH controllers available depends on the device variant.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - maximum controllers available for the device		
Default value	maximum controllers available for the device		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 86 (continued) Specification for EthMaxCtrlsSupported

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.13 EthMultiCoreErrorDetect

Table 87 Specification for EthMultiCoreErrorDetect

Name	`EthMultiCoreErrorDetect`		
Description	This parameter enables or disables the Multi core related error detection and reporting. It is applicable only when development error detection is enabled. *Note: By default, the value of this parameter is set to FALSE since it is dependent on EthDevErrorDetect parameter. The parameter is disabled for single core devices.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthDevErrorDetect		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.14 EthOperationFrequency

Table 88 Specification for EthOperationFrequency

Name	`EthOperationFrequency`		
Description	Contains reference to the fGETH (basic frequency for the Gigabit ETH kernel) frequency value contained in the MCU module (in the Mcu/McuModuleConfiguration/McuClockSettingConfig/McuClockReferencePointConfig container). This parameter is used to calculate the frequency ratio value required for configuration of the system timer. *Note: Since the dependent container is user configurable, the default value of this parameter is kept as NULL.*		
Multiplicity	1..1	**Type**	EcucReferenceDef

(table continues...)

Table 88 (continued) Specification for EthOperationFrequency

Range	Reference to Node: McuClockReferencePointConfig		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.15 EthPeripheralBusClock

Table 89 Specification for EthPeripheralBusClock

Name	`EthPeripheralBusClock`		
Description	Contains reference to the fSPB (System Peripheral Bus) frequency value contained in the MCU module (in the Mcu/McuModuleConfiguration/McuClockSettingConfig/ McuClockReferencePointConfig container). This parameter is used to program the clock value of the MDIO interface. *Note: Since the dependent container is user configurable, the default value of this parameter is kept as NULL.*		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: McuClockReferencePointConfig		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.16 EthRuntimeApiMode

Table 90 Specification for EthRuntimeApiMode

Name	`EthRuntimeApiMode`
Description	Provides the mode in which the Runtime API is used. *Note: Since the ETH driver accesses the SFRs, it is more efficient to operate the ETH driver in the Supervisor mode. Therefore, the default mode of operation is supervisor.*

(table continues...)

Table 90 (continued) Specification for EthRuntimeApiMode

Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ETH_MCAL_SUPERVISOR: Operating mode used is Supervisory. The access to supervisor mode registers is abstracted via McalLib module. McalLib routes the call to OS APIs. ETH_MCAL_USER1: Operating mode used is User-1.		
Default value	ETH_MCAL_SUPERVISOR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.17 EthTimeoutCount

Table 91 Specification for EthTimeoutCount

Name	`EthTimeoutCount`		
Description	Specifies the maximum waiting time in nanoseconds for hardware timeout errors. *Note: A nominal value of 100 microsecond is chosen as the default value assuming that the hardware should not take more than this time for an operation to complete.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	100 - 4294967295		
Default value	100000		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.16.18 EthVersionInfoApi

Table 92 Specification for EthVersionInfoApi

Name	`EthVersionInfoApi`

(table continues...)

Table 92 (continued) Specification for EthVersionInfoApi

Description	Enables or disables the version info API, Eth_17_GEthMacV2_GetVersionInfo(). *Note: The optional APIs are disabled by default to minimize the executable code size.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.2 Functions - Type definitions

1.3.2.1 Eth_CounterType

Table 93 Specification for Eth_CounterType

Syntax	`Eth_CounterType`	
Type	Structure	
File	`Eth_GeneralTypes.h`	
Range	uint32 DropPktBufOverrun	Dropped packets due to buffer overrun.
	uint32 DropPktCrc	Dropped packets due to CRC errors
	uint32 UndersizePkt	Number of undersize packets which were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed
	uint32 OversizePkt	Number of oversize packets which are longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed.
	uint32 AlgnmtErr	Number of alignment errors, i.e. packets which are received and are not an integral number of octets in length and do not pass the CRC.
	uint32 SqeTestErr	SQE test error

(table continues...)

Table 93 (continued) Specification for Eth_CounterType

	uint32 DiscInbdPkt	Number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol
	uint32 ErrInbdPkt	Total number of erroneous inbound packets
	uint32 DiscOtbdPkt	Number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted
	uint32 ErrOtbdPkt	Total number of erroneous outbound packets
	uint32 SnglCollPkt	Single collision frames: A count of successfully transmitted frames on a particular interface for which transmission is inhibited by exactly one collision
	uint32 MultCollPkt	Multiple collision frames: A count of successfully transmitted frames on a particular interface for which transmission is inhibited by more than one collision
	uint32 DfrdPkt	Number of deferred transmission: A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy
	uint32 LatCollPkt	Number of late collisions: The number of times that a collision is detected on a particular interface later than 512 bit-times into the transmission of a packet
	uint32 HwDepCtr0	Hardware dependent counter value - Number of good VLAN packets transmitted
	uint32 HwDepCtr1	Hardware dependent counter value - Number of good IPv4 packets received
	uint32 HwDepCtr2	Hardware dependent counter value - Number of good IP datagrams received with a good TCP payload
	uint32 HwDepCtr3	Hardware dependent counter value - Number of good IP datagrams received with a good UDP payload
Description	Statistic counter for diagnostics.	
Source	AUTOSAR	

(table continues...)

Table 93 (continued) Specification for Eth_CounterType

Autosar Version	Applicable for Autosar version 4.4.0.

1.3.2.2 Eth_MacVlanType

Table 94 Specification for Eth_MacVlanType

Syntax	`Eth_MacVlanType`	
Type	Structure	
File	`Eth_GeneralTypes.h`	
Range	uint8 MacAddr[6]	Specifies the MAC address [0..255, 0..255, 0..255, 0..255, 0..255, 0..255]
	uint16 VlanId	Specifies the VLAN address 0..65535
	uint32 SwitchPort	Specifies the ports of the switch as bit mask (0x00000001-> Port0, 0x80000001-> Port31+Port0)
Description	This type is used to read out addresses from the address resolution logic (ARL) table of the switch. In case MacAddr contains a Multicast Address MacVlanType, SwitchPort should be handled as Bitmask where each bit represents a Switch Port, e.g. Bit 0 represents EthSwitchPortIdx = 0, Bit 1 represents EthSwitchPortIdx = 1 and so on. In case of MacAddr contains not a Multicast Address MacVlanType, SwitchPort shall be handled as a value representing the EthSwitchPortIdx.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.2.3 Eth_RxStatsType

Table 95 Specification for Eth_RxStatsType

Syntax	`Eth_RxStatsType`	
Type	Structure	
File	`Eth_GeneralTypes.h`	
Range	uint32 RxStatsDropEvents	Total number of events in which packets were dropped by the probe due to lack of resources
	uint32 RxStatsOctets	Total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including FCS octets)
	uint32 RxStatsPkts	Total number of packets (including bad packets, broadcast packets, and multicast packets) received

(table continues...)

Table 95 (continued) Specification for Eth_RxStatsType

	uint32 RxStatsBroadcastPkts	Total number of good packets received that were directed to the broadcast address
	uint32 RxStatsMulticastPkts	Total number of good packets received that were directed to a multicast address
	uint32 RxStatsCrcAlignErrors	Total number of packets received that had a length between 64 and 1518 octets that had either a bad Frame Check Sequence (FCS) with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error)
	uint32 RxStatsUndersizePkts	Total number of packets received that were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise well formed
	uint32 RxStatsOversizePkts	Total number of packets received that were longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed
	uint32 RxStatsFragments	Total number of packets received that were less than 64 octets in length (excluding framing bits but including FCS octets) and had either a bad Frame Check Sequence (FCS) with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error)
	uint32 RxStatsJabbers	Total number of packets received that were longer than 1518 octets, and had either a bad Frame Check Sequence (FCS) with an integral number of octets (FCS Error) or a bad FCS with a non-integral number of octets (Alignment Error)
	uint32 RxStatsCollisions	The best estimate of the total number of collisions on this Ethernet segment
	uint32 RxStatsPkts64Octets	Total number of packets (including bad packets) received that were 64 octets in length
	uint32 RxStatsPkts65to127Octets	Total number of packets (including bad packets) received that were between 65 and 127 octets in length

(table continues...)

Table 95 (continued) Specification for Eth_RxStatsType

	uint32 RxStatsPkts128to255Octets	Total number of packets (including bad packets) received that were between 128 and 255 octets in length
	uint32 RxStatsPkts256to511Octets	Total number of packets (including bad packets) received that were between 256 and 511 octets in length
	uint32 RxStatsPkts512to1023Octets	Total number of packets (including bad packets) received that were between 512 and 1023 octets in length
	uint32 RxStatsPkts1024to1518Octets	Total number of packets (including bad packets) received that were between 1024 and 1518 octets in length
	uint32 RxUnicastFrames	Number of subnetwork-unicast packets delivered to a higher-layer protocol
Description	Receive statistic counter for diagnostics	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.2.4 Eth_TxErrorCounterValuesType

Table 96 Specification for Eth_TxErrorCounterValuesType

Syntax	`Eth_TxErrorCounterValuesType`	
Type	Structure	
File	`Eth_GeneralTypes.h`	
Range	uint32 TxDroppedNoErrorPkts	Number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted
	uint32 TxDroppedErrorPkts	Number of outbound packets that could not be transmitted because of errors
	uint32 TxDeferredTrans	Count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy
	uint32 TxSingleCollision	Count of successfully transmitted frames on a particular interface for which transmission is inhibited by exactly one collision
	uint32 TxMultipleCollision	Count of successfully transmitted frames on a particular interface for which transmission is inhibited by more than one collision

(table continues...)

