MCAL User Manual for Eth_17_GEthMac

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_GEthMac 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 Ethernet Driver, AUTOSAR_SWS_Ethernet_Driver, AUTOSAR Release 4.2.2

Table of contents

Table of contents

Table of contents

Table of contents

1 Eth_17_GEthMac driver

1.1 User information

1.1.1 Description

The Ethernet (ETH) driver is responsible for providing standard ETH controller services specified by AUTOSAR. This enables the upper layer (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, ETH statistics and ETH drop count.

1.1.2 Hardware-software mapping

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

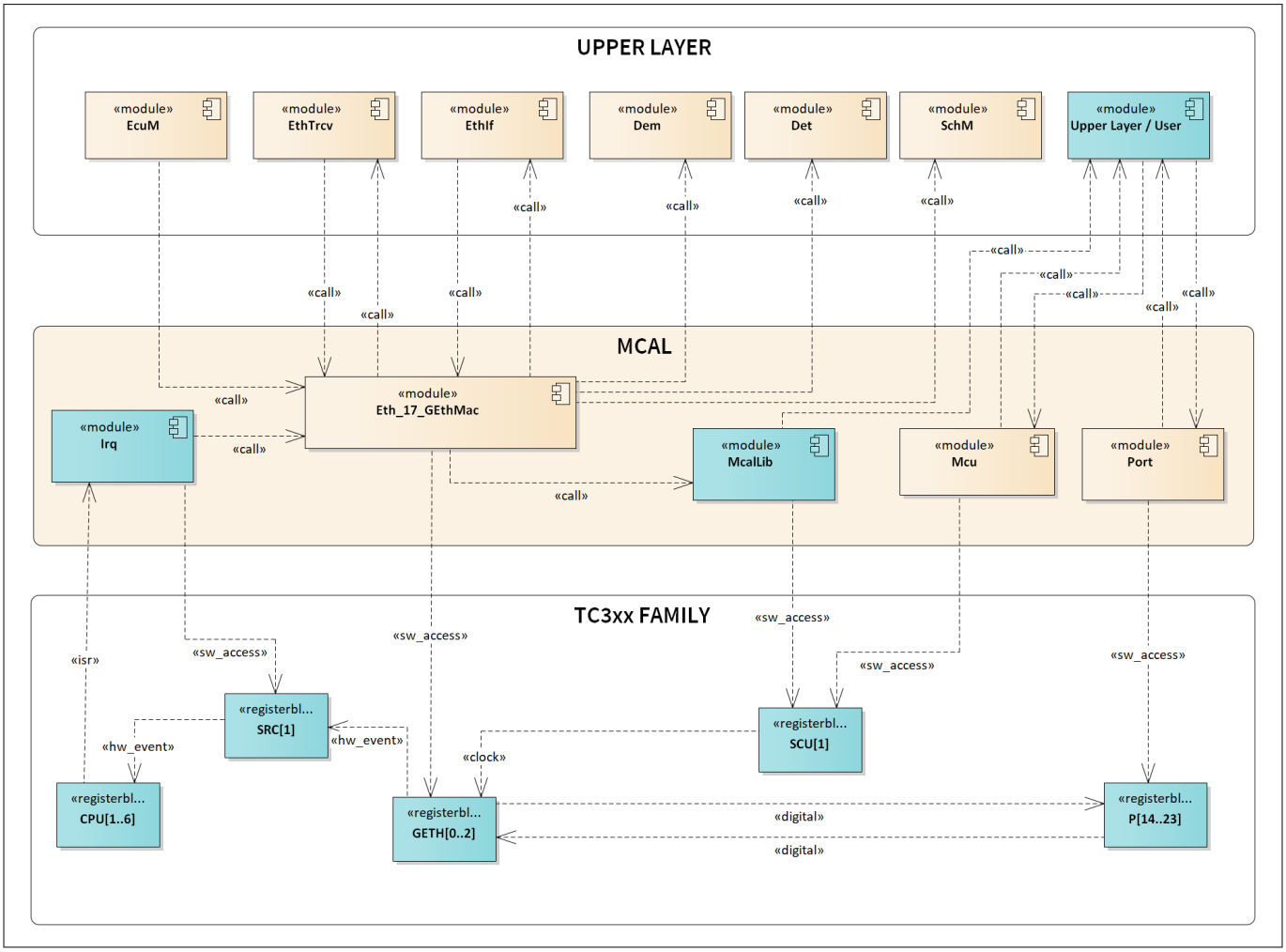

Figure 1 Mapping of hardware-software interfaces

1.1.2.1 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 either by the IRQ driver or 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.

1.1.2.2 GETH: primary hardware peripheral

Hardware functional features

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, 100 and 1000 Mbps
- 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 networked 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
- Single DMA channel and single MTL queue 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 be configured upto 4 queues. Each queue can be configured of different size in multiples of 256 bytes. ETH driver is designed to configure one transmit queue of size 4 Kbyte and one receive queue of size 8 Kbyte .
- Standard ETH of frame size 1518 bytes
- 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:

- Single and double VLAN tagged frames
- Jumbo frames
- Source address filters are not supported

- Multiple DMA channels
- Multiple priority based for queues
- Loopback mode
- Energy efficient ETH (EEE)
- Pulse per second output

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

1.1.2.3 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.

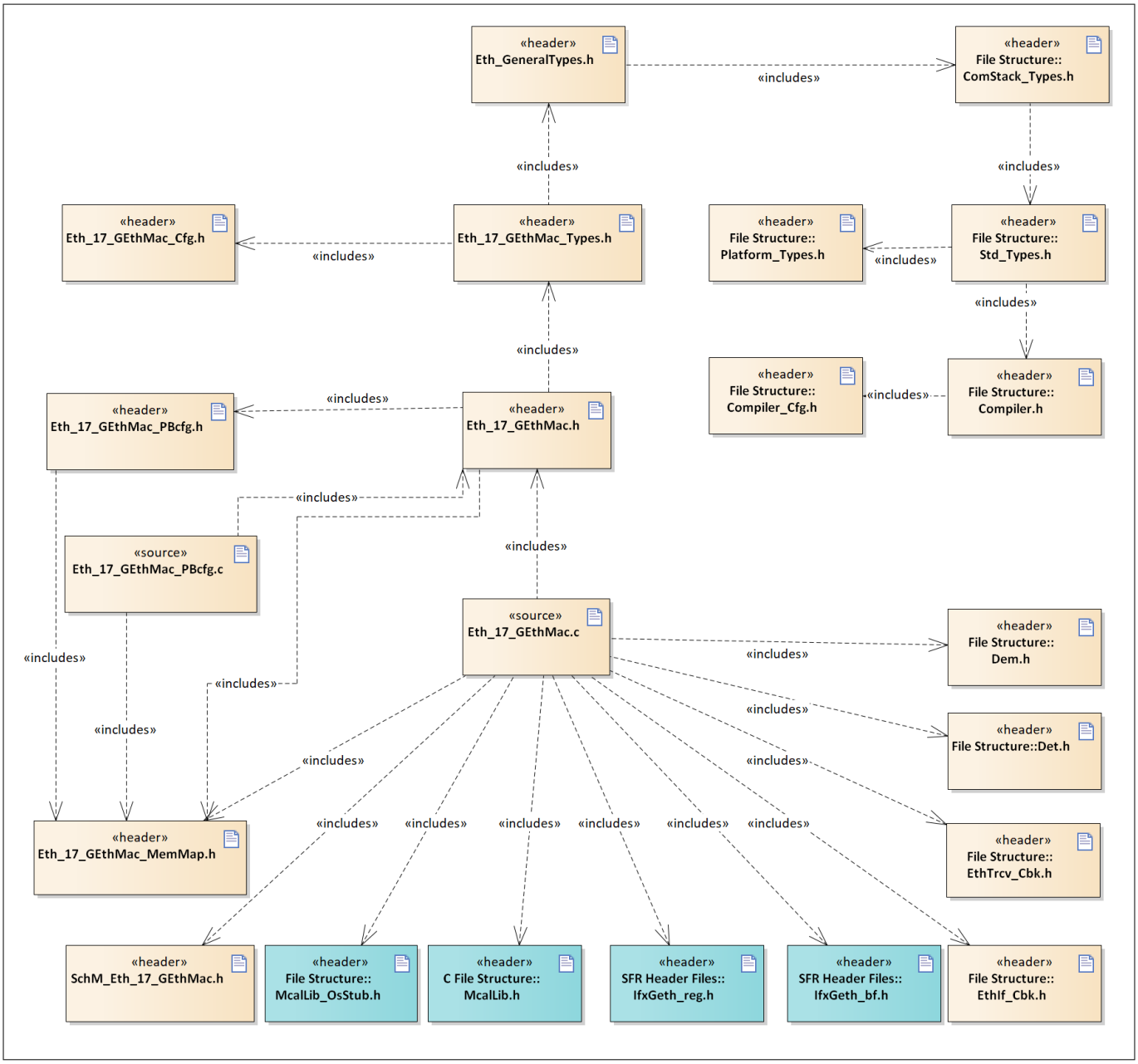

Figure 2 Eth_17_GEthMac_C_File_Structure-1.png

Table 2 C file structure

File name	Description
ComStack_Types.h	Type Definition for Com stack
Compiler.h	Provides abstraction from compiler-specific keywords
Compiler_Cfg.h	Configuration header file for compiler abstraction
Dem.h	Provides the exported interfaces of Diagnostic Event Manager
Det.h	Provides the exported interfaces of Development Error Tracer
EthIf_Cbk.h	Contains the declarations of the callback functions to ETH Interface (EthIf) module
EthTrcv_Cbk.h	Contains the declarations of the callback functions to ETH Transceiver (EthTrcv) module
Eth_17_GEthMac.c	File (Static) containing implementation of APIs
Eth_17_GEthMac.h	Header file (Static) defining prototypes of data structures, APIs and interrupt handlers
Eth_17_GEthMac_Cfg.h	Header file (Generated) containing constants and pre-processor macros as #defines
Eth_17_GEthMac_MemMap.h	File (Static) containing the memory section definitions used by the ETH driver
Eth_17_GEthMac_PBcfg.c	File (Generated) containing definition of the configuration data structures
Eth_17_GEthMac_PBcfg.h	File (Generated) containing declaration of the post-build configuration data structures of ETH driver
Eth_17_GEthMac_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.
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Eth_17_GEthMac.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.