Table 96 (continued) Specification for Eth_TxErrorCounterValuesType

	uint32 TxLateCollision	Number of times that a collision is detected on a particular interface later than 512 bit-times into the transmission of a packet
	uint32 TxExcessiveCollison	Count of frames for which transmission on a particular interface fails due to excessive collisions
Description	Transmission error statistic counters for diagnostics.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.2.5 Eth_TxStatsType

Table 97 Specification for Eth_TxStatsType

Syntax	`Eth_TxStatsType`	
Type	Structure	
File	`Eth_GeneralTypes.h`	
Range	uint32 TxNumberOfOctets	Total number of octets transmitted out of the interface, including framing characters
	uint32 TxNUcastPkts	Total number of packets that higher level protocols requested be transmitted to a non-unicast address, including those that were discarded or not sent
	uint32 TxUniCastPkts	Total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent
Description	Transmission statistic counter for diagnostics	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.2.6 Eth_17_GEthMacV2_ConfigType

Table 98 Specification for Eth_17_GEthMacV2_ConfigType

Syntax	`Eth_17_GEthMacV2_ConfigType`	
Type	Structure	
File	`Eth_17_GEthMacV2.h`	
Range	--	The elements of the data structure are specific to the micro-controller

(table continues...)

Table 98 (continued) Specification for Eth_17_GEthMacV2_ConfigType

Description	Defines the type for data structure containing the set of configuration parameters required for initializing the ETH driver and controller.
Source	AUTOSAR
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.2.7 Eth_BufIdxType

Table 99 Specification for Eth_BufIdxType

Syntax	`Eth_BufIdxType`	
Type	uint32	
File	`Eth_GeneralTypes.h`	
Range	0x00000000 - 0xFFFFFFFF	Ethernet buffer identifier
Description	Ethernet buffer identifier type.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.2.8 Eth_DataType

Table 100 Specification for Eth_DataType

Syntax	`Eth_DataType`	
Type	uint8	
File	`Eth_GeneralTypes.h`	
Range	0-255	One byte data
Description	This type defines the Ethernet data type used for data transmission and reception.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.2.9 Eth_FilterActionType

Table 101 Specification for Eth_FilterActionType

Syntax	`Eth_FilterActionType`	
Type	Enumeration	
File	`Eth_GeneralTypes.h`	
Range	0 - ETH_ADD_TO_FILTER	Add the MAC address to the filter, that is, allow reception
	1 - ETH_REMOVE_FROM_FILTER	Remove the MAC address from the filter, that is, reception is blocked in the lower layer.

(table continues...)

Table 101 (continued) Specification for Eth_FilterActionType

Description	The Eth_FilterActionType enumeration type describes the action to be taken for the MAC address given in *PhysAddrPtr of Eth_17_GEthMacV2_UpdatePhysAddrFilter() API.
Source	AUTOSAR
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.2.10 Eth_FrameType

Table 102 Specification for Eth_FrameType

Syntax	`Eth_FrameType`	
Type	uint16	
File	`Eth_GeneralTypes.h`	
Range	0x0000 - 0xFFFF	Ethernet frame type used in the Ethernet frame header
Description	This type defines the Ethernet frame type used in the Ethernet frame header	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.2.11 Eth_ModeType

Table 103 Specification for Eth_ModeType

Syntax	`Eth_ModeType`	
Type	Enumeration	
File	`Eth_GeneralTypes.h`	
Range	0 - ETH_MODE_DOWN	Controller disabled
	1 - ETH_MODE_ACTIVE	Controller enabled
Description	This type defines the controller modes	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.2.12 Eth_RxStatusType

Table 104 Specification for Eth_RxStatusType

Syntax	`Eth_RxStatusType`	
Type	Enumeration	
File	`Eth_GeneralTypes.h`	
Range	0 - ETH_RECEIVED	Ethernet frame has been received, no further frames available

(table continues...)

Table 104 (continued) Specification for Eth_RxStatusType

	1 - ETH_NOT_RECEIVED	Ethernet frame has not been received, no further frames available
	2 - ETH_RECEIVED_MORE_DATA_AVAILABLE	Ethernet frame has been received, more frames are available
Description	Used as OUT parameter in the Eth_17_GEthMacV2_Receive() API that indicates whether a frame has been received and if so, whether more frames are available.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.2.13 Eth_TimeStampQualType

Table 105 Specification for Eth_TimeStampQualType

Syntax	`Eth_TimeStampQualType`	
Type	Enumeration	
File	`Eth_GeneralTypes.h`	
Range	0 - ETH_VALID	0- Valid time stamp
	1 - ETH_INVALID	1- Invalid time stamp
	2 - ETH_UNCERTAIN	2-Uncertain time stamp
Description	Quality information regarding the evaluated time stamp	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.2.14 Eth_TimeStampType

Table 106 Specification for Eth_TimeStampType

Syntax	`Eth_TimeStampType`	
Type	Structure	
File	`Eth_GeneralTypes.h`	
Range	uint32 nanoseconds	Nanoseconds part of the time
	uint32 seconds	32 bit LSB of the 48 bits seconds part of the time
	uint16 secondsHi	16 bit MSB of the 48 bits seconds part of the time

(table continues...)

Table 106 (continued) Specification for Eth_TimeStampType

Description	Variables of this type are used for expressing time stamps including relative time and absolute calendar time. The absolute time starts at 1970-01-01. 0 to 281474976710655s == 3257812230d (0xFFFF FFFF FFFF) 0 to 999999999ns (0x3B9A C9FF) invalid value in nanoseconds:(0x3B9A CA00) to (0x3FFF FFFF) Bit 30 and 31 reserved, default: 0
Source	AUTOSAR
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.3 Functions - APIs

This section lists all the APIs of the ETH driver.

1.3.3.1 Eth_17_GEthMacV2_Init

Table 107 Specification for `Eth_17_GEthMacV2_Init` API

Syntax	`void Eth_17_GEthMacV2_Init ( const Eth_17_GEthMacV2_ConfigType * const CfgPtr )`	
Service ID	0x01	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	CfgPtr	Points to the implementation specific structure
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This function enables the module, chooses the mode of the external PHY interface according to the configured mode and prepares the MDIO interface. *Note: The initialization is performed only for the controllers allocated to the core from which Eth_17_GEthMacV2_Init() API is invoked.*	
Source	AUTOSAR	
Error handling	ETH_E_ACCESS, ETH_17_GETHMACV2_E_INIT_FAILED, ETH_17_GETHMACV2_E_CORE_NOT_CONFIGURED	

(table continues...)

Table 107 (continued) Specification for `Eth_17_GEthMacV2_Init` API

Configuration dependencies	-
User hints	None
SFR accessed	CPU_CORE_ID(r), GETH_CLC(rw), GETH_GPCTL(w), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.3.2 Eth_17_GEthMacV2_SetControllerMode

Table 108 Specification for `Eth_17_GEthMacV2_SetControllerMode` API

Syntax	`Std_ReturnType Eth_17_GEthMacV2_SetControllerMode ( const uint8 CtrlIdx, const Eth_ModeType CtrlMode )`	
Service ID	0x03	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx CtrlMode	Index of the ETH controller within the context of the ETH driver Mode of the controller
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: success E_NOT_OK: Controller mode could not be changed
Description	This function performs two actions: Action 1: It chooses the selected MII interface and completes the ETH controller initialization only for the controller ID passed as the input parameter. This action is done only once when this API is called for the first time after Eth_17_GEthMacV2_Init() API. Action 2: It enables or disables the ETH controller with controller ID passed as the input parameter.	
Source	AUTOSAR	
Error handling	ETH_E_ACCESS, ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_INV_PARAM, ETH_17_GETHMACV2_E_UNINIT	

(table continues...)

Table 108 (continued) Specification for `Eth_17_GEthMacV2_SetControllerMode` API

Configuration dependencies	-
User hints	None.
SFR accessed	CPU_CORE_ID(r), GETH_CLC(r), GETH_DMA_CH_CONTROL(w), GETH_DMA_CH_INTERRUPT_ENABLE(w), GETH_DMA_CH_RXDESC_LIST_ADDRESS(w), GETH_DMA_CH_RXDESC_RING_LENGTH(w), GETH_DMA_CH_RXDESC_TAIL_POINTER(w), GETH_DMA_CH_RX_CONTROL(w), GETH_DMA_CH_STATUS(w), GETH_DMA_CH_TXDESC_LIST_ADDRESS(w), GETH_DMA_CH_TXDESC_RING_LENGTH(w), GETH_DMA_CH_TXDESC_TAIL_POINTER(w), GETH_DMA_CH_TX_CONTROL(w), GETH_DMA_MODE(rw), GETH_DMA_SYSBUS_MODE(w), GETH_GPCTL(w), GETH_KRST0(rw), GETH_KRST1(w), GETH_KRSTCLR(w), GETH_MAC_ADDRESS_HIGH(w), GETH_MAC_ADDRESS_HIGH0(w), GETH_MAC_ADDRESS_LOW(w), GETH_MAC_ADDRESS_LOW0(w), GETH_MAC_CONFIGURATION(rw), GETH_MAC_PACKET_FILTER(w), GETH_MAC_RXQ_CTRL0(w), GETH_MAC_RXQ_CTRL1(w), GETH_MAC_RXQ_CTRL2(w), GETH_MAC_SUB_SECOND_INCREMENT(w), GETH_MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS(w), GETH_MAC_SYSTEM_TIME_NANOSECONDS_UPDATE(w), GETH_MAC_SYSTEM_TIME_SECONDS_UPDATE(w), GETH_MAC_TIMESTAMP_ADDEND(w), GETH_MAC_TIMESTAMP_CONTROL(rw), GETH_MMC_IPC_RX_INTERRUPT_MASK(w), GETH_MMC_RX_INTERRUPT_MASK(w), GETH_MMC_TX_INTERRUPT_MASK(w), GETH_MTL_OPERATION_MODE(w), GETH_MTL_RXQ0_OPERATION_MODE(w), GETH_MTL_RXQ_DMA_MAP0(w), GETH_MTL_RXQ_OPERATION_MODE(w), GETH_MTL_TXQ0_OPERATION_MODE(rw), GETH_MTL_TXQ0_QUANTUM_WEIGHT(w), GETH_MTL_TXQ_ETS_CONTROL(w), GETH_MTL_TXQ_HICREDIT(w), GETH_MTL_TXQ_LOCREDIT(w), GETH_MTL_TXQ_OPERATION_MODE(rw), GETH_MTL_TXQ_QUANTUM_WEIGHT(w), GETH_MTL_TXQ_SENDSLOPECREDIT(w), GETH_SKEWCTL(w), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.3.3 Eth_17_GEthMacV2_GetControllerMode

Table 109 Specification for `Eth_17_GEthMacV2_GetControllerMode` API

Syntax	`Std_ReturnType Eth_17_GEthMacV2_GetControllerMode ( const uint8 CtrlIdx, Eth_ModeType * const CtrlModePtr )`
Service ID	0x04
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info
Re-entrancy	Non Reentrant

(table continues...)

Table 109 (continued) Specification for `Eth_17_GEthMacV2_GetControllerMode` API

Parameters (in)	CtrlIdx	Index of the controller within the context of the ETH Driver
Parameters (out)	CtrlModePtr	ETH_MODE_DOWN: the controller is disabled ETH_MODE_ACTIVE: the controller is enabled
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: success E_NOT_OK: controller mode could not be obtained
Description	Obtains the state of the indexed controller	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_PARAM_POINTER, ETH_17_GETHMACV2_E_UNINIT	
Configuration dependencies	-	
User hints	None.	
SFR accessed	CPU_CORE_ID(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.3.4 Eth_17_GEthMacV2_GetPhysAddr

Table 110 Specification for `Eth_17_GEthMacV2_GetPhysAddr` API

Syntax	`void Eth_17_GEthMacV2_GetPhysAddr ( const uint8 CtrlIdx, uint8 * const PhysAddrPtr )`	
Service ID	0x08	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of ETH Controller within the context of the ETH driver.
Parameters (out)	PhysAddrPtr	Physical source address (MAC address) in the network byte order.
Parameters (in - out)	-	-

(table continues...)

Table 110 (continued) Specification for `Eth_17_GEthMacV2_GetPhysAddr` API

Return	void	-
Description	Obtains the physical source address used by the indexed controller	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_PARAM_POINTER, ETH_17_GETHMACV2_E_INV_CTRL_IDX	
Configuration dependencies	-	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_MAC_ADDRESS_HIGH0(r), GETH_MAC_ADDRESS_LOW0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.3.5 Eth_17_GEthMacV2_SetPhysAddr

Table 111 Specification for `Eth_17_GEthMacV2_SetPhysAddr` API

Syntax	`void Eth_17_GEthMacV2_SetPhysAddr ( const uint8 CtrlIdx, const uint8 * const PhysAddrPtr )`	
Service ID	0x13	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in)	CtrlIdx PhysAddrPtr	Index of the ETH controller within the context of the ETH driver. Pointer to memory containing the physical source address (MAC address) in the network byte order.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Sets the physical source address used by the indexed controller	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_PARAM_POINTER, ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_UNINIT	

(table continues...)

Table 111 (continued) Specification for `Eth_17_GEthMacV2_SetPhysAddr` API

Configuration dependencies	-
User hints	-
SFR accessed	CPU_CORE_ID(r), GETH_MAC_ADDRESS_HIGH0(w), GETH_MAC_ADDRESS_LOW0(w) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.3.6 Eth_17_GEthMacV2_UpdatePhysAddrFilter

Table 112 Specification for `Eth_17_GEthMacV2_UpdatePhysAddrFilter` API

Syntax	`Std_ReturnType Eth_17_GEthMacV2_UpdatePhysAddrFilter ( const uint8 CtrlIdx, const uint8 * const PhysAddrPtr, const Eth_FilterActionType Action )`	
Service ID	0x12	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in)	CtrlIdx PhysAddrPtr Action	Index of the ETH controller within the context of the ETH driver Pointer to the memory containing the physical destination address (MAC address) in the network byte order. This is the multicast destination address of the layer 2 ETH frame. Add or remove the address from the ETH controllers filter.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: filter is successfully changed E_NOT_OK: filter could not be changed
Description	Add or remove the MAC address from the hardware filters The filtering is only done based on the destination address of the received ETH frame. If the physical source address (MAC address) is set to FF:FF:FF:FF:FF:FF, this will completely open the filter. If the physical source address (MAC address) is set to 00:00:00:00:00:00, this will cause to reduce the filter to the controller's unique unicast MAC address and end promiscuous mode when turned on. A broadcast frame will always be allowed to pass the filter irrespective of the filter state.	

(table continues...)

Table 112 (continued) Specification for `Eth_17_GEthMacV2_UpdatePhysAddrFilter` API

Source	AUTOSAR
Error handling	ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_PARAM_POINTER, ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_INV_PARAM
Configuration dependencies	-
User hints	None
SFR accessed	CPU_CORE_ID(r), GETH_MAC_ADDRESS_HIGH(rw), GETH_MAC_ADDRESS_HIGH0(w), GETH_MAC_ADDRESS_LOW(rw), GETH_MAC_ADDRESS_LOW0(w), GETH_MAC_PACKET_FILTER(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.3.7 Eth_17_GEthMacV2_WriteMii

Table 113 Specification for `Eth_17_GEthMacV2_WriteMii` API

Syntax	`Std_ReturnType Eth_17_GEthMacV2_WriteMii ( const uint8 CtrlIdx, const uint8 TrcvIdx, const uint8 RegIdx, const uint16 RegVal )`	
Service ID	0x05	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx TrcvIdx RegIdx RegVal	Index of ETH Controller within the context of the ETH driver Index of the transceiver on the RGMII/RMII/MII Index of the transceiver register on the RGMII/RMII/MII Value to be written into the indexed register
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied

(table continues...)

Table 113 (continued) Specification for `Eth_17_GEthMacV2_WriteMii` API

Description	Configures or writes a transceiver register with the requested value. *Note 1: The Eth_17_GEthMacV2_WriteMii() API is available only when EthCtrlEnableMii is enabled.* *Note 2: The AUTOSAR specification refers to this API as asynchronous. However, this API is implemented as synchronous to satisfy the requirement to call EthTrcv_WriteMiiIndication when the MII access is finished.*
Source	AUTOSAR
Error handling	ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_INV_CTRL_IDX
Configuration dependencies	EthCtrlEnableMii
User hints	None.
SFR accessed	CPU_CORE_ID(r), GETH_MAC_MDIO_ADDRESS(rw), GETH_MAC_MDIO_DATA(w) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.3.8 Eth_17_GEthMacV2_ReadMii

Table 114 Specification for `Eth_17_GEthMacV2_ReadMii` API

Syntax	`Std_ReturnType  Eth_17_GEthMacV2_ReadMii`
	`(`
	`  const uint8 CtrlIdx,`
	`  const uint8 TrcvIdx,`
	`  const uint8 RegIdx,`
	`  uint16 * const RegValPtr`
	`)`	

Service ID	0x06	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx TrcvIdx RegIdx	Index of the controller within the context of the ETH driver Index of the transceiver on the RGMII/RMII/MII Index of the transceiver register on the RGMII/RMII/MII
Parameters (out)	RegValPtr	Filled with the register content of the indexed register
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied

(table continues...)

Table 114 (continued) Specification for `Eth_17_GEthMacV2_ReadMii` API

Description	Reads a transceiver register. *Note 1: The Eth_17_GEthMacV2_ReadMii() API is available only when EthCtrlEnableMii is enabled.* *Note 2: The AUTOSAR specification refers to this API as asynchronous. However, this API is implemented as synchronous to satisfy the requirement to call EthTrcv_ReadMiiIndication when the MII access is finished.*
Source	AUTOSAR
Error handling	ETH_17_GETHMACV2_E_PARAM_POINTER, ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_UNINIT
Configuration dependencies	EthCtrlEnableMii
User hints	None.
SFR accessed	CPU_CORE_ID(r), GETH_MAC_MDIO_ADDRESS(rw), GETH_MAC_MDIO_DATA(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.3.9 Eth_17_GEthMacV2_GetCounterValues

Table 115 Specification for `Eth_17_GEthMacV2_GetCounterValues` API

Syntax	`Std_ReturnType  Eth_17_GEthMacV2_GetCounterValues`
	`(`
	`  const uint8 CtrlIdx,`
	`  Eth_CounterType * const CounterPtr`
	`)`	

Service ID	0x14	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the ETH driver
Parameters (out)	CounterPtr	List of statistic counter values of the corresponding controller
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: success E_NOT_OK: counter values read failure

(table continues...)

Table 115 (continued) Specification for `Eth_17_GEthMacV2_GetCounterValues` API

Description	Reads a list of statistic counter values of the corresponding controller. The meaning of these counter values is described at Eth_CounterType. The maximum possible value denotes an invalid value, for example if this counter is not supported. List of statistic counter values read by this API: - DropPktBufOverrun - DropPktCrc - UndersizePkt - OversizePkt - AlgnmtErr - SqeTestErr - DiscInbdPkt - ErrInbdPkt - DiscOtbdPkt - ErrOtbdPkt - SnglCollPkt - MultCollPkt - DfrdPkt - LatCollPkt - HwDepCtr0 - HwDepCtr1 - HwDepCtr2 - HwDepCtr3 *Note 1: The counters SqeTestErr, DiscInbdPkt and DiscOtbdPkt are not supported by ETH driver. The value 0xFFFFFFFF (ETH_COUNTER_NOT_AVAILABLE) will be stored corresponding to these counter values within the structure of type Eth_CounterType.* *Note 2: Eth_17_GEthMacV2_GetCounterValues() API is available only when EthGetDropCountApi is enabled.*
Source	AUTOSAR
Error handling	ETH_17_GETHMACV2_E_PARAM_POINTER, ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_UNINIT
Configuration dependencies	EthGetDropCountApi
User hints	-

(table continues...)