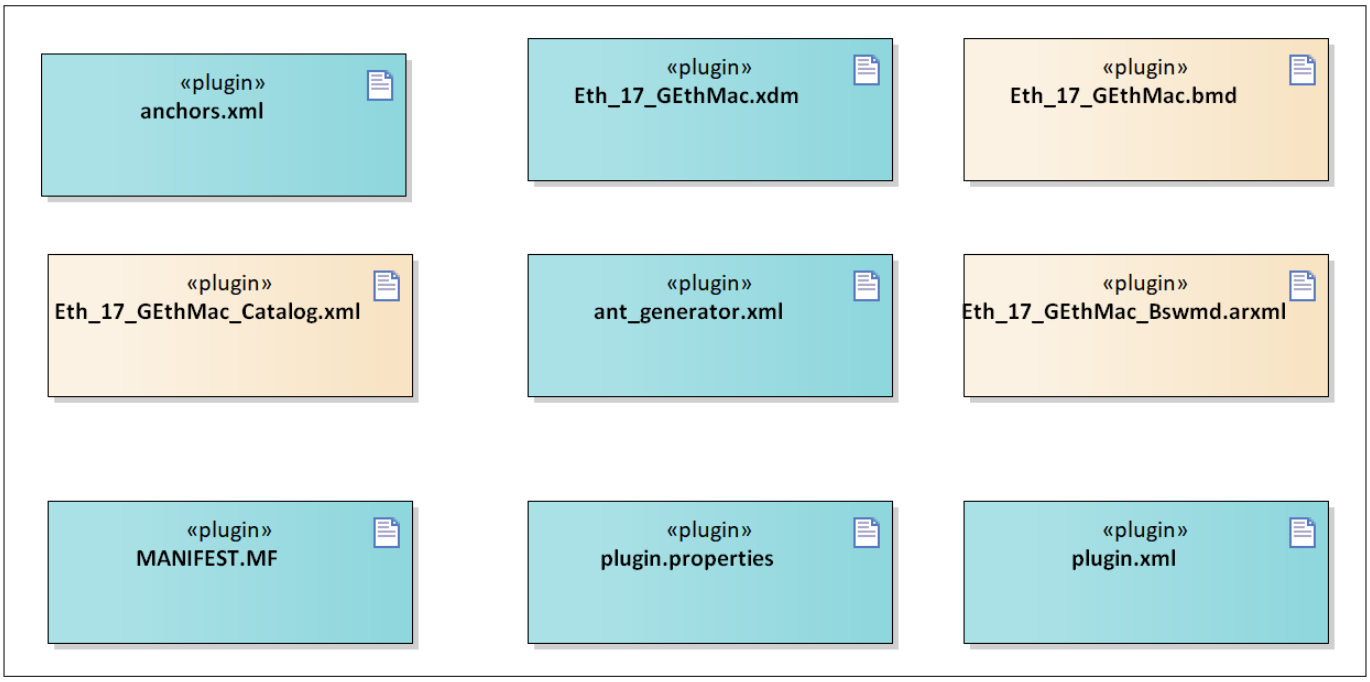

Figure 3 **Eth_17_GEthMac_Code_Generator_Plugin_Files-1.png**

Table 3 **Code generator plugin files**

File name	Description
`Eth_17_GEthMac.bmd`	AUTOSAR format XML data model schema file
`Eth_17_GEthMac.xdm`	Tresos format XML data model schema file
`Eth_17_GEthMac_Bswmd.arxml`	AUTOSAR format module description file
`Eth_17_GEthMac_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 the 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 upper layers, a hardware independent interface, to the ETH communication system comprising multiple different ETH controllers.

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

- **ETH transceiver driver (EthTrcv)**

  The EthTrcv module is part of the AUTOSAR stack that provides 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_Cbk.c` and `EthTrcv_Cbk.h` are provided as stub code and needs to be replaced with complete EthTrcv 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_GEthMac_MemMap.h`. The 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. 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_GETHMAC_START_SEC_VAR_CLEARED_QM_CORE0_UNSPECIFIED
 /* User Pragma here */
 #undef ETH_17_GETHMAC_START_SEC_VAR_CLEARED_QM_CORE0_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMAC_STOP_SEC_VAR_CLEARED_QM_CORE0_UNSPECIFIED
 /* User Pragma here */
 #undef ETH_17_GETHMAC_STOP_SEC_VAR_CLEARED_QM_CORE0_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMAC_START_SEC_VAR_TXBUFFER_CLEARED_QM_CORE0_32
 /* User Pragma here */
 #undef ETH_17_GETHMAC_START_SEC_VAR_TXBUFFER_CLEARED_QM_CORE0_32
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMAC_STOP_SEC_VAR_TXBUFFER_CLEARED_QM_CORE0_32
 /* User Pragma here */
 #undef ETH_17_GETHMAC_STOP_SEC_VAR_TXBUFFER_CLEARED_QM_CORE0_32
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMAC_START_SEC_VAR_RXBUFFER_CLEARED_QM_CORE0_32
 /* User Pragma here */
 #undef ETH_17_GETHMAC_START_SEC_VAR_RXBUFFER_CLEARED_QM_CORE0_32
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMAC_STOP_SEC_VAR_RXBUFFER_CLEARED_QM_CORE0_32
 /* User Pragma here */
 #undef ETH_17_GETHMAC_STOP_SEC_VAR_RXBUFFER_CLEARED_QM_CORE0_32
 #undef MEMMAP_ERROR

/**** ETH MODULE CONFIG DATA ****/
#elif defined ETH_17_GETHMAC_START_SEC_CONFIG_DATA_QM_GLOBAL_UNSPECIFIED
 /* User Pragma here */
 #undef ETH_17_GETHMAC_START_SEC_CONFIG_DATA_QM_GLOBAL_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMAC_STOP_SEC_CONFIG_DATA_QM_GLOBAL_UNSPECIFIED
 /* User Pragma here */
 #undef ETH_17_GETHMAC_STOP_SEC_CONFIG_DATA_QM_GLOBAL_UNSPECIFIED
 #undef MEMMAP_ERROR

/**** CODE SECTION ****/
#elif defined ETH_17_GETHMAC_START_SEC_CODE_QM_GLOBAL
 /* User Pragma here */
 #undef ETH_17_GETHMAC_START_SEC_CODE_QM_GLOBAL
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMAC_STOP_SEC_CODE_QM_GLOBAL
 /* User Pragma here */
 #undef ETH_17_GETHMAC_STOP_SEC_CODE_QM_GLOBAL
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMAC_START_SEC_CODE_QM_LOCAL
 /* User Pragma here */
 #undef ETH_17_GETHMAC_START_SEC_CODE_QM_LOCAL
 #undef MEMMAP_ERROR
#elif defined ETH_17_GETHMAC_STOP_SEC_CODE_QM_LOCAL
```

```
  /* User Pragma here */
  #undef ETH_17_GETHMAC_STOP_SEC_CODE_QM_LOCAL
  #undef MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Eth_17_EthMac_MemMap.h, wrong pragma command"
#endif
```

- **DET**

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

- **DEM**

  The DEM module is a part of the AUTOSAR stack that handles all the production errors reported by the BSW modules. The ETH driver reports all the production errors to the DEM modules through the `Dem_ReportErrorStatus()` API. The user of the ETH driver must process all the production errors (fail / pass) reported to the DEM module through the `Dem_ReportErrorStatus()` API.

  The `Dem.h` and `Dem.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DEM 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_GEthMac.c` file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the ETH driver are:

  - TransmitData

  - UpdateGlobalTime

  The `SchM_Eth_17_GEthMac.h` and `SchM_Eth_17_GEthMac.c` files are provided in the MCAL package as an example code and needs to 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. A sample implementation of the SchM functions are shown as follows:

```
/**** Sample implementation of SchM_Eth_17_GEthMac.c ****/
#include "Os.h"

void SchM_Enter_Eth_17_GEthMac_TransmitData(void)
{
 SuspendAllInterrupts();/* Suspend CPU core interrupt */
}

void SchM_Exit_Eth_17_GEthMac_TransmitData(void)
{
 ResumeAllInterrupts();/* Resume CPU core interrupt */
}

void SchM_Enter_Eth_17_UpdateGlobalTime(void)
{
 SuspendAllInterrupts();/* Suspend CPU core interrupt */
}

void SchM_Exit_Eth_17_UpdateGlobalTime(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 and controller mode change through notification functions of the EthIf module and successful Media independent accesses(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 is 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 ETH controller id passed as parameter while invoking an API, belongs to the same core.
- DETs are raised in case APIs are invoked with mismatch of core and controller id.
- 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.

  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 initialization. The fGETH defines the application clock frequency for the Gigabit ETH Kernel. The fGETH is independent to 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` from 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:

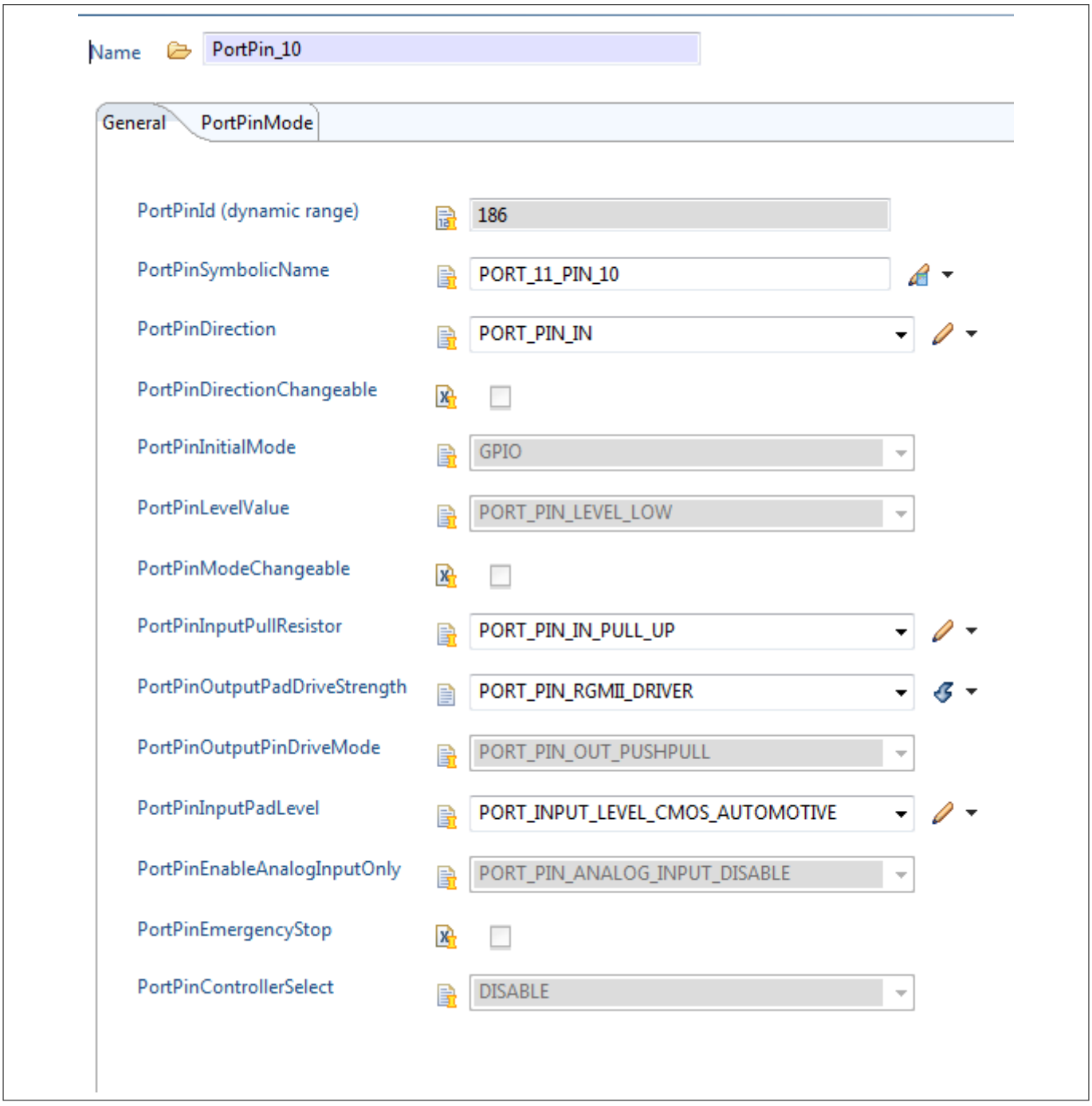

Figure 4 Input pin configuration

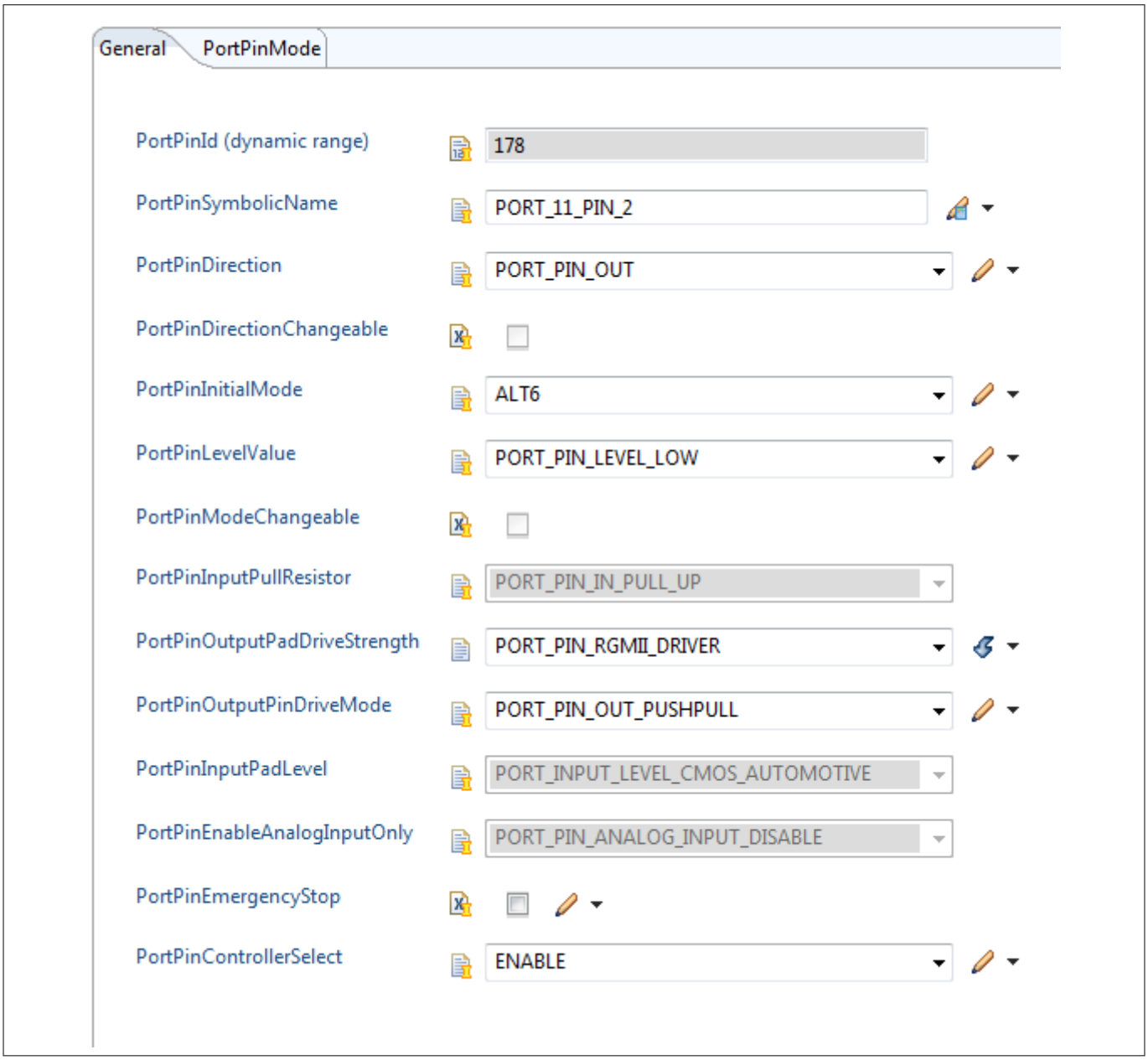

Figure 5 Output pin configuration

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 address accessed by DMA 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 DMA channel-0 for controller Id 0**

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

  In the ETH controller, the service request line number SRC_GETH2 is used for transmission complete interrupt.

  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 is as follows:

```
/* Include Eth header file Eth.h */
#include "Eth_17_GEthMac.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_GEthMac_TxDmaCh0IrqHdlr(0);
}
```

- **Packet receive complete interrupt from DMA channel-0 for controller Id 0**

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

  In the ETH controller, service request line number SRC_GETH6 is used for receive complete interrupt.

  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 is as follows:

```
/* Include Eth header file Eth.h */
#include "Eth_17_GEthMac.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_GEthMac_RxDmaCh0IrqHdlr(0);
}
```

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 `EthCtrlRxBufLenByte` parameter 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 for transmission and SRC_GETH6 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 to Core0. (If this is not configured, by default the controller is assumed to be allocated to the master core as per the Resource Manager module).

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_GEthMac.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_GEthMac_Init(&Eth_Config);

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

/*To check if Ethernet driver is initialized for the current core*/
if(*(Eth_17_GEthMac_CoreInitStatus[CoreId]) == (uint32)ETH_STATE_INIT)
{
 /*Successful initialization*/
}

/* Perform Ethernet transceiver initialization */
```

Setting controller mode

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

```
Eth_17_GEthMac_SetControllerMode(0, ETH_MODE_ACTIVE);
```

Set physical address (MAC address)

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

```
Eth_17_GEthMac_SetPhysAddr(0, &MacAddress[0]);
```

Get physical address (MAC address)

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

```
Eth_17_GEthMac_GetPhysAddr((0, &MacAddressRead[0]);
```

Configure ETH transceiver (PHY) device

The code sequence to configure ETH Transceiver (PHY) by using `Eth_17_GEthMac_ReadMii` and `Eth_17_GEthMac_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_GEthMac_ReadMii( 0, 1, MII_PHYSID1, &phy_id1);
Eth_17_GEthMac_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_GEthMac_WriteMii(0, 1, MII_BMCR, reg_value);
```

Transmit frame

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

```
/* Request the data buffer for frame Transmission */
Eth_17_GEthMac_ProvideTxBuffer(0, BufIdxPtr, BufPtr, LenBytePtr)

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

/* Transmit already filled buffer using the BuffIdx*/
Eth_17_GEthMac_Transmit(0, BufIdx,FrameType, TxConfirmation, LenByte,
PhysAddrPtr)
 ..............................

/* Call TX confirmation API to check if the packet transmission is completed */
Eth_17_GEthMac_TxConfirmation(0);
```

Receive frame

The received ETH frames are transferred to the driver buffer automatically. In polling mode, the application may call `Eth_17_GEthMac_Receive` API to get indication of all the filled buffers. In the interrupt mode, the ETH driver will call receive indication from receive ISR context. The code sequence to receive ETH frames in polling mode is as follows:

```
/* Poll for received frames */
Eth_17_GEthMac_Receive(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_GEthMac_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_GEthMac_WriteMii()` and `Eth_17_GEthMac_ReadMii()` API services provided by ETH driver.

But 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 MAC cannot finish PHY interface (MII) selection. Otherwise, this will cause unpredictable behavior.

Hence the ETH controller initialization by ETH driver is split into two parts.

Part 1: The `Eth_17_GEthMac_Init()` API enables the module and prepares the MDIO interface.

Note: The `Eth_17_GEthMac_Init()` API will not choose the MII interface because it is probable that the clock from the PHY is not yet available at the MAC input pin. The `Eth_17_GEthMac_Init()` API initializes 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 then `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_GEthMac_SetControllerMode()` API performs the following actions:

- Flush the transmit and receive queue 0 (since only queue 0 is used in the driver design)
- 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_GEthMac_SetControllerMode()` API).
  This action is performed only once when this `Eth_17_GEthMac_SetControllerMode()` API is called for the first time after `Eth_17_GEthMac_Init()` API.

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

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

Note:

- *Since `Eth_17_GEthMac_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_GEthMac_Init()` API enables the clock of 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 then `ETH_E_ACCESS` error is reported as PREFAILED for the failed controller and core initialization status is set to unsuccessful.*
- *`Eth_17_GEthMac_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_GETHMAC_E_NOT_INITIALIZED`, if indexed controller initialization or core initialization is unsuccessful.*

1.1.5.2 Eth_17_GEthMac_SetControllerMode API implemented as synchronous

The AUTOSAR specification mentions the `Eth_17_GEthMac_SetControllerMode()` API as asynchronous function. Since the ETH controller in TC3xx supports the change of the controller mode instantaneously, the `Eth_17_GEthMac_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 DET error check is enabled:

- The `Eth_17_GEthMac_SetControllerMode()` API will check the `CtrlMode` parameter for being valid. If the check fails, the API will report the `ETH_17_E_INV_PARAM DET` and return `E_NOT_OK`.
- The `Eth_17_GEthMac_UpdatePhysAddrFilter()` API will check the `Action` parameter for being valid. If the check fails, the API will report `ETH_17_E_INV_PARAM DET` and return `E_NOT_OK`.
- The `Eth_17_GEthMac_Transmit()` API will check the `LenByte` parameter for being valid. If the value of `LenByte` is more than granted through the `Eth_17_GEthMac_ProvideTxBuffer()`, then the API `Eth_17_GEthMac_Transmit()` will report `ETH_17_E_INV_PARAM DET` and return `E_NOT_OK`. If the value of `LenByte` is less than granted through the `Eth_17_GEthMac_ProvideTxBuffer()` API, then the API `Eth_17_GEthMac_Transmit()` will proceed with transmission of ETH packet of length `LenByte`.
- The APIs `Eth_17_GEthMac_EnableEgressTimeStamp()` and `Eth_17_GEthMac_GetEgressTimeStamp()` will check the `BufIdx` parameter for being valid. If the `BufIdx` is within total number of buffer configured and if `BufIdx` same as the one allocated to application by ETH driver from the `Eth_17_GEthMac_ProvideTxBuffer()` API. If aforementioned checks fail, the API will report `ETH_17_E_INV_PARAM` development error.

1.1.5.4 Timer support APIs report ETH_E_ACCESS production error

Any timer update operation in the ETH controller needs a feedback from register bits to ensure that requested operation is completed. If the requested operation is not completed within a configured time limit, time out occurs and `ETH_E_ACCESS` production error is reported from following APIs:

- `Eth_17_GEthMac_SetGlobalTime`
- `Eth_17_GEthMac_SetCorrectionTime`

Note: The `EthTimeoutCount` configuration parameter is used to configure the time limit.

1.1.5.5 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_GEthMac_Init()` API is invoked from a core to which no controller is allocated then `ETH_17_GETHMAC_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_GETHMAC_E_INV_CTRL_IDX` development error is reported.

1.1.5.6 Specific hardware features used for nominal operation of ETH driver

To implement the ETH driver as per the product requirement, 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 need 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 need 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 has transmit/receive FIFO space shared by multiple queues and multiple transmit/receive DMA channels. The ETH driver uses one transmit queue of maximum FIFO size and one transmit DMA channel for the transmit operation and one receive queue of maximum FIFO size and one receive DMA channel for the receive operation.
- Configuration of Transmit and Receive timing skew in the RGMII mode:

  The ETH driver provides the `EthSkewTxClockDelay` and `EthSkewRxClockDelay` configuration parameters to configure 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 an Ethernet controller. The minimum 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.

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.

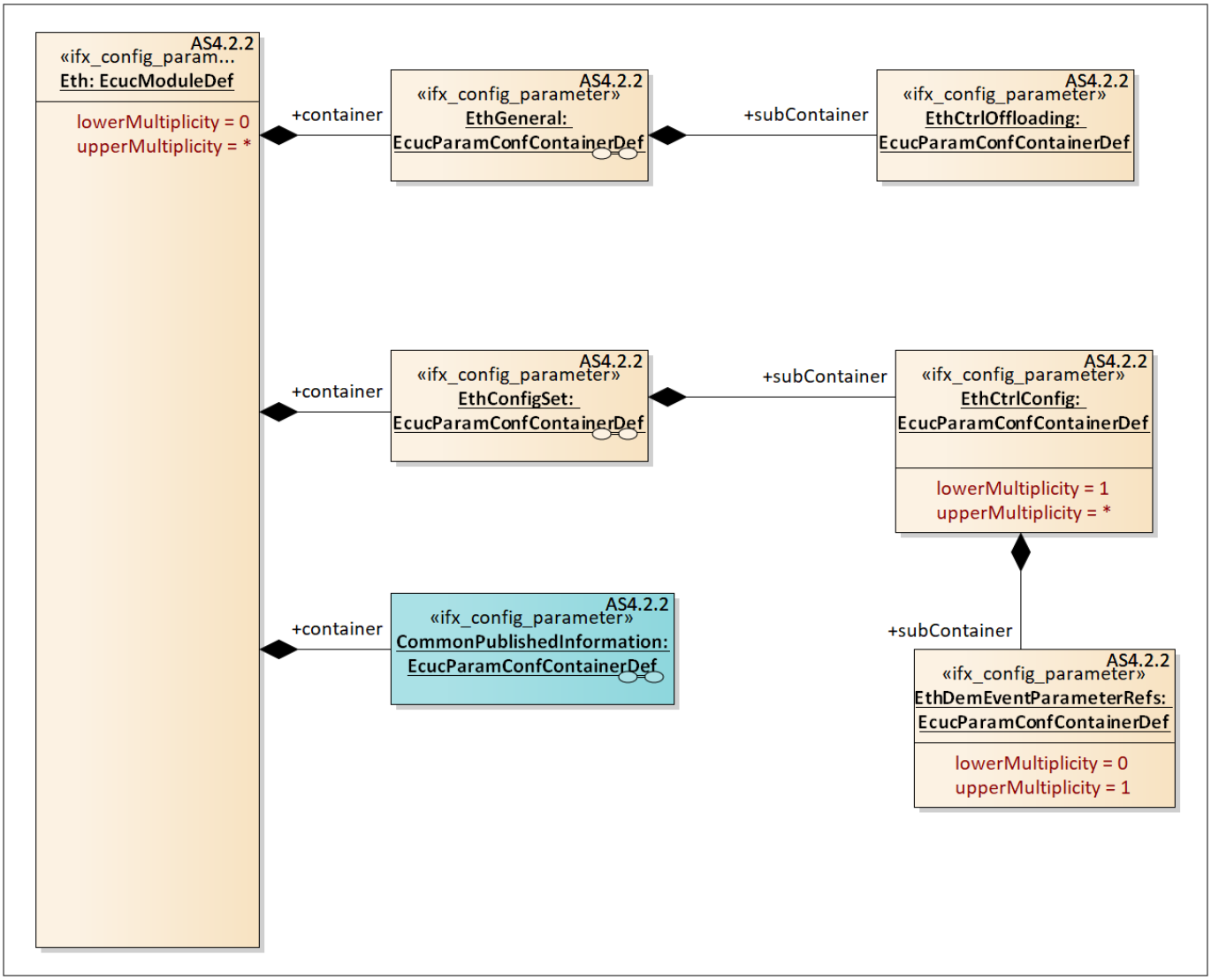

Figure 6 Container hierarchy along with their configuration parameters

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 4 Specification for ArMajorVersion

Name	`ArMajorVersion`

Table 4 Specification for ArMajorVersion (continued)

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 4.2.2.		

1.3.1.1.2 ArMinorVersion

Table 5 Specification for ArMinorVersion

Name	`ArMinorVersion`		
Description	Provides the minor version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	2		
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 4.2.2.		

1.3.1.1.3 ArPatchVersion

Table 6 Specification for ArPatchVersion

Name	`ArPatchVersion`		
Description	Provides the patch version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		

Table 6	Specification for ArPatchVersion (continued)		
Default value	2		
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 4.2.2.		

1.3.1.1.4 ModuleId

Table 7	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 4.2.2.		

1.3.1.1.5 Release

Table 8	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 8 Specification for Release (continued)

Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.1.6 SwMajorVersion

Table 9 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 4.2.2.		

1.3.1.1.7 SwMinorVersion

Table 10 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

Table 10	Specification for SwMinorVersion (continued)
Dependency	-
Autosar Version	Applicable for Autosar version 4.2.2.

1.3.1.1.8 SwPatchVersion

Table 11 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 4.2.2.		

1.3.1.1.9 VendorApiInfix

Table 12 Specification for VendorApiInfix

Name	`VendorApiInfix`		
Description	Provides the VendorApiInfix.		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	GEthMac		
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 4.2.2.		

1.3.1.1.10 VendorId

Table 13 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 4.2.2.		

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.

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 14 Specification for EthCRSDVRMIIInput

Name	`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.*

Table 14 Specification for EthCRSDVRMIIInput (continued)

	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_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	EthPhyInterface		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.2 EthCarrierSenseMIIInput

Table 15 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_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	EthPhyInterface		

Table 15	Specification for EthCarrierSenseMIIInput (continued)
Autosar Version	Applicable for Autosar version 4.2.2.

1.3.1.4.3 EthCollisionMII

Table 16	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_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	EthPhyInterface		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.4 EthCtrlEnableCrcStripping

Table 17	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		

Table 17 Specification for EthCtrlEnableCrcStripping (continued)

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 4.2.2.		

1.3.1.4.5 EthCtrlEnableMii

Table 18 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 4.2.2.		

1.3.1.4.6 EthCtrlEnableRxInterrupt

Table 19 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.

Table 19 Specification for EthCtrlEnableRxInterrupt (continued)

	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	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.7 EthCtrlEnableTxInterrupt

Table 20 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 4.2.2.		

1.3.1.4.8 EthCtrlIdx

Table 21 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
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.9 EthCtrlPhyAddress

Table 22 Specification for EthCtrlPhyAddress

Name	`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, 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 to match the Vendor ID of MAC is Infineon's ID. 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**	-

Table 22 Specification for EthCtrlPhyAddress (continued)

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.10 EthCtrlRxBufLenByte

Table 23 Specification for EthCtrlRxBufLenByte

Name	`EthCtrlRxBufLenByte`		
Description	Limits the maximum receive buffer length (frame length) in bytes. This configured length includes ETH frame header and frame checksum (total 18 bytes). *Note 1: During configuration, the buffer size should be (Header + Payload data + CRC).* *Note 2: Maximum length of one ETH frame packet is 1522. (1504 (Payload) + 14 (Header) + 4 (CRC) = 1522).* *Note 3: The default value for this parameter is kept as 1522 because the maximum size of one ETH frame is 1522.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 1522		
Default value	1522		
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 4.2.2.		

1.3.1.4.11 EthCtrlTxBufLenByte

Table 24 Specification for EthCtrlTxBufLenByte

Name	`EthCtrlTxBufLenByte`
Description	Limits the maximum transmit buffer length (frame length) in bytes. This configured length includes ETH frame header and frame checksum (Total 18 Bytes). *Note 1: During configuration, the buffer size shall be (Header + Payload data + CRC).* *Note 2: Maximum length of one ETH frame packet is 1522. (1504 (Payload) +14 (Header) + 4 (CRC) = 1522).*

Table 24 Specification for EthCtrlTxBufLenByte (continued)

	Note 3: Since the maximum size of one ETH frame is 1522, the default value for this parameter is kept as 1522.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 1522		
Default value	1522		
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 4.2.2.		

1.3.1.4.12 EthMDCClockFrequency

Table 25 Specification for EthMDCClockFrequency

Name	`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*

Table 25 Specification for EthMDCClockFrequency (continued)