Table 115 (continued) Specification for `Eth_17_GEthMacV2_GetCounterValues` API

SFR accessed	CPU_CORE_ID(r), GETH_RXIPV4_GOOD_PACKETS(r), GETH_RXTCP_GOOD_PACKETS(r), GETH_RXUDP_GOOD_PACKETS(r), GETH_RX_ALIGNMENT_ERROR_PACKETS(r), GETH_RX_CRC_ERROR_PACKETS(r), GETH_RX_FIFO_OVERFLOW_PACKETS(r), GETH_RX_JABBER_ERROR_PACKETS(r), GETH_RX_LENGTH_ERROR_PACKETS(r), GETH_RX_OUT_OF_RANGE_TYPE_PACKETS(r), GETH_RX_OVERSIZE_PACKETS_GOOD(r), GETH_RX_RECEIVE_ERROR_PACKETS(r), GETH_RX_RUNT_ERROR_PACKETS(r), GETH_RX_UNDERSIZE_PACKETS_GOOD(r), GETH_TX_CARRIER_ERROR_PACKETS(r), GETH_TX_DEFERRED_PACKETS(r), GETH_TX_EXCESSIVE_COLLISION_PACKETS(r), GETH_TX_EXCESSIVE_DEFERRAL_ERROR(r), GETH_TX_LATE_COLLISION_PACKETS(r), GETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS(r), GETH_TX_SINGLE_COLLISION_GOOD_PACKETS(r), GETH_TX_VLAN_PACKETS_GOOD(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.3.10 Eth_17_GEthMacV2_GetRxStats

Table 116 Specification for `Eth_17_GEthMacV2_GetRxStats` API

Syntax	`Std_ReturnType Eth_17_GEthMacV2_GetRxStats ( const uint8 CtrlIdx, Eth_RxStatsType * const RxStats )`	
Service ID	0x15	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the ETH driver
Parameters (out)	RxStats	List of values according to IETF RFC 2819 (Remote Network Monitoring Management Information Base)
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: success E_NOT_OK: Rx statistics could not be obtained

(table continues...)

Table 116 (continued) Specification for `Eth_17_GEthMacV2_GetRxStats` API

Description	Returns the following statistic counter values (according to IETF RFC2819), where the maximal possible value will denote an invalid value, for example, if this counter is not available. - etherStatsDropEvents - etherStatsOctets - etherStatsPkts - etherStatsBroadcastPkts - etherStatsMulticastPkts - etherStatsCrcAlignErrors - etherStatsUndersizePkts - etherStatsOversizePkts - etherStatsFragments - etherStatsJabbers - etherStatsCollisions - etherStatsPkts64Octets - etherStatsPkts65to127Octets - etherStatsPkts128to255Octets - etherStatsPkts256to511Octets - etherStatsPkts512to1023Octets - etherStatsPkts1024to1518Octets Note 1: In the above list, items that are not available are set to the value 0xFFFFFFFF (ETH_COUNTER_NOT_AVAILABLE). Note 2: The Eth_17_GEthMacV2_GetRxStats() API is available only when EthGetEtherStatsApi is enabled.
Source	AUTOSAR
Error handling	ETH_17_GETHMACV2_E_PARAM_POINTER, ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_UNINIT
Configuration dependencies	EthGetEtherStatsApi
User hints	None.

(table continues...)

Table 116 (continued) Specification for `Eth_17_GEthMacV2_GetRxStats` API

SFR accessed	CPU_CORE_ID(r), GETH_RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD(r), GETH_RX_128TO255OCTETS_PACKETS_GOOD_BAD(r), GETH_RX_256TO511OCTETS_PACKETS_GOOD_BAD(r), GETH_RX_512TO1023OCTETS_PACKETS_GOOD_BAD(r), GETH_RX_64OCTETS_PACKETS_GOOD_BAD(r), GETH_RX_65TO127OCTETS_PACKETS_GOOD_BAD(r), GETH_RX_ALIGNMENT_ERROR_PACKETS(r), GETH_RX_BROADCAST_PACKETS_GOOD(r), GETH_RX_CRC_ERROR_PACKETS(r), GETH_RX_FIFO_OVERFLOW_PACKETS(r), GETH_RX_JABBER_ERROR_PACKETS(r), GETH_RX_MULTICAST_PACKETS_GOOD(r), GETH_RX_OCTET_COUNT_GOOD_BAD(r), GETH_RX_OVERSIZE_PACKETS_GOOD(r), GETH_RX_PACKETS_COUNT_GOOD_BAD(r), GETH_RX_RUNT_ERROR_PACKETS(r), GETH_RX_UNDERSIZE_PACKETS_GOOD(r), GETH_RX_UNICAST_PACKETS_GOOD(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.3.11 Eth_17_GEthMacV2_GetTxStats

Table 117 Specification for `Eth_17_GEthMacV2_GetTxStats` API

Syntax	`Std_ReturnType Eth_17_GEthMacV2_GetTxStats ( const uint8 CtrlIdx, Eth_TxStatsType * const TxStats )`	
Service ID	0x1C	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the ETH driver
Parameters (out)	TxStats	List of statistic values for transmission.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: success E_NOT_OK: Tx statistics could not be obtained

(table continues...)

Table 117 (continued) Specification for `Eth_17_GEthMacV2_GetTxStats` API

Description	Returns the following list of transmission statistics (described in IETF RFC1213) defined with Eth_TxStatsType, where the maximal possible value denotes an invalid value, for example, if this counter is not available. - TxNumberOfOctets - TxNUcastPkts - TxUniCastPkts *Note: The Eth_17_GEthMacV2_GetTxStats() API is available only when EthGetTxStatsApi is enabled.*
Source	AUTOSAR
Error handling	ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_PARAM_POINTER
Configuration dependencies	EthGetTxStatsApi
User hints	None.
SFR accessed	CPU_CORE_ID(r), GETH_TX_BROADCAST_PACKETS_GOOD_BAD(ex_r), GETH_TX_MULTICAST_PACKETS_GOOD_BAD(ex_r), GETH_TX_OCTET_COUNT_GOOD_BAD(ex_r), GETH_TX_UNICAST_PACKETS_GOOD_BAD(ex_r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.3.12 Eth_17_GEthMacV2_GetTxErrorCounterValues

Table 118 Specification for `Eth_17_GEthMacV2_GetTxErrorCounterValues` API

Syntax	`Std_ReturnType  Eth_17_GEthMacV2_GetTxErrorCounterValues`
	`(`
	`  const uint8 CtrlIdx,`
	`  Eth_TxErrorCounterValuesType * const TxErrorCounterValues`
	`)`	

Service ID	0x1D	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the ETH driver.
Parameters (out)	TxErrorCounterValues	List of statistic error counter values for transmission.
Parameters (in - out)	-	-

(table continues...)

Table 118 (continued) Specification for `Eth_17_GEthMacV2_GetTxErrorCounterValues` API

Return	Std_ReturnType	E_OK: success E_NOT_OK: Tx error statistics could not be obtained
Description	Returns the following list of transmission error counters (described in IETF RFC1213 and RFC1643) defined with Eth_TxErrorCounterValuesType, where the maximal possible value shall denote an invalid value, for example, if this counter is not available. - TxDroppedNoErrorPkts - TxDroppedErrorPkts - TxDeferredTrans - TxSingleCollision - TxMultipleCollision - TxLateCollision - TxExcessiveCollison *Note 1: In the above list, items that are not available are set to the value 0xFFFFFFFF (ETH_COUNTER_NOT_AVAILABLE).* *Note 2: The Eth_17_GEthMacV2_GetTxErrorCounterValues() API is available only when EthGetTxErrorCounterValuesApi is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_PARAM_POINTER	
Configuration dependencies	EthGetTxErrorCounterValuesApi	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_TX_CARRIER_ERROR_PACKETS(r), GETH_TX_DEFERRED_PACKETS(r), GETH_TX_EXCESSIVE_COLLISION_PACKETS(r), GETH_TX_EXCESSIVE_DEFERRAL_ERROR(r), GETH_TX_LATE_COLLISION_PACKETS(r), GETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS(r), GETH_TX_SINGLE_COLLISION_GOOD_PACKETS(r), GETH_TX_UNDERFLOW_ERROR_PACKETS(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.3.13 Eth_17_GEthMacV2_GetCurrentTime

Table 119 Specification for `Eth_17_GEthMacV2_GetCurrentTime` API

Syntax	`Std_ReturnType  Eth_17_GEthMacV2_GetCurrentTime`
	`(`
	`  const uint8 CtrlIdx,`
	`  Eth_TimeStampQualType * const timeQualPtr,`
	`  Eth_TimeStampType * const timeStampPtr`
	`)`

(table continues...)

Table 119 (continued) Specification for `Eth_17_GEthMacV2_GetCurrentTime` API

Service ID	0x16	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the ETH driver
Parameters (out)	timeQualPtr timeStampPtr	Quality of hardware time stamp, for example, based on current drift. Note: Since the TC3xx ETH controller does not provide quality information, the reported value is always valid. Current time stamp
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Returns the time value from the hardware timer registers. *Note: The Eth_17_GEthMacV2_GetCurrentTime() function is available only when EthGlobalTimeSupport is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_PARAM_POINTER, ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_INV_CTRL_IDX	
Configuration dependencies	EthGlobalTimeSupport	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS(r), GETH_MAC_SYSTEM_TIME_NANOSECONDS(r), GETH_MAC_SYSTEM_TIME_SECONDS(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.3.14 Eth_17_GEthMacV2_EnableEgressTimeStamp

Table 120 Specification for `Eth_17_GEthMacV2_EnableEgressTimeStamp` API

Syntax	`void Eth_17_GEthMacV2_EnableEgressTimeStamp`
	`(`
	`  const uint8 CtrlIdx,`
	`  const Eth_BufIdxType BufIdx`
	`)`

Service ID	0x17

(table continues...)

Table 120 (continued) Specification for `Eth_17_GEthMacV2_EnableEgressTimeStamp` API

Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx BufIdx	Index of the controller within the context of the ETH driver Index of the message buffer, where application expects egress time stamping
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Activates egress time stamping on a dedicated message object (or message buffer) *Note: The Eth_17_GEthMacV2_EnableEgressTimeStamp() API is available only when EthGlobalTimeSupport is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_INV_PARAM	
Configuration dependencies	EthGlobalTimeSupport	
User hints	None.	
SFR accessed	CPU_CORE_ID(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.3.15 Eth_17_GEthMacV2_GetEgressTimeStamp

Table 121 Specification for `Eth_17_GEthMacV2_GetEgressTimeStamp` API

Syntax	`Std_ReturnType  Eth_17_GEthMacV2_GetEgressTimeStamp`
	`(`
	`  const uint8 CtrlIdx,`
	`  const Eth_BufIdxType BufIdx,`
	`  Eth_TimeStampQualType * const timeQualPtr,`
	`  Eth_TimeStampType * const timeStampPtr`
	`)`

Service ID	0x18
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info
Re-entrancy	Non Reentrant

(table continues...)

Table 121 (continued) Specification for `Eth_17_GEthMacV2_GetEgressTimeStamp` API

Parameters (in)	CtrlIdx BufIdx	Index of the controller within the context of the ETH driver Index of the message buffer, where application expects egress time stamping
Parameters (out)	timeQualPtr timeStampPtr	Quality of hardware time stamp, for example based on current drift Current time stamp
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: success E_NOT_OK: failed to read time stamp
Description	Reads back the egress time stamp on a dedicated message object. It must be called within the Eth_17_GEthMacV2_TxConfirmation() function. *Note: The Eth_17_GEthMacV2_GetEgressTimeStamp() function is available only when EthGlobalTimeSupport is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_PARAM_POINTER, ETH_17_GETHMACV2_E_INV_PARAM, ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_INV_CTRL_IDX	
Configuration dependencies	EthGlobalTimeSupport	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.3.16 Eth_17_GEthMacV2_GetIngressTimeStamp

Table 122 Specification for `Eth_17_GEthMacV2_GetIngressTimeStamp` API

Syntax	`Std_ReturnType  Eth_17_GEthMacV2_GetIngressTimeStamp`
	`(`
	`  const uint8 CtrlIdx,`
	`  const Eth_DataType * const DataPtr,`
	`  Eth_TimeStampQualType * const timeQualPtr,`
	`  Eth_TimeStampType * const timeStampPtr`
	`)`

Service ID	0x19
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info
Re-entrancy	Non Reentrant

(table continues...)

Table 122 (continued) Specification for `Eth_17_GEthMacV2_GetIngressTimeStamp` API

Parameters (in)	CtrlIdx DataPtr	Index of the controller within the context of the ETH driver Pointer to the message buffer, where application expects ingress time stamping Note: Since the ETH driver does not need content of message buffer for extracting time stamp, this parameter is not used in ETH driver design
Parameters (out)	timeQualPtr timeStampPtr	Quality of hardware time stamp, for example based on current drift Current time stamp
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: success E_NOT_OK: failed to read time stamp
Description	Reads back the ingress time stamp on a dedicated message object. It must be called within the EthIf_RxIndication() function. *Note: The Eth_17_GEthMacV2_GetIngressTimeStamp() function is available only when EthGlobalTimeSupport is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_PARAM_POINTER, ETH_17_GETHMACV2_E_UNINIT	
Configuration dependencies	EthGlobalTimeSupport	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.3.17 Eth_17_GEthMacV2_ProvideTxBuffer

Table 123 Specification for `Eth_17_GEthMacV2_ProvideTxBuffer` API

Syntax	`BufReq_ReturnType  Eth_17_GEthMacV2_ProvideTxBuffer`
	`(`
	`  const uint8 CtrlIdx,`
	`  const uint8 Priority,`
	`  Eth_BufIdxType * const BufIdxPtr,`
	`  uint8 ** const BufPtr,`
	`  uint16 * const LenBytePtr`
	`)`

Service ID	0x09

(table continues...)

Table 123 (continued) Specification for `Eth_17_GEthMacV2_ProvideTxBuffer` API

Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	CtrlIdx Priority	Index of the ETH controller within the context of the ETH driver Frame priority for transmit buffer FIFO selection
Parameters (out)	BufIdxPtr BufPtr	Index to the granted buffer resource. To be used for subsequent requests Pointer to the granted buffer
Parameters (in - out)	LenBytePtr	IN: desired length in bytes, OUT: granted length in bytes.
Return	BufReq_ReturnType	BUFREQ_OK: buffer provided successfully BUFREQ_E_BUSY: all buffers are used BUFREQ_E_OVFL: (i) requested buffer too large (ii) the buffer length modified (to insert management information) by EthSwt driver exceeds the configured buffer length – Only in case switch management support is enabled. BUFREQ_E_NOT_OK: API call aborted due to either (i) development error or (ii) requested priority is not configured in any of the FIFO
Description	Provides access to a transmit buffer of the FIFO related to the specified priority. *Note: The index to the granted buffer (output parameter BufIdxPtr) is also appended by the FIFO index internally used by the SW for the purpose of identification of the FIFO from which the buffer resource is provided. Thus, this granted buffer index can be a large value provided by the API.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_PARAM_POINTER, ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_INV_PARAM	
Configuration dependencies	-	
User hints	None.	
SFR accessed	CPU_CORE_ID(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.3.18 Eth_17_GEthMacV2_Transmit

Table 124 Specification for `Eth_17_GEthMacV2_Transmit` API

Syntax	`Std_ReturnType Eth_17_GEthMacV2_Transmit ( const uint8 CtrlIdx, const Eth_BufIdxType BufIdx, const Eth_FrameType FrameType , const boolean TxConfirmation, const uint16 LenByte, const uint8 * const PhysAddrPtr )`	
Service ID	0xA	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different buffer indexes and Ctrl indexes	
Parameters (in)	CtrlIdx BufIdx FrameType TxConfirmation LenByte PhysAddrPtr	Index of the controller within the context of the ETH driver Index of the buffer resource ETH frame type Activates transmission confirmation Data length in byte Physical target address (MAC address) in the network byte order
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: success E_NOT_OK: (i) transmission failed or (ii) EthSwt function invoked by this API incorrectly modifies the output parameters (Only in case switch management support is enabled).
Description	Triggers transmission of a previously filled transmit buffer	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_INV_PARAM, ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_PARAM_POINTER, ETH_17_GETHMACV2_E_INV_MODE	
Configuration dependencies	-	
User hints	None	
SFR accessed	CPU_CORE_ID(r), GETH_DMA_CH_TXDESC_TAIL_POINTER(w), GETH_MAC_ADDRESS_HIGH0(r), GETH_MAC_ADDRESS_LOW0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	

(table continues...)

Table 124 (continued) Specification for `Eth_17_GEthMacV2_Transmit` API

Autosar Version	Applicable for Autosar version 4.4.0.

1.3.3.19 Eth_17_GEthMacV2_Receive

Table 125 Specification for `Eth_17_GEthMacV2_Receive` API

Syntax	`void Eth_17_GEthMacV2_Receive ( const uint8 CtrlIdx, const uint8 FifoIdx, Eth_RxStatusType * const RxStatusPtr )`	
Service ID	0xB	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different FIFO indexes and Ctrl indexes	
Parameters (in)	CtrlIdx FifoIdx	Index of the controller within the context of the ETH driver Index of the related FIFO
Parameters (out)	RxStatusPtr	ETH_RECEIVED: Ethernet frame has been received, no further frames available ETH_RECEIVED_MORE_DATA_AVAILABLE: Ethernet frame has been received, more frames are available ETH_NOT_RECEIVED: (i) Ethernet frame has not been received or (ii) EthSwt function invoked by this API incorrectly modifies the output parameters (Only in case switch management support is enabled).
Parameters (in - out)	-	-
Return	void	-
Description	Receive a frame from the related FIFO. The Eth_17_GEthMacV2_Receive() API reads the next frame from the receive buffers of the specified FIFO. This function passes the received frame to the ETH interface using the EthIf_RxIndication() callback function and indicates if there are more frames in the receive buffers through RxStatusPtr. When calling the EthIf_RxIndication() callback function, the broadcast frames are indicated to the ETH interface.	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_INV_MODE, ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_INV_PARAM	
Configuration dependencies	-	
User hints	None	

(table continues...)

Table 125 (continued) Specification for `Eth_17_GEthMacV2_Receive` API

SFR accessed	CPU_CORE_ID(r), GETH_DMA_CH_RXDESC_TAIL_POINTER(w) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar version 4.4.0.

1.3.3.20 Eth_17_GEthMacV2_TxConfirmation

Table 126 Specification for `Eth_17_GEthMacV2_TxConfirmation` API

Syntax	`void Eth_17_GEthMacV2_TxConfirmation ( const uint8 CtrlIdx )`	
Service ID	0xC	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the ETH driver
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Triggers frame transmission confirmation	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_INV_MODE, ETH_17_GETHMACV2_E_UNINIT	
Configuration dependencies	-	
User hints	None	
SFR accessed	CPU_CORE_ID(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.3.21 Eth_17_GEthMacV2_GetVersionInfo

Table 127 Specification for `Eth_17_GEthMacV2_GetVersionInfo` API

Syntax	`void Eth_17_GEthMacV2_GetVersionInfo ( Std_VersionInfoType * const VersionInfoPtr )`	
Service ID	0xD	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	VersionInfoPtr	Version information of this module
Parameters (in - out)	-	-
Return	void	-
Description	Returns the version information of the ETH driver. *Note: The Eth_17_GEthMacV2_GetVersionInfo() API is available only when EthVersionInfoApi is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMACV2_E_PARAM_POINTER	
Configuration dependencies	EthVersionInfoApi	
User hints	None.	
SFR accessed	-	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.4 Notifications and Callbacks

The ETH driver does not provide any notification or callbacks.

1.3.5 Scheduled functions

This section lists all the scheduled functions of the ETH driver.