	2.5 MHz considering that, any IEEE standard PHY device shall support MDC clock value up to 2.5 MHz.		
Multiplicity	1..1	**Type**	EcucFloatParamDef
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 4.2.2.		

1.3.1.4.13 EthMdioAlternateInput

Table 26 Specification for EthMdioAlternateInput

Name	`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_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 4.2.2.		

1.3.1.4.14 EthOpMode

Table 27 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 4.2.2.		

1.3.1.4.15 EthPhyInterface

Table 28 Specification for EthPhyInterface

Name	`EthPhyInterface`		
Description	Specifies the interface used between the MAC and PHY. The availability of modes is dependent on the micro-controller package. *Note 1: Since the commonly used application is RMII, the default value of this parameter is selected as RMII.* *Note 2: Availability of the Ethernet controller speed with respect to the PHY interface 1000/100/10MBPS depends on the device.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	MII: Media Independent Interface RGMII: Reduced Gigabit Media Independent Interface RMII: Reduced Media Independent Interface		
Default value	RMII		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL

Table 28	Specification for EthPhyInterface (continued)
Dependency	-
Autosar Version	Applicable for Autosar version 4.2.2.

1.3.1.4.16 EthRecDataValidMIIInput

Table 29	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_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	EthPhyInterface		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.17 EthReceiveData0Input

Table 30	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

Table 30 Specification for EthReceiveData0Input (continued)

Range	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	EthPhyInterface		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.18 EthReceiveData1Input

Table 31 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_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	EthPhyInterface		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.19 EthReceiveData2Input

Table 32 Specification for EthReceiveData2Input

Name	`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_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	EthPhyInterface		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.20 EthReceiveData3Input

Table 33 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_PXy_Yz: Xy and Yz are port and pin number respectively which depends on the device variant.		
Default value	ALTx_SELECT_NONE		

Table 33 Specification for EthReceiveData3Input (continued)

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthPhyInterface		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.21 EthRefClkRMIIInput

Table 34 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_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	EthPhyInterface		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.22 EthRxBufTotal

Table 35 Specification for EthRxBufTotal

Name	`EthRxBufTotal`
Description	Configures the number of receive buffers.

Table 35 Specification for EthRxBufTotal (continued)

	*Note 1: Total buffer size in the RAM that is reserved by the ETH driver for receive packets is calculated as EthCtrlRxBufLenByte*EthRxBufTotal.* *Note 2: By default, the number of buffers reserved is kept as 4 and, therefore, up to 4 packets can be received without overflow.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
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 4.2.2.		

1.3.1.4.23 EthRxErrMIIInput

Table 36 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_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	EthPhyInterface		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.24 EthRxclkInput

Table 37 Specification for EthRxclkInput

Name	`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_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	EthPhyInterface		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.25 EthSkewRxClockDelay

Table 38 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

Table 38	Specification for EthSkewRxClockDelay (continued)
Dependency	EthPhyInterface
Autosar Version	Applicable for Autosar version 4.2.2.

1.3.1.4.26 EthSkewTxClockDelay

Table 39 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		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	EthPhyInterface		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.27 EthSpeed

Table 40 Specification for EthSpeed

Name	`EthSpeed`		
Description	Selects the speed of the ETH controller. *Note 1: Since the commonly used application is 100 Mbps, therefore, the default value of this parameter is selected as 100 Mbps.* *Note 2: Availability of the 1000/100/10 MBPS depends on the device.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ETH_1000MBPS: Data transfer rate is 1000 Mbps ETH_100MBPS: Data transfer rate is 100 Mbps ETH_10MBPS: Data transfer rate is 10 Mbps		
Default value	ETH_100MBPS		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

Table 40 Specification for EthSpeed (continued)

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.4.28 EthTxBufTotal

Table 41 Specification for EthTxBufTotal

Name	`EthTxBufTotal`		
Description	Configures the number of transmit buffers. *Note 1: Total buffer size in the RAM that is reserved by the ETH driver for transmission packet is calculated as EthCtrlTxBufLenByte*EthTxBufTotal.* *Note 2: By default, the number of buffers reserved is kept as 4 and, therefore, up to 4 transmit request can be made without waiting for transmit completion.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
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 4.2.2.		

1.3.1.4.29 EthTxClockMIIInput

Table 42 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.

Table 42 Specification for EthTxClockMIIInput (continued)

Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	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	EthPhyInterface		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.5 Container: EthCtrlOffloading

Configuration of hardware checksum offloading features.

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 parameter EthCtrlEnableOffloadChecksumIPV4, EthCtrlEnableOffloadChecksumUDP, EthCtrlEnableOffloadChecksumTCP, EthCtrlEnableOffloadChecksumICMP would enable the checksum offload functionality for IPV4, UDP, TCP and ICMP. This is a deviation to AUTOSAR requirements.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.5.1 EthCtrlEnableOffloadChecksumICMP

Table 43 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 GETHMAC hardware supports either checksum offloading can be enabled for all types of packets or disable all types of packets.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		

Table 43 Specification for EthCtrlEnableOffloadChecksumICMP (continued)

Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	EthCtrlEnableOffloadChecksumUDP, EthCtrlEnableOffloadChecksumTCP, EthCtrlEnableOffloadChecksumIPv4		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.5.2 EthCtrlEnableOffloadChecksumIPv4

Table 44 Specification for EthCtrlEnableOffloadChecksumIPv4

Name	`EthCtrlEnableOffloadChecksumIPv4`		
Description	It enables/disables checksum offloading of IPv4, TCP, UDP, ICMP frames for both transmission (that is, calculation and insertion of checksum in the transmitted frames at hardware level) and reception (That is checksum calculation at hardware level for received frames to check for checksum mismatch). *Note: This is a deviation from the AUTOSAR requirement. The GETHMAC hardware supports either checksum offloading can be enabled for all types of packets or disable all types of packets.*		
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	EthCtrlEnableOffloadChecksumUDP, EthCtrlEnableOffloadChecksumTCP, EthCtrlEnableOffloadChecksumICMP		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.5.3 EthCtrlEnableOffloadChecksumTCP

Table 45 Specification for EthCtrlEnableOffloadChecksumTCP

Name	`EthCtrlEnableOffloadChecksumTCP`

Table 45	Specification for EthCtrlEnableOffloadChecksumTCP (continued)		
Description	It 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, checksum calculation at the hardware level for received frames to check for checksum mismatch). *Note: This is a deviation from the AUTOSAR requirement. The GETHMAC hardware supports either checksum offloading can be enabled for all types of packets or disable all types of packets.*		
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	EthCtrlEnableOffloadChecksumUDP, EthCtrlEnableOffloadChecksumIPv4, EthCtrlEnableOffloadChecksumICMP		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.5.4 EthCtrlEnableOffloadChecksumUDP

Table 46	Specification for EthCtrlEnableOffloadChecksumUDP		
Name	`EthCtrlEnableOffloadChecksumUDP`		
Description	It enables/disables checksum offloading of IPv4, TCP, UDP, ICMP frames for both transmission (that is, calculation and insertion of checksum in the transmitted frames at hardware level) and reception (That is checksum calculation at hardware level for received frames to check for checksum mismatch). *Note: This is a deviation from the AUTOSAR requirement. The GETHMAC hardware supports either checksum offloading can be enabled for all types of packets or disable all types of packets.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 46	Specification for EthCtrlEnableOffloadChecksumUDP (continued)		
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	EthCtrlEnableOffloadChecksumTCP, EthCtrlEnableOffloadChecksumIPv4, EthCtrlEnableOffloadChecksumICMP		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.6 Container: EthDemEventParameterRefs

This is a container for the references to the DemEventParameter elements, which are invoked using the Dem_ReportErrorStatus() 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.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

1.3.1.6.1 ETH_E_ACCESS

Table 47	Specification for ETH_E_ACCESS		
Name	`ETH_E_ACCESS`		
Description	Provides preference to the DemEventParameter, which is issued when the error controller access fails.		
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 4.2.2.		

1.3.1.6.2 ETH_E_ALIGNMENT

Table 48	Specification for ETH_E_ALIGNMENT
Name	`ETH_E_ALIGNMENT`

Table 48 Specification for ETH_E_ALIGNMENT (continued)

Description	Provides reference to the DemEventParameter, which is issued when the error 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
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.6.3 ETH_E_CRC

Table 49 Specification for ETH_E_CRC

Name	`ETH_E_CRC`		
Description	Provides reference to the DemEventParameter, which is issued when the error CRC failure 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 4.2.2.		

1.3.1.6.4 ETH_E_LATECOLLISION

Table 50 Specification for ETH_E_LATECOLLISION

Name	`ETH_E_LATECOLLISION`

Table 50	Specification for ETH_E_LATECOLLISION (continued)		
Description	Provides reference to the DemEventParameter, which is issued when the ETH 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
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.6.5 ETH_E_MULTIPLECOLLISION

Table 51	Specification for ETH_E_MULTIPLECOLLISION		
Name	`ETH_E_MULTIPLECOLLISION`		
Description	Provides reference to the DemEventParameter, which is issued when the ETH 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 4.2.2.		

1.3.1.6.6 ETH_E_OVERSIZEFRAME

Table 52 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
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.6.7 ETH_E_RX_FRAMES_LOST

Table 53 Specification for ETH_E_RX_FRAMES_LOST

Name	ETH_E_RX_FRAMES_LOST		