1.3.5.1 Eth_17_GEthMacV2_MainFunction

Table 128 Specification for `Eth_17_GEthMacV2_MainFunction` API

Syntax	`void Eth_17_GEthMacV2_MainFunction ( void )`	
Service ID	0x20	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This function checks for the controller errors and lost frames.	
Source	AUTOSAR	
Error handling	ETH_E_RX_FRAMES_LOST, ETH_E_LATECOLLISION, ETH_E_ALIGNMENT, ETH_E_OVERSIZEFRAME , ETH_E_CRC, ETH_E_UNDERSIZEFRAME, ETH_E_SINGLECOLLISION, ETH_E_MULTIPLECOLLISION	
Configuration dependencies	-	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_RX_ALIGNMENT_ERROR_PACKETS(r), GETH_RX_CRC_ERROR_PACKETS(r), GETH_RX_FIFO_OVERFLOW_PACKETS(r), GETH_RX_OVERSIZE_PACKETS_GOOD(r), GETH_RX_UNDERSIZE_PACKETS_GOOD(r), GETH_TX_LATE_COLLISION_PACKETS(r), GETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS(r), GETH_TX_SINGLE_COLLISION_GOOD_PACKETS(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.6 Interrupt service routines

This section lists all the interrupt handlers of the ETH driver.

1.3.6.1 Eth_17_GEthMacV2_RxDmaIrqHdlr

Table 129 Specification for `Eth_17_GEthMacV2_RxDmaIrqHdlr` API

Syntax	`void Eth_17_GEthMacV2_RxDmaIrqHdlr ( const uint8 CtrlIdx, const uint8 DmaChnlIdx )`	
Service ID	0x10	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different DMA channel indexes and Ctrl indexes	
Parameters (in)	CtrlIdx DmaChnlIdx	Controller Index Receive DMA channel index
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	None.
Description	IRQ handler for the frame reception interrupt from the receive DMA channel and the controller index passed as input parameters. *Note 1: The ETH driver is not handling any error-related interrupts.* *Note 2: This receive interrupt handler is available only when EthCtrlEnableRxInterrupt is enabled.*	
Source	IFX	
Error handling	ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_INV_PARAM	
Configuration dependencies	EthCtrlEnableRxInterrupt	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_DMA_CH_RXDESC_TAIL_POINTER(w), GETH_DMA_CH_STATUS(rw) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.6.2 Eth_17_GEthMacV2_TxDmaIrqHdlr

Table 130 Specification for `Eth_17_GEthMacV2_TxDmaIrqHdlr` API

Syntax	`void Eth_17_GEthMacV2_TxDmaIrqHdlr ( const uint8 CtrlIdx, const uint8 DmaChnlIdx )`	
Service ID	0x11	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different DMA channel indexes and Ctrl indexes	
Parameters (in)	CtrlIdx DmaChnlIdx	Controller Index Transmit DMA channel index
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	None.
Description	IRQ handler for the frame transmission interrupt from the transmit DMA channel and the controller index passed as input parameters. *Note 1: The ETH driver does not handle any error-related interrupts.* *Note 2: This transmit interrupt handler is available only when EthCtrlEnableTxInterrupt is enabled.*	
Source	IFX	
Error handling	ETH_17_GETHMACV2_E_INV_CTRL_IDX, ETH_17_GETHMACV2_E_UNINIT, ETH_17_GETHMACV2_E_INV_PARAM	
Configuration dependencies	EthCtrlEnableTxInterrupt	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_DMA_CH_STATUS(rw) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar version 4.4.0.	

1.3.7 Callout

The ETH driver does not provide any callout.

1.3.8 Errors Handling

This section describes the various errors reported by the ETH driver.

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
ETH_17_GETHMACV2_E_CORE_NOT_CONFIGURED: ETH controller not configured to the core.	IFX	NA	NA	0x64	DET
ETH_E_ACCESS: ETH controller access failure	AUTOSAR	NA	NA	Value Assigned by DEM	Production Error
ETH_E_ALIGNMENT: Frame alignment error	AUTOSAR	NA	NA	Value Assigned by DEM	Production Error
ETH_E_CRC: CRC failure	AUTOSAR	NA	NA	Value Assigned by DEM	Production Error
ETH_E_LATECOLLISION: Late frame collision	AUTOSAR	NA	NA	Value Assigned by DEM	Production Error
ETH_E_MULTIPLECOLLISION: Multiple frame collision	AUTOSAR	NA	NA	Value Assigned by DEM	Production Error
ETH_E_OVERSIZEFRAME : Frame size overflow	AUTOSAR	NA	NA	Value Assigned by DEM	Production Error
ETH_E_RX_FRAMES_LOST: ETH frames lost	AUTOSAR	NA	NA	Value Assigned by DEM	Production Error
ETH_E_SINGLECOLLISION: Single frame collision	AUTOSAR	NA	NA	Value Assigned by DEM	Production Error
ETH_E_UNDERSIZEFRAME: Frame size underflow	AUTOSAR	NA	NA	Value Assigned by DEM	Production Error
ETH_17_GETHMACV2_E_INV_CTRL_IDX: Invalid controller index. Note: This development error is reported if the controller index is invalid/ if the controller is not allocated to the current core.	AUTOSAR	NA	NA	0x01	DET

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
ETH_17_GETHMACV2_E_UNINIT: ETH driver or controller is not initialized. Note: This development error is reported if Eth_17_GEthMacV2_Init() API is not called before invoking runtime APIs and if controller is not configured to the current core.	AUTOSAR	NA	NA	0x02	DET
ETH_17_GETHMACV2_E_PARAM_POINTER: Invalid pointer in parameter list.	AUTOSAR	NA	NA	0x03	DET
ETH_17_GETHMACV2_E_INV_PARAM: Invalid parameter.	AUTOSAR	NA	NA	0x04	DET
ETH_17_GETHMACV2_E_INV_MODE: Invalid controller mode.	AUTOSAR	NA	NA	0x05	DET
ETH_17_GETHMACV2_E_INIT_FAILED: Invalid configuration set selection.	AUTOSAR	NA	NA	0x20	DET

1.3.9 Deviations and limitations

The section describes the deviations and limitations of the ETH driver.

1.3.9.1 Deviations

The section describes the deviations of the ETH driver.

1.3.9.1.1 Software specification deviations

This section describes the deviations from software specification.

Table 131 Known deviations

Reference	Deviation
AUTOSAR requirements - SWS_Eth_00216 ,SWS_Eth_00217	Individual enabling of hardware checksum offload functionality for IPV4, UDP, TCP, ICMP frames is not possible due to hardware limitation. Enabling any one of the following configuration parameters: `EthCtrlEnableOffloadChecksumIPV4`, `EthCtrlEnableOffloadChecksumUDP`, `EthCtrlEnableOffloadChecksumTCP`,`EthCtrlEnableOffloadChecksumICMP` enables the checksum offload functionality. This is a deviation from the AUTOSAR requirements [SWS_Eth_00216] , [SWS_Eth_00217].

(table continues...)

Table 131 (continued) Known deviations

Reference	Deviation
For all requirements related to Production errors	Reporting of Production error: Dem_SetEventStatus is done through Mcal_Wrapper_Dem_SetEventStatus interface for AUTOSAR 4.4.0. All production error related datatypes and modified interfaces inclusion shall be done via Mcal_Wrapper.h.
AUTOSAR requirement - SWS_Eth_00034	The `Eth_17_GEthMacV2_Init()` API does not perform the configuration of all controller configuration parameters (e.g. interrupts, frame length, frame filter,..) and configuration of all transmit / receive resources (e.g. buffer initialization) as per the AUTOSAR requirement [SWS_Eth_00034]. This configuration is done by `Eth_17_GEthMacV2_SetControllerMode()` API when it is invoked for the first time for the controller. Refer to the *ETH controller initialization sequence* section of Key architectural considerations for more details on this.
AUTOSAR requirements - SWS_Eth_00257, SWS_Eth_00258, SWS_Eth_00260, SWS_Eth_00261, ECUC_Eth_00065, ECUC_Eth_00064	These AUTOSAR requirements are related to the AUTOSAR multicore concept using EcuC partitions that is not supported by the driver. The multicore support is implemented according to the INFINEON multicore concept involving the Resource Manager module.
AUTOSAR requirements - ECUC_Eth_00063, ECUC_Eth_00062	In the AUTOSAR specification, for the `EthCtrlMacLayerSpeed` and `EthCtrlMacLayerSubType` parameters, the Value Configuration Class is mentioned as Link time in VARIANT-POST-BUILD. However, for these parameters, the Value Configuration Class is implemented as Post-build time that is of a higher Configuration Class.
AUTOSAR requirement - ECUC_Eth_00056	In the AUTOSAR specification, the `EthCtrlConfigSchedulerPredecessorRef` parameter can be configured to have a reference to either `EthCtrlConfigEgressFifo`, `EthCtrlConfigScheduler` or `EthCtrlConfigShaper`. However, since multiple schedulers are not supported within the GETH MAC hardware, the reference to `EthCtrlConfigScheduler` should not be configured. If configured, an error is reported during code generation.

1.3.9.1.2 AMDC Violations

The ETH driver does not have any AMDC violations.

1.3.9.1.3 VSMD Violations

This section describes the violations reported by the EB VSMD checker tool with respect to AUTOSAR.

Table 132 Violations reported by VSMD checker tool for TpsEcuc_08006

Rule ID:	TpsEcuc_08006
VSMD Node(s):	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo
Description:	If the multiplicityConfigClass attribute of an EcucContainerDef is defined in the StMD and its upperMultiplicity is greater than lowerMultiplicity, multiplicityConfigClass.configClass for each multiplicityConfigClass.configVariant in the VSMD shall be the same or higher as in the StMD with respect to the selected subset defined by the actually implemented supportedConfigVariant of the corresponding EcucModuleDef.
Additional Information:	The multiplicityConfigClass for the above configuration parameters is deviated (changed to Precompile) from AUTOSAR due to the following reasons: 1. The parameters within these containers (`EthCtrlConfigEgressFifo`, `EthCtrlConfigIngressFifo`) are used for generating pre-compile macro for the total number of buffers allocated to each egress/ ingress FIFO identified by a FIFO index. 2. These parameters within these containers (`EthCtrlConfigEgressFifo`, `EthCtrlConfigIngressFifo`) are used for generating pre-compile macro for total size allocated to each egress/ ingress FIFO identified by a FIFO index. These generated pre-compile macros are further used in code for memory allocation. So the multiplicity of the containers cannot be changed at post-build and hence the multiplicityConfigClass is set to Precompile.

Table 133 Violations reported by VSMD checker tool for TpsEcuc_08027

Rule ID:	TpsEcuc_08027
VSMD Node(s):	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo
Description:	If the EcucModuleDef.postBuildVariantSupport is set to true and the postBuildVariantMultiplicity attribute of an EcucContainerDef in this EcucModuleDef in the StMD is set to true, the corresponding VSMD shall also set it to true.

(table continues...)

Table 133 (continued) Violations reported by VSMD checker tool for TpsEcuc_08027

Additional Information:	The postBuildVariantMultiplicity for the above configuration parameters is deviated (changed to FALSE) from AUTOSAR due to the following reasons: 1. The parameters within these containers (`EthCtrlConfigEgressFifo`, `EthCtrlConfigIngressFifo`) are used for generating pre-compile macro for the total number of buffers allocated to each egress/ ingress FIFO identified by a FIFO index. 2. These parameters within these containers (`EthCtrlConfigEgressFifo`, `EthCtrlConfigIngressFifo` are used for generating pre-compile macro for total size allocated to each egress/ ingress FIFO identified by a FIFO index. These generated pre-compile macros are further used in code for memory allocation. So the multiplicity of the containers cannot be changed at post-build and hence the postBuildVariantMultiplicity is set to FALSE.

Table 134 Violations reported by VSMD checker tool for TpsEcuc_08032

Rule ID:	TpsEcuc_08032
VSMD Node(s):	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/ EthCtrlConfigEgress/EthCtrlConfigEgressFifo/ EthCtrlConfigEgressFifoBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/ EthCtrlConfigEgress/EthCtrlConfigEgressFifo/ EthCtrlConfigEgressFifoBufTotal /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/ EthCtrlConfigEgress/EthCtrlConfigEgressFifo/ EthCtrlConfigEgressFifoIdx /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/ EthCtrlConfigIngress/EthCtrlConfigIngressFifo/ EthCtrlConfigIngressFifoBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/ EthCtrlConfigIngress/EthCtrlConfigIngressFifo/ EthCtrlConfigIngressFifoBufTotal /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/ EthCtrlConfigIngress/EthCtrlConfigIngressFifo/ EthCtrlConfigIngressFifoIdx
Description:	If the EcucModuleDef.postBuildVariantSupport is set to true and the postBuildVariantValue for an EcucParameterDef or an EcucAbstractReferenceDef in this EcucModuleDef in the StMD is set to true, the corresponding VSMD shall also set it to true.

(table continues...)

Table 134 (continued) Violations reported by VSMD checker tool for TpsEcuc_08032

Additional Information:	The postBuildVariantValue for the above configuration parameters is deviated(changed to FALSE) from AUTOSAR due to the following reasons: 1. These parameters are used for generating pre-compile macro for the total number of buffers allocated to each egress/ ingress FIFO identified by a FIFO index. 2. These parameters are used for generating pre-compile macro for total size allocated to each egress/ ingress FIFO identified by a FIFO index. These generated pre-compile macros are further used in code for memory allocation and hence the postBuildVariantValue is set to FALSE.

Table 135 Violations reported by VSMD checker tool for TpsEcuc_08033

Rule ID:	TpsEcuc_08033
VSMD Node(s):	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlMacLayerSpeed /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlMacLayerSubType /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlPhyAddress
Description:	If the EcucModuleDef.postBuildVariantSupport is set to true and the postBuildVariantMultiplicity for an EcucParameterDef or an EcucAbstractReferenceDef in this EcucModuleDef in the StMD is set to true, the corresponding VSMD shall also set it to true.
Additional Information:	For Ethernet Controller initialization, it is required to configure the values for MAC layer speed, MAC layer Subtype and the physical address. Hence, the container multiplicity is fixed to 1 and hence the post build multiplicity (0..1) cannot be supported.

Table 136 Violations reported by VSMD checker tool for TpsEcuc_08038

Rule ID:	TpsEcuc_08038

(table continues...)

Table 136 (continued) Violations reported by VSMD checker tool for TpsEcuc_08038

VSMD Node(s):	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo/EthCtrlConfigEgressFifoBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo/EthCtrlConfigEgressFifoBufTotal /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo/EthCtrlConfigEgressFifoIdx /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo/EthCtrlConfigIngressFifoBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo/EthCtrlConfigIngressFifoBufTotal /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo/EthCtrlConfigIngressFifoIdx
Description:	If the valueConfigClass attribute for an EcucParameterDef or an EcucAbstractReferenceDef is defined in the StMD, valueConfigClass.configClass for each valueConfigClass.configVariant in the VSMD shall be the same or higher as in the StMD with respect to the selected subset defined by the actually implemented supportedConfigVariant of the corresponding EcucModuleDef.
Additional Information:	The value configuration class for the above configuration parameters is deviated (changed to Precompile) from AUTOSAR due to the following reasons 1. These parameters are used for generating pre-compile macro for the total number of buffers allocated to each egress/ ingress FIFO identified by a FIFO index. 2. These parameters are used for generating pre-compile macro for total size allocated to each egress/ ingress FIFO identified by a FIFO index. These generated pre-compile macros are further used in code for memory allocation and hence should be pre-compile time.

Table 137 Violations reported by VSMD checker tool for EB03

Rule ID:	EB03

(table continues...)

Table 137 (continued) Violations reported by VSMD checker tool for EB03

VSMD Node(s):	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigShaper/EthCtrlConfigShaperIdleSlope /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlEcucPartitionRef /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlMacLayerSpeed /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlMacLayerSubType /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlPhyAddress /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_ACCESS /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_ALIGNMENT /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_CRC /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_LATECOLLISION /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_MULTIPLECOLLISION /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_OVERSIZEFRAME /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_RX_FRAMES_LOST /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_SINGLECOLLISION /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_UNDERSIZEFRAME
Description:	The StMD node has LOWER-MULTIPLICITY=0 and UPPER-MULTIPLICITY=1. The VSMD-node shall get the OPTIONAL-attribute instead of creating a list!
Additional Information:	-

Table 138 Violations reported by VSMD checker tool for EB09

Rule ID:	EB09
VSMD Node(s):	/AURIX2G/EcucDefs/Eth
Description:	EB specific rule to check consistency of parameter postBuildVariantUsed.

(table continues...)

Table 138 (continued) Violations reported by VSMD checker tool for EB09

Additional Information:	-

Table 139 Violations reported by VSMD checker tool for EcucSws_1014

Rule ID:	EcucSws_1014
VSMD Node(s):	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigShaper /AURIX2G/EcucDefs/Eth/EthGeneral
Description:	Additional vendor specific parameter definitions (using ParameterTypes), container definitions and references shall be added to the VSMD according to the alphabetical order.
Additional Information:	-

Table 140 Violations reported by VSMD checker tool for EcucSws_1035

Rule ID:	EcucSws_1035

(table continues...)

Table 140 (continued) Violations reported by VSMD checker tool for EcucSws_1035

VSMD Node(s):	/AURIX2G/EcucDefs/Eth /AURIX2G/EcucDefs/Eth/EthConfigSet /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo/EthCtrlConfigEgressFifoBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo/EthCtrlConfigEgressFifoBufTotal /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo/EthCtrlConfigEgressFifoIdx /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo/EthCtrlConfigEgressFifoPriorityAssignment /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressLastSchedulerRef /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigScheduler /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigScheduler/EthCtrlConfigSchedulerPredecessor /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigScheduler/EthCtrlConfigSchedulerPredecessor/EthCtrlConfigSchedulerPredecessorOrder /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigScheduler/EthCtrlConfigSchedulerPredecessor/EthCtrlConfigSchedulerPredecessorRef /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigShaper /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigShaper/EthCtrlConfigShaperIdleSlope /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigShaper/EthCtrlConfigShaperPredecessorFifoRef /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress

(table continues...)

Table 140 (continued) Violations reported by VSMD checker tool for EcucSws_1035

	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo/EthCtrlConfigIngressFifoBufLenByte
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo/EthCtrlConfigIngressFifoBufTotal
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo/EthCtrlConfigIngressFifoIdx
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo/EthCtrlConfigIngressFifoPriorityAssignment
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlEcucPartitionRef
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlEnableMii
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlEnableRxInterrupt
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlEnableTxInterrupt
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlIdx
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlMacLayerSpeed
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlMacLayerSubType
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlMacLayerType
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlPhyAddress
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_ACCESS
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_ALIGNMENT
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_CRC
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_LATECOLLISION
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_MULTIPLECOLLISION

(table continues...)