Description	Provides reference to the DemEventParameter, which is issued when the error 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 4.2.2.		

1.3.1.6.8 ETH_E_SINGLECOLLISION

Table 54 Specification for ETH_E_SINGLECOLLISION

Name	ETH_E_SINGLECOLLISION		
Description	Provides reference to the DemEventParameter, which is issued when the ETH 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
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.6.9 ETH_E_UNDERSIZEFRAME

Table 55 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 4.2.2.		

1.3.1.7 Container: EthGeneral

General configuration of the ETH driver module
Post-Build Variant Multiplicity: -
Multiplicity Configuration Class: -

1.3.1.7.1 EthDevErrorDetect

Table 56 Specification for EthDevErrorDetect

Name	`EthDevErrorDetect`		
Description	Enables or disables the DET detection and reporting.		
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 4.2.2.		

1.3.1.7.2 EthDmaSwResetWaitCycle

Table 57 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

Table 57	Specification for EthDmaSwResetWaitCycle (continued)
Dependency	-
Autosar Version	Applicable for Autosar version 4.2.2.

1.3.1.7.3 EthGetDropCountApi

Table 58	Specification for EthGetDropCountApi		
Name	`EthGetDropCountApi`		
Description	Enables or disables the Eth_17_GEthMac_GetDropCount() 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 4.2.2.		

1.3.1.7.4 EthGetEtherStatsApi

Table 59	Specification for EthGetEtherStatsApi		
Name	`EthGetEtherStatsApi`		
Description	Enables or disables the Eth_17_GEthMac_GetEtherStats() 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**	-

Table 59	Specification for EthGetEtherStatsApi (continued)		
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.7.5 EthGlobalTimeSupport

Table 60	Specification for EthGlobalTimeSupport		
Name	`EthGlobalTimeSupport`		
Description	Enables or disables the following GlobalTime APIs. - Eth_17_GEthMac_GetCurrentTime() - Eth_17_GEthMac_EnableEgressTimeStamp() - Eth_17_GEthMac_GetEgressTimeStamp() - Eth_17_GEthMac_GetIngressTimeStamp() - Eth_17_GEthMac_SetCorrectionTime() - Eth_17_GEthMac_SetGlobalTime() *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 4.2.2.		

1.3.1.7.6 EthIndex

Table 61	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.

Table 61 Specification for EthIndex (continued)

	Note: Since there is only one instance present in most of the TC3xx devices, 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 4.2.2.		

1.3.1.7.7 EthInitApiMode

Table 62 Specification for EthInitApiMode

Name	`EthInitApiMode`		
Description	Defines the mode in which the Init() API is used. *Note: Since 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 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	EthRuntimeApiMode		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.7.8 EthMainFunctionPeriod

Table 63 Specification for EthMainFunctionPeriod

Name	`EthMainFunctionPeriod`		
Description	Specifies the period of main function Eth_17_GEthMac_MainFunction in seconds. ETH driver does not require this information but the BSW schedule uses this information.		
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 4.2.2.		

1.3.1.7.9 EthMaxCtrlsSupported

Table 64 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**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar version 4.2.2.		

1.3.1.7.10 EthMultiCoreErrorDetect

Table 65 Specification for EthMultiCoreErrorDetect

Name	`EthMultiCoreErrorDetect`

Table 65	Specification for EthMultiCoreErrorDetect (continued)		
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 4.2.2.		

1.3.1.7.11 EthOperationFrequency

Table 66	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 value required to be initialized in the register bits GETH_MAC_SUB_SECOND_INCREMENT.B.SSINC to operate timer correctly. *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 4.2.2.		

1.3.1.7.12 EthPeripheralBusClock

Table 67 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 to program 2.5 MHz 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 4.2.2.		

1.3.1.7.13 EthRuntimeApiMode

Table 68 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.*		
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

Table 68	Specification for EthRuntimeApiMode (continued)
Dependency	-
Autosar Version	Applicable for Autosar version 4.2.2.

1.3.1.7.14 EthTimeoutCount

Table 69	Specification for EthTimeoutCount		
Name	`EthTimeoutCount`		
Description	Specifies the maximum waiting time in nanoseconds for hardware timeout errors. *Note: The maximum value is kept as the default value for this parameter.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	100 - 4294967295		
Default value	4294967295		
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 4.2.2.		

1.3.1.7.15 EthUpdatePhysAddrFilter

Table 70	Specification for EthUpdatePhysAddrFilter		
Name	`EthUpdatePhysAddrFilter`		
Description	Enables or disables the API, Eth_17_GEthMac_UpdatePhysAddrFilter. *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 70	Specification for EthUpdatePhysAddrFilter (continued)
Dependency	-
Autosar Version	Applicable for Autosar version 4.2.2.

1.3.1.7.16 EthVersionInfoApi

Table 71	Specification for EthVersionInfoApi		
Name	`EthVersionInfoApi`		
Description	Enables or disables the version info API, Eth_17_GEthMac_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 4.2.2.		

1.3.2 Functions - Type definitions

1.3.2.1 Eth_17_GEthMac_ConfigType

Table 72	Specification for Eth_17_GEthMac_ConfigType	
Syntax	`Eth_17_GEthMac_ConfigType`	
Type	Structure	
File	`Eth_GeneralTypes.h`	
Range	--	The elements of the data structure are specific to the micro-controller
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 4.2.2.	

1.3.2.2 Eth_BufIdxType

Table 73 Specification for Eth_BufIdxType

Syntax	`Eth_BufIdxType`	
Type	uint32	
File	`Eth_GeneralTypes.h`	
Range	0x00000000 - 0xFFFFFFFF	ETH buffer identifier
Description	ETH buffer identifier type	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.2.2.	

1.3.2.3 Eth_DataType

Table 74 Specification for Eth_DataType

Syntax	`Eth_DataType`	
Type	uint8	
File	`Eth_GeneralTypes.h`	
Range	0-255	One byte data
Description	This type defines the ETH data type used for data transmission and reception.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.2.2.	

1.3.2.4 Eth_FilterActionType

Table 75 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.
Description	The Eth_FilterActionType enumeration type describes the action to be taken for the MAC address given in *PhysAddrPtr of API function Eth_UpdatePhysAddrFilter()	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.2.2.	

1.3.2.5 Eth_FrameType

Table 76 Specification for Eth_FrameType

Syntax	`Eth_FrameType`	
Type	uint16	
File	`Eth_GeneralTypes.h`	
Range	0x0000 - 0xFFFF	ETH frame type used in the ETH frame header
Description	This type defines the ETH frame type used in the ETH frame header	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.2.2.	

1.3.2.6 Eth_ModeType

Table 77 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 4.2.2.	

1.3.2.7 Eth_RateRatioType

Table 78 Specification for Eth_RateRatioType

Syntax	`Eth_RateRatioType`	
Type	Structure	
File	`Eth_GeneralTypes.h`	
Range	Eth_TimeIntDiffType IngressTimeStampDelta	IngressTimeStampSync2 - IngressTimeStampSync1
	Eth_TimeIntDiffType OriginTimeStampDelta	OriginTimeStampSync2 - OriginTimeStampSync1
Description	Variables of this type are used to express frequency ratios	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.2.2.	

1.3.2.8 Eth_ReturnType

Table 79 Specification for Eth_ReturnType

Syntax	`Eth_ReturnType`	
Type	Enumeration	
File	`Eth_GeneralTypes.h`	
Range	0 - ETH_OK	Success
	1 - ETH_E_NOT_OK	General failure
	2 - ETH_E_NO_ACCESS	ETH hardware access failure
Description	ETH Driver specific return type	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.2.2.	

1.3.2.9 Eth_RxStatusType

Table 80 Specification for Eth_RxStatusType

Syntax	`Eth_RxStatusType`	
Type	Enumeration	
File	`Eth_GeneralTypes.h`	
Range	0 - ETH_RECEIVED	ETH frame has been received, no further frames available
	1 - ETH_NOT_RECEIVED	ETH frame has not been received, no further frames available
	2 - ETH_RECEIVED_MORE_DATA_AVAILABLE	ETH frame has been received, more frames are available
Description	Used as OUT parameter in the Eth_Receive() API that indicates whether a frame has been received and if so, whether more frames are available or frames are lost.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar version 4.2.2.	

1.3.2.10 Eth_TimeIntDiffType

Table 81 Specification for Eth_TimeIntDiffType

Syntax	`Eth_TimeIntDiffType`	
Type	Structure	
File	`Eth_GeneralTypes.h`	
Range	Eth_TimeStampType diff	Time difference
	boolean sign	Positive (True) / negative (False) time
Description	Variables of this type are used to express time differences	

Table 81 Specification for Eth_TimeIntDiffType (continued)

Source	AUTOSAR
Autosar Version	Applicable for Autosar version 4.2.2.

1.3.2.11 Eth_TimeStampQualType

Table 82 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 4.2.2.	

1.3.2.12 Eth_TimeStampType

Table 83 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
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	

Table 83	Specification for Eth_TimeStampType (continued)
Autosar Version	Applicable for Autosar version 4.2.2.

1.3.3 Functions - APIs

This section lists all the APIs of the ETH driver.

1.3.3.1 Eth_17_GEthMac_Init

Table 84 Specification for `Eth_17_GEthMac_Init` API

Syntax	`void Eth_17_GEthMac_Init ( const Eth_17_GEthMac_ConfigType * const CfgPtr )`	
Service ID	0x01	
Sync/Async	Synchronous	
ASIL Level	QM	
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 the core from which Eth_17_GEthMac_Init API is invoked.*	
Source	AUTOSAR	
Error handling	ETH_E_ACCESS, ETH_17_GETHMAC_E_INIT_FAILED, ETH_17_GETHMAC_E_CORE_NOT_CONFIGURED	
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 4.2.2.	

1.3.3.2 Eth_17_GEthMac_SetControllerMode

Table 85 Specification for `Eth_17_GEthMac_SetControllerMode` API

Syntax	`Std_ReturnType Eth_17_GEthMac_SetControllerMode ( const uint8 CtrlIdx, const Eth_ModeType CtrlMode )`	
Service ID	0x03	
Sync/Async	Synchronous	
ASIL Level	QM	
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 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_GEthMac_Init. - Action 2: It enables or disables the ETH controller with controller ID passed as the input parameter.	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_E_ACCESS, ETH_17_GETHMAC_E_INV_PARAM	
Configuration dependencies	-	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_CLC(r), GETH_DMA_CH_CONTROL(ex_w), GETH_DMA_CH_INTERRUPT_ENABLE(w), GETH_DMA_CH_RXDESC_LIST_ADDRESS(ex_w), GETH_DMA_CH_RXDESC_RING_LENGTH(ex_w), GETH_DMA_CH_RXDESC_TAIL_POINTER(w), GETH_DMA_CH_RX_CONTROL(rw), GETH_DMA_CH_STATUS(w), GETH_DMA_CH_TXDESC_LIST_ADDRESS(ex_w), GETH_DMA_CH_TXDESC_RING_LENGTH(ex_w), GETH_DMA_CH_TXDESC_TAIL_POINTER(w), GETH_DMA_CH_TX_CONTROL(rw), GETH_DMA_MODE(rw), GETH_DMA_SYSBUS_MODE(w), GETH_GPCTL(w), GETH_KRST0(rw), GETH_KRST1(ex_w), GETH_KRSTCLR(ex_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_SUB_SECOND_INCREMENT(ex_w),	

Table 85 Specification for `Eth_17_GEthMac_SetControllerMode` API (continued)

	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(ex_w), GETH_MMC_RX_INTERRUPT_MASK(ex_w), GETH_MMC_TX_INTERRUPT_MASK(ex_w), GETH_MTL_OPERATION_MODE(w), GETH_MTL_RXQ0_OPERATION_MODE(w), GETH_MTL_RXQ_DMA_MAP0(w), GETH_MTL_TXQ0_OPERATION_MODE(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 4.2.2.

1.3.3.3 Eth_17_GEthMac_GetControllerMode

Table 86 Specification for `Eth_17_GEthMac_GetControllerMode` API

Syntax	`Std_ReturnType  Eth_17_GEthMac_GetControllerMode ( const uint8 CtrlIdx, Eth_ModeType * const CtrlModePtr )`	
Service ID	0x04	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
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_GETHMAC_E_INV_CTRL_IDX, ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_PARAM_POINTER	
Configuration dependencies	-	

Table 86	Specification for `Eth_17_GEthMac_GetControllerMode` API (continued)	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_DMA_CH_RX_CONTROL(r), GETH_DMA_CH_TX_CONTROL(r), GETH_MAC_CONFIGURATION(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 4.2.2.	

1.3.3.4 Eth_17_GEthMac_GetPhysAddr

Table 87	Specification for `Eth_17_GEthMac_GetPhysAddr` API	
Syntax	`void Eth_17_GEthMac_GetPhysAddr ( const uint8 CtrlIdx, uint8 * const PhysAddrPtr )`	
Service ID	0x08	
Sync/Async	Synchronous	
ASIL Level	QM	
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)	-	-
Return	void	-
Description	Obtains the physical source address used by the indexed controller	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_PARAM_POINTER	
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.*	

Table 87 Specification for `Eth_17_GEthMac_GetPhysAddr` API (continued)

Autosar Version	Applicable for Autosar version 4.2.2.

1.3.3.5 Eth_17_GEthMac_SetPhysAddr

Table 88 Specification for `Eth_17_GEthMac_SetPhysAddr` API

Syntax	`void Eth_17_GEthMac_SetPhysAddr ( const uint8 CtrlIdx, const uint8 * const PhysAddrPtr )`	
Service ID	0x13	
Sync/Async	Synchronous	
ASIL Level	QM	
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_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_17_GETHMAC_E_PARAM_POINTER	
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 4.2.2.	

1.3.3.6 Eth_17_GEthMac_UpdatePhysAddrFilter

Table 89 Specification for `Eth_17_GEthMac_UpdatePhysAddrFilter` API

Syntax	`Std_ReturnType Eth_17_GEthMac_UpdatePhysAddrFilter ( const uint8 CtrlIdx, const uint8 * const PhysAddrPtr, const Eth_FilterActionType Action )`	
Service ID	0x12	
Sync/Async	Synchronous	
ASIL Level	QM	
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. The Eth_17_GEthMac_UpdatePhysAddrFilter() function is available only when EthUpdatePhysAddrFilter is enabled.	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_PARAM_POINTER, ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_17_GETHMAC_E_INV_PARAM	
Configuration dependencies	EthUpdatePhysAddrFilter	
User hints	-	
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)	

Table 89	Specification for `Eth_17_GEthMac_UpdatePhysAddrFilter` API (continued)
	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 4.2.2.

1.3.3.7 Eth_17_GEthMac_WriteMii

Table 90 Specification for `Eth_17_GEthMac_WriteMii` API

Syntax	`Eth_ReturnType  Eth_17_GEthMac_WriteMii`
	`(`
	`  const uint8 CtrlIdx,`
	`  const uint8 TrcvIdx,`
	`  const uint8 RegIdx,`
	`  const uint16 RegVal`
	`)`	

Service ID	0x05	
Sync/Async	Synchronous	
ASIL Level	QM	
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	Eth_ReturnType	ETH_OK: Service accepted ETH_E_NOT_OK: Service denied ETH_E_NO_ACCESS: ETH transceiver access failure
Description	Configures or writes a transceiver register with the requested value *Note: The Eth_17_GEthMac_WriteMii() function is available only when EthCtrlEnableMii is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_17_GETHMAC_E_NOT_INITIALIZED	
Configuration dependencies	EthCtrlEnableMii	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_MAC_MDIO_ADDRESS(rw), GETH_MAC_MDIO_DATA(w)	

Table 90	Specification for `Eth_17_GEthMac_WriteMii` API (continued)
	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 4.2.2.

1.3.3.8 Eth_17_GEthMac_ReadMii

Table 91 Specification for `Eth_17_GEthMac_ReadMii` API

Syntax	`Eth_ReturnType  Eth_17_GEthMac_ReadMii ( const uint8 CtrlIdx, const uint8 TrcvIdx, const uint8 RegIdx, uint16 * const RegValPtr )`	
Service ID	0x06	
Sync/Async	Synchronous	
ASIL Level	QM	
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	Eth_ReturnType	ETH_OK: service accepted ETH_E_NOT_OK: service denied ETH_E_NO_ACCESS: ETH transceiver access failure
Description	Reads a transceiver register. *Note: The Eth_17_GEthMac_ReadMii() function is available only when EthCtrlEnableMii is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_PARAM_POINTER	
Configuration dependencies	EthCtrlEnableMii	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_MAC_MDIO_ADDRESS(rw), GETH_MAC_MDIO_DATA(r)	

Table 91 Specification for `Eth_17_GEthMac_ReadMii` API (continued)

	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 4.2.2.

1.3.3.9 Eth_17_GEthMac_GetDropCount

Table 92 Specification for `Eth_17_GEthMac_GetDropCount` API

Syntax	`Std_ReturnType Eth_17_GEthMac_GetDropCount ( const uint8 CtrlIdx, const uint8 CountValues, uint32 * const DropCount )`	
Service ID	0x14	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx CountValues	Index of the controller within the context of the ETH driver Maximal number of values that can be written from DropCount. Note: As per the AUTOSAR specification 4.2.2, the CountValues parameter is IN-OUT. But the parameter is not a pointer. This is recognized as an error and completely reworked for AUTOSAR 4.3 based on Bugzilla 68804. To keep the compatibility of the interface, the CountValues parameter will stay as variable and will not be changed to a pointer. Hence, this parameter will be used as only IN and will not be considered for OUT.
Parameters (out)	DropCount	A pointer to an array where the drop count values of different errors are written
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: success E_NOT_OK: drop counter could not be obtained
Description	Reads a list with drop counter values of the corresponding controller. In the TC3xx devices, the list DropCount[] contains the following values in the given order, where the maximal possible value denotes an invalid value. For example if this counter is not available: - Dropped packets due to buffer overrun - Dropped packets due to CRC errors - Number of undersize packets which were less than 64 octets long (excluding framing bits, but including FCS octets) and were otherwise will formed. (see IETF RFC 1757)	

Table 92 Specification for `Eth_17_GEthMac_GetDropCount` API (continued)

	- Number of oversize packets which are longer than 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed. (see IETF RFC 1757) - Number of alignment errors, that is, packets which are received and are not an integral number of octets in length and do not pass the CRC. - SQE test error according to IETF RFC1643 dot3StatsSQETestErrors - The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space. (see IETF RFC 2233 ifInDiscards) - Total number of erroneous in-bound packets - The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. (see IETF RFC 2233 ifOutDiscards) - total number of erroneous outbound packets - Single collision frames: A count of successfully transmitted frames on a particular interface for which transmission is inhibited by exactly one collision. (see IETF RFC1643 dot3StatsSingleCollisionFrames) - Multiple collision frames: A count of successfully transmitted frames on a particular interface for which transmission is inhibited by more than one collision. (see IETF RFC1643 dot3StatsMultipleCollisionFrames) - Number of deferred transmission: A count of frames for which the first transmission attempt on a particular interface is delayed because the medium is busy. (see IETF RFC1643 dot3StatsDeferredTransmissions) - Number of late collisions: The number of times that a collision is detected on a particular interface later than 512 bit-times into the transmission of a packet. (see IETF RFC1643 dot3StatsLateCollisions) *Note 1: From above list, item numbers 6 (SQE test error) is not supported in ETH driver. Corresponding this error count, a value 0xFFFFFFFF (ETH_COUNTER_NOT_AVAILABLE) will be filled in DropCount array .Collision related count will be available only if mode of operation is half duplex.* *Note 2: The EthGetDropCountApi() function is available only when EthGetDropCountApi is enabled.*
Source	AUTOSAR
Error handling	ETH_17_GETHMAC_E_PARAM_POINTER, ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_17_GETHMAC_E_NOT_INITIALIZED
Configuration dependencies	EthGetDropCountApi
User hints	-
SFR accessed	CPU_CORE_ID(r), GETH_MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT(r), GETH_RX_ALIGNMENT_ERROR_PACKETS(r), GETH_RX_CRC_ERROR_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),

Table 92 Specification for `Eth_17_GEthMac_GetDropCount` API (continued)

	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 4.2.2.

1.3.3.10 Eth_17_GEthMac_GetEtherStats

Table 93 Specification for `Eth_17_GEthMac_GetEtherStats` API

Syntax	`Std_ReturnType  Eth_17_GEthMac_GetEtherStats ( const uint8 CtrlIdx, uint32 * const etherStats )`	
Service ID	0x15	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the ETH driver
Parameters (out)	etherStats	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: drop counter could not be obtained
Description	Returns the following, 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	

Table 93 Specification for `Eth_17_GEthMac_GetEtherStats` API (continued)

	- etherStatsFragments - etherStatsJabbers - etherStatsCollisions - etherStatsPkts64Octets - etherStatsPkts65to127Octets - etherStatsPkts128to255Octets - etherStatsPkts256to511Octets - etherStatsPkts512to1023Octets - etherStatsPkts1024to1518Octets *Note 1: In the above list, items which are not available are filled with value 0xFFFFFFFF (ETH_COUNTER_NOT_AVAILABLE). Collision-related status is available only if the mode of operation is half duplex.* *Note 2: The Eth_17_GEthMac_GetEtherStats() function is available only when EthGetEtherStatsApi is enabled.*
Source	AUTOSAR
Error handling	ETH_17_GETHMAC_E_PARAM_POINTER, ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_17_GETHMAC_E_NOT_INITIALIZED
Configuration dependencies	EthGetEtherStatsApi
User hints	-
SFR accessed	CPU_CORE_ID(r), GETH_MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT(r), GETH_RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD(ex_r), GETH_RX_128TO255OCTETS_PACKETS_GOOD_BAD(ex_r), GETH_RX_256TO511OCTETS_PACKETS_GOOD_BAD(ex_r), GETH_RX_512TO1023OCTETS_PACKETS_GOOD_BAD(ex_r), GETH_RX_64OCTETS_PACKETS_GOOD_BAD(ex_r), GETH_RX_65TO127OCTETS_PACKETS_GOOD_BAD(ex_r), GETH_RX_ALIGNMENT_ERROR_PACKETS(r), GETH_RX_BROADCAST_PACKETS_GOOD(ex_r), GETH_RX_CRC_ERROR_PACKETS(r), GETH_RX_JABBER_ERROR_PACKETS(r), GETH_RX_MULTICAST_PACKETS_GOOD(ex_r), GETH_RX_OCTET_COUNT_GOOD_BAD(ex_r), GETH_RX_OVERSIZE_PACKETS_GOOD(r), GETH_RX_PACKETS_COUNT_GOOD_BAD(ex_r), GETH_RX_RUNT_ERROR_PACKETS(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 4.2.2.

1.3.3.11 Eth_17_GEthMac_GetCurrentTime

Table 94 Specification for `Eth_17_GEthMac_GetCurrentTime` API

Syntax	`Std_ReturnType Eth_17_GEthMac_GetCurrentTime ( const uint8 CtrlIdx, Eth_TimeStampQualType * const timeQualPtr, Eth_TimeStampType * const timeStampPtr )`	
Service ID	0x16	
Sync/Async	Synchronous	
ASIL Level	QM	
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 TC38xx ETH controller does not provide a 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 a time value from the hardware timer registers. *Note: The Eth_17_GEthMac_GetCurrentTime() function is available only when EthGlobalTimeSupport is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_PARAM_POINTER, ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_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(ex_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 4.2.2.	