Table 140 (continued) Violations reported by VSMD checker tool for EcucSws_1035

	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_OVERSIZEFRAME /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_RX_FRAMES_LOST /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_SINGLECOLLISION /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_UNDERSIZEFRAME /AURIX2G/EcucDefs/Eth/EthGeneral/AURIX2G/EcucDefs/Eth/EthGeneral/EthCtrlOffloading /AURIX2G/EcucDefs/Eth/EthGeneral/EthCtrlOffloading/EthCtrlEnableOffloadChecksumICMP /AURIX2G/EcucDefs/Eth/EthGeneral/EthCtrlOffloading/EthCtrlEnableOffloadChecksumIPv4 /AURIX2G/EcucDefs/Eth/EthGeneral/EthCtrlOffloading/EthCtrlEnableOffloadChecksumTCP /AURIX2G/EcucDefs/Eth/EthGeneral/EthCtrlOffloading/EthCtrlEnableOffloadChecksumUDP /AURIX2G/EcucDefs/Eth/EthGeneral/EthDevErrorDetect /AURIX2G/EcucDefs/Eth/EthGeneral/EthEcucPartitionRef /AURIX2G/EcucDefs/Eth/EthGeneral/EthGetDropCountApi /AURIX2G/EcucDefs/Eth/EthGeneral/EthGetEtherStatsApi /AURIX2G/EcucDefs/Eth/EthGeneral/EthGetTxErrorCounterValuesApi /AURIX2G/EcucDefs/Eth/EthGeneral/EthGetTxStatsApi /AURIX2G/EcucDefs/Eth/EthGeneral/EthGlobalTimeSupport /AURIX2G/EcucDefs/Eth/EthGeneral/EthIndex /AURIX2G/EcucDefs/Eth/EthGeneral/EthMainFunctionPeriod /AURIX2G/EcucDefs/Eth/EthGeneral/EthMaxCtrlsSupported /AURIX2G/EcucDefs/Eth/EthGeneral/EthVersionInfoApi
Description:	For Containers, Parameters and References elements UUID must be unique (also between StMD and VSMD).

(table continues...)

Table 140 (continued) Violations reported by VSMD checker tool for EcucSws_1035

Additional Information:	-

Table 141 Violations reported by VSMD checker tool for EcucSws_2101

Rule ID:	EcucSws_2101
VSMD Node(s):	/AURIX2G/EcucDefs/Eth/POST_BUILD_VARIANT_USED
Description:	For each ConfigurationVariant supported by the ModuleDef, there must be one ImplementationConfigClass element. In VSMD, the ImplementationConfigClass is mandatory.
Additional Information:	-

Table 142 Violations reported by VSMD checker tool for EcucSws_6003

Rule ID:	EcucSws_6003
VSMD Node(s):	/AURIX2G/EcucDefs/Eth
Description:	The SHORT-NAME of the AR-PACKAGEs of StMD and VSMD must be different to ensure a unique SHORT-NAME-path.
Additional Information:	-

Table 143 Violations reported by VSMD checker tool for TpsEcuc_06051_ASR41

Rule ID:	TpsEcuc_06051_ASR41
VSMD Node(s):	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo/EthCtrlConfigEgressFifoBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo/EthCtrlConfigEgressFifoBufTotal /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigEgress/EthCtrlConfigEgressFifo/EthCtrlConfigEgressFifoIdx /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo/EthCtrlConfigIngressFifoBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo/EthCtrlConfigIngressFifoBufTotal /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlConfigIngress/EthCtrlConfigIngressFifo/EthCtrlConfigIngressFifoIdx

(table continues...)

Table 143 (continued) Violations reported by VSMD checker tool for TpsEcuc_06051_ASR41

Description:	The implementationConfigClass of an EcucParameterDef or EcucAbstractReferenceDef in VSMD shall be the same or higher (where PreCompile configuration class is considered to be the lowest and PostBuild the highest) as in StMD with respect to the selected subset defined by the actually implemented supportedConfigVariant.
Additional Information:	The implementationConfigClass for the above configuration parameters is deviated (changed to PreCompile) from AUTOSAR due to the following reasons 1. These parameters are used for generating pre-compile macro for the total number of buffers allocated to each egress/ ingress FIFO identified by a FIFO index. 2. These parameters are used for generating pre-compile macro for total size allocated to each egress/ ingress FIFO identified by a FIFO index. These generated pre-compile macros are further used in code for memory allocation and hence the implementationConfigClass is set to PreCompile.

1.3.9.2 Limitations

The section describes the limitations of the ETH driver.

Table 144 Known limitations

Reference	Limitation
Ethernet transceivers intermittently fails to transmit first Ethernet packet transmitted from Ethernet MAC	Ethernet driver testing is performed using Triboard which is an evaluation board from Infineon. It is observed that the transceivers used in Triboard intermittently fail to transmit first packet transmitted from Ethernet MAC. But it is tested and confirmed that all packets which are sent from Ethernet MAC are transmitted successfully to Ethernet transceivers. Using a qualified Ethernet transceiver subsystem (hardware and driver software) should resolve this behavior. The work around followed while testing the Ethernet driver is, after the transceiver is initialized a delay of 3 to 4 seconds is added in test code.

(table continues...)

Table 144 (continued) Known limitations

Reference	Limitation
Maximum number of egress FIFOs and ingress FIFOs that can be configured is limited to 4. AUTOSAR requirements - ECUC_Eth_00048, ECUC_Eth_00043	The ETH controller supports only 4 Tx DMA channels and 4 Rx DMA channels. Hence it is allowed to configure upto 4 egress FIFOs / 4 ingress FIFOs as each FIFO is associated with a DMA channel. Due to this, the container multiplicity of `EthCtrlConfigEgressFifo` and `EthCtrlConfigIngressFifo` is changed to 0..4, and the range of `EthCtrlConfigEgressFifoIdx` and `EthCtrlConfigIngressFifoIdx` is changed to 0..3. Also, the container multiplicity of `EthCtrlConfigSchedulerPredecessor` is changed to 1..4 and `EthCtrlConfigShaper` is changed to 0..3.
With switch management support functionality enabled, only one ingress FIFO is supported.	When the switch management support functionality is enabled, it is expected that the switch management related information is present within the Ethernet packet and the position of this information resides after the address field (MAC source address) and before the EthType field within the Ethernet packet. Due to this, the VLAN tag position along with the data field is shifted by the number of bytes of switch management information. As the VLAN field is shifted, the routing of packets to the ingress FIFOs cannot be performed by the hardware based on the priority field. Thus, multiple ingress FIFOs cannot be supported together with switch management support feature.
Number of buffers within an egress FIFO AUTOSAR requirement - ECUC_Eth_00050	The range of `EthCtrlConfigEgressFifoBufTotal` parameter is changed to 0 - 8191. The range is reduced to 13 bits from 16 bits, because the upper 3 bits (out of the 16 bits for total buffer range) is utilized by the software to store the internal index to refer the FIFO from which the buffer is allocated by `Eth_17_GEthMacV2_ProvideTxBuffer()` API.
The sum of the individual buffer lengths (converted to the multiple of 256 bytes) of the egress FIFOs should not exceed 4 kB	On the MAC transaction layer, the ETH controller supports Tx FIFO size of 4 kB. This Tx FIFO size of 4 kB is allocated among the Tx queues based on the configured egress FIFO buffer length (`EthCtrlConfigEgressFifoBufLenByte`) and also on the egress FIFO buffer count (`EthCtrlConfigEgressFifoBufTotal`). The egress FIFOs should be configured such that the sum of the individual buffer lengths (converted to the multiple of 256 bytes) should not exceed 4 kB. For example, if 3 egress FIFOs are configured with the parameter `EthCtrlConfigEgressFifoBufLenByte` configured to a value of 1500 bytes, then the total size exceeds 4 kB and this cannot be supported. The code generation tool detects this condition and reports this as an error.

(table continues...)

Table 144 (continued) Known limitations

Reference	Limitation
The total application RAM area allocated for the configured ingress FIFO should not be less than the size of any received Ethernet packet.	Issue is observed in the receive operation when the total application RAM area allocated for an ingress FIFO is insufficient to hold the frame that is currently being received. Due to this, the Rx DMA is unable to transfer the complete frame from the internal queue (MTL queue) to application RAM and the frame transfer completion indication is not provided. After the occurrence of such an event, none of the frames are received further. To avoid this issue, the user has to ensure during configuration that the size of RAM allocated for ingress FIFO (that is, `EthCtrlConfigIngressFifoBufLenByte` * `EthCtrlConfigIngressFifoBufTotal`) should be greater than or equal to the maximum frame size that can be received over the Ethernet network. When sufficient ingress FIFO RAM space is allocated for the incoming frames, then this issue is not observed.
MDIO (Clause 22 and Clause 45) master interface for PHY device configuration and management	Only Clause 22 MDIO frame structure is currently supported by the Ethernet driver. Clause 45 frame structure is not supported.

Revision history

Table 145 Revision History

Date	Version	Description
2022-06-13	4.0	Released
2023-06-12	3.1	- 1.1.3.1 C file Structure section, Figure 2 Can_C_File_Structure-1.png updated to add Mcal_Wrapper.h inclusion by Eth_17_GEthMacV2.h and remove Dem.h - 1.1.3.1 C File Structure section, Table 2 C File Structure updated to add Mcal_Wrapper.h and remove Dem.h - 1.1.2 Hardware-software mapping section, Figure 1 Mapping of hardware-software interfaces updated to add Mcal_Wrapper and remove Dem - DEM module removed and Mcal_Wrapper module added in 1.1.4.1 Integration with AUTOSAR stack section - Runtime error information removed from DET module and added in Mcal_Wrapper module in 1.1.4.1 Integration with AUTOSAR stack section - References to DEM and Dem_SetEventStatus() are changed to Production error and Mcal_Wrapper_Dem_SetEventStatus() respectively in the following sections: • 1.3.8 Errors Handling • 1.3.1.6 Container: EthDemEventParameterRefs - 1.3.9.1.1 Software specification deviations section, Table 108 Known deviations updated: • AUTOSAR requirements - SWS_Eth_00026 reference updated to "For all requirements related to Production errors". • Deviation description updated to add Mcal_Wrapper information. - ASIL Level field changed to Safety Level with value as 'Refer to the release notes for the safety related info' for all functions under 1.3.3 Functions - APIs, 1.3.5 Scheduled functions, and 1.3.6 Interrupt service routines. - 1.3.9.2 Limitations section, Table 117 Known Limitations updated to add limitation for Ethernet supporting only clause 22 and not clause 45 for MDIO Frame Stucture
2021-11-17	3.0	Released
2021-11-15	2.1	Updated config variant info
2020-11-24	2.0	Released
2020-11-19	1.1	Updated the Limitations section with the limitation on application RAM size for ingress FIFO buffer.
2020-09-18	1.0	Document is released
2020-09-15	0.1	Initial Version

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2023-06-13

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-ocr1484806431059

Important notice

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.