1.3.3.12 Eth_17_GEthMac_EnableEgressTimeStamp

Table 95 Specification for `Eth_17_GEthMac_EnableEgressTimeStamp` API

Syntax	`void Eth_17_GEthMac_EnableEgressTimeStamp ( const uint8 CtrlIdx, const uint8 BufIdx )`	
Service ID	0x17	
Sync/Async	Synchronous	
ASIL Level	QM	
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_GEthMac_EnableEgressTimeStamp() function is available only when EthGlobalTimeSupport is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_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 4.2.2.	

1.3.3.13 Eth_17_GEthMac_GetEgressTimeStamp

Table 96 Specification for `Eth_17_GEthMac_GetEgressTimeStamp` API

Syntax	`void Eth_17_GEthMac_GetEgressTimeStamp ( const uint8 CtrlIdx,`

Table 96 Specification for `Eth_17_GEthMac_GetEgressTimeStamp` API (continued)

	`const uint8 BufIdx,`
	`Eth_TimeStampQualType * const timeQualPtr,`
	`Eth_TimeStampType * const timeStampPtr`
	`)`	

Service ID	0x18	
Sync/Async	Synchronous	
ASIL Level	QM	
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)	timeQualPtr timeStampPtr	Quality of hardware time stamp, for example based on current drift Current time stamp
Parameters (in - out)	-	-
Return	void	-
Description	Reads back the egress time stamp on a dedicated message object. It must be called within the TxConfirmation() function. *Note: The Eth_17_GEthMac_GetEgressTimeStamp() function is available only when EthGlobalTimeSupport is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_PARAM_POINTER, ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_17_GETHMAC_E_INV_PARAM	
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 4.2.2.	

1.3.3.14 Eth_17_GEthMac_GetIngressTimeStamp

Table 97 Specification for `Eth_17_GEthMac_GetIngressTimeStamp` API

Syntax	`void Eth_17_GEthMac_GetIngressTimeStamp`
	`(`
	`const uint8 CtrlIdx,`

Table 97 Specification for `Eth_17_GEthMac_GetIngressTimeStamp` API (continued)

	`const Eth_DataType * const DataPtr,`
	`Eth_TimeStampQualType * const timeQualPtr,`
	`Eth_TimeStampType * const timeStampPtr`
	`)`	

Service ID	0x19	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
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	void	-
Description	Reads back the ingress time stamp on a dedicated message object. It must be called within the RxIndication() function. *Note: The Eth_17_GEthMac_GetIngressTimeStamp() function is available only when EthGlobalTimeSupport is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_PARAM_POINTER, ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_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 4.2.2.	

1.3.3.15 Eth_17_GEthMac_SetCorrectionTime

Table 98 Specification for `Eth_17_GEthMac_SetCorrectionTime` API

Syntax	`void Eth_17_GEthMac_SetCorrectionTime ( const uint8 CtrlIdx, const Eth_TimeIntDiffType * const timeOffsetPtr, const Eth_RateRatioType * const rateRatioPtr )`	
Service ID	0x1a	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx timeOffsetPtr rateRatioPtr	Index of the controller within the context of the ETH driver Offset between time stamp grandmaster and time stamp by local clock: (OriginTimeStampSync – IngressTimeStampSync) + Pdelay Time elements to calculate and to modify the ratio of the frequency of the grandmaster in relation to the frequency of the local clock with: ratio = OriginTimeStampDelta / IngressTimeStampDelta
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	None.
Description	Allows the time slave to adjust the local ETH reference clock in the hardware. This function updates or offsets the ETH driver timer as per timeOffsetPtr and corrects the time difference by correcting the timer clock as per rateRatioPtr. *Note: The Eth_17_GEthMac_SetCorrectionTime() function is available only when EthGlobalTimeSupport is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_PARAM_POINTER, ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_E_ACCESS	
Configuration dependencies	EthGlobalTimeSupport	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS(w), GETH_MAC_SYSTEM_TIME_NANOSECONDS_UPDATE(w), GETH_MAC_SYSTEM_TIME_SECONDS(r), GETH_MAC_SYSTEM_TIME_SECONDS_UPDATE(w), GETH_MAC_TIMESTAMP_ADDEND(w), GETH_MAC_TIMESTAMP_CONTROL(rw), 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.*	

Table 98	Specification for `Eth_17_GEthMac_SetCorrectionTime` API (continued)
Autosar Version	Applicable for Autosar version 4.2.2.

1.3.3.16 Eth_17_GEthMac_SetGlobalTime

Table 99 Specification for `Eth_17_GEthMac_SetGlobalTime` API

Syntax	`Std_ReturnType  Eth_17_GEthMac_SetGlobalTime`
	`(`
	`  const uint8 CtrlIdx,`
	`  const Eth_TimeStampType * const timeStampPtr`
	`)`	

Service ID	0x1b	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx timeStampPtr	Index of the controller within the context of the ETH driver New time stamp
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Allows the time master to adjust the global ETH reference clock in the hardware. We can use this method to set a global time base on the ETH in general or to synchronize the global ETH time base with another time base, for example, FlexRay. *Note: The Eth_17_GEthMac_SetGlobalTime() function is available only when EthGlobalTimeSupport is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_PARAM_POINTER, ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_E_ACCESS	
Configuration dependencies	EthGlobalTimeSupport	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), 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(rw), GETH_MAC_TIMESTAMP_CONTROL(rw), 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.*	

Table 99 Specification for `Eth_17_GEthMac_SetGlobalTime` API (continued)

Autosar Version	Applicable for Autosar version 4.2.2.

1.3.3.17 Eth_17_GEthMac_ProvideTxBuffer

Table 100 Specification for `Eth_17_GEthMac_ProvideTxBuffer` API

Syntax	`BufReq_ReturnType  Eth_17_GEthMac_ProvideTxBuffer`
	`(`
	`  const uint8 CtrlIdx,`
	`  Eth_BufIdxType * const BufIdxPtr,`
	`  uint8 ** const BufPtr,`
	`  uint16 * const LenBytePtr`
	`)`	

Service ID	0x09	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the ETH controller within the context of the ETH driver
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_NOT_OK: API call aborted due to development errors BUFREQ_E_BUSY: all buffers are used BUFREQ_E_OVFL: requested buffer too large
Description	Provides access to a transmit buffer of the specified controller	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_PARAM_POINTER	
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.*	

Table 100 Specification for `Eth_17_GEthMac_ProvideTxBuffer` API (continued)

Autosar Version	Applicable for Autosar version 4.2.2.

1.3.3.18 Eth_17_GEthMac_Transmit

Table 101 Specification for `Eth_17_GEthMac_Transmit` API

Syntax	`Std_ReturnType Eth_17_GEthMac_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	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
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: transmission failed
Description	Triggers transmission of a previously filled transmit buffer	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_INV_PARAM, ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_PARAM_POINTER, ETH_17_GETHMAC_E_INV_MODE, ETH_17_GETHMAC_E_INV_CTRL_IDX	
Configuration dependencies	-	
User hints	-	

Table 101	Specification for `Eth_17_GEthMac_Transmit` API (continued)
SFR accessed	CPU_CORE_ID(r), GETH_DMA_CH_RX_CONTROL(r), GETH_DMA_CH_TXDESC_TAIL_POINTER(w), GETH_DMA_CH_TX_CONTROL(r), GETH_MAC_ADDRESS_HIGH0(r), GETH_MAC_ADDRESS_LOW0(r), GETH_MAC_CONFIGURATION(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 4.2.2.

1.3.3.19 Eth_17_GEthMac_Receive

Table 102 Specification for `Eth_17_GEthMac_Receive` API

Syntax	`void Eth_17_GEthMac_Receive ( const uint8 CtrlIdx, Eth_RxStatusType * const RxStatusPtr )`	
Service ID	0xB	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the ETH driver
Parameters (out)	RxStatusPtr	Indicates whether a frame has been received and if so, whether more frames are available or frames got lost.
Parameters (in - out)	-	-
Return	void	-
Description	Triggers frame reception. The Eth_17_GEthMac_Receive function reads the next frame from the receive buffers. 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_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_INV_MODE, ETH_17_GETHMAC_E_INV_CTRL_IDX	
Configuration dependencies	-	
User hints	-	

Table 102	Specification for `Eth_17_GEthMac_Receive` API (continued)
SFR accessed	CPU_CORE_ID(r), GETH_DMA_CH_RXDESC_TAIL_POINTER(w), GETH_DMA_CH_RX_CONTROL(r), GETH_DMA_CH_TX_CONTROL(r), GETH_MAC_CONFIGURATION(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 4.2.2.

1.3.3.20 Eth_17_GEthMac_TxConfirmation

Table 103	Specification for `Eth_17_GEthMac_TxConfirmation` API	
Syntax	`void Eth_17_GEthMac_TxConfirmation ( const uint8 CtrlIdx )`	
Service ID	0xC	
Sync/Async	Synchronous	
ASIL Level	QM	
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_GETHMAC_E_INV_MODE, ETH_17_GETHMAC_E_INV_CTRL_IDX, ETH_17_GETHMAC_E_NOT_INITIALIZED	
Configuration dependencies	-	
User hints	-	
SFR accessed	CPU_CORE_ID(r), GETH_DMA_CH_RX_CONTROL(r), GETH_DMA_CH_TX_CONTROL(r), GETH_MAC_CONFIGURATION(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 103	Specification for `Eth_17_GEthMac_TxConfirmation` API (continued)
Autosar Version	Applicable for Autosar version 4.2.2.

1.3.3.21 Eth_17_GEthMac_GetVersionInfo

Table 104	Specification for `Eth_17_GEthMac_GetVersionInfo` API	
Syntax	`void Eth_17_GEthMac_GetVersionInfo ( Std_VersionInfoType * const VersionInfoPtr )`	
Service ID	0xD	
Sync/Async	Synchronous	
ASIL Level	QM	
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_GEthMac_GetVersionInfo() function is available only when, EthVersionInfoApi() is enabled.*	
Source	AUTOSAR	
Error handling	ETH_17_GETHMAC_E_PARAM_POINTER	
Configuration dependencies	EthVersionInfoApi	
User hints	None.	
SFR accessed	-	
Autosar Version	Applicable for Autosar version 4.2.2.	

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_GEthMac_MainFunction

Table 105 Specification for `Eth_17_GEthMac_MainFunction` API

Syntax	`void Eth_17_GEthMac_MainFunction ( void )`	
Service ID	0x20	
Sync/Async	Synchronous	
ASIL Level	QM	
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_CRC, ETH_E_OVERSIZEFRAME , ETH_E_UNDERSIZEFRAME, ETH_E_ALIGNMENT, ETH_E_SINGLECOLLISION, ETH_E_LATECOLLISION, ETH_E_MULTIPLECOLLISION	
Configuration dependencies	-	
User hints	None.	
SFR accessed	CPU_CORE_ID(r), GETH_MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT(r), GETH_RX_ALIGNMENT_ERROR_PACKETS(r), GETH_RX_CRC_ERROR_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 4.2.2.	

1.3.6 Interrupt service routines

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

1.3.6.1 Eth_17_GEthMac_RxDmaCh0IrqHdlr

Table 106 Specification for `Eth_17_GEthMac_RxDmaCh0IrqHdlr` API

Syntax	`void Eth_17_GEthMac_RxDmaCh0IrqHdlr ( const uint8 CtrlIdx )`	
Service ID	0x10	
Sync/Async	Synchronous	
ASIL Level	QM	
Re-entrancy	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in)	CtrlIdx	Controller Index
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	None.
Description	IRQ handler for the frame reception interrupt and receive buffer unavailable interrupt from the receive DMA channel-0 for the controller with ID passed as the input parameter. *Note: The ETH driver is not handling any error-related interrupts.*	
Source	IFX	
Error handling	ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_INV_CTRL_IDX	
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 4.2.2.	

1.3.6.2 Eth_17_GEthMac_TxDmaCh0IrqHdlr

Table 107 Specification for `Eth_17_GEthMac_TxDmaCh0IrqHdlr` API

Syntax	`void Eth_17_GEthMac_TxDmaCh0IrqHdlr ( const uint8 CtrlIdx )`
Service ID	0x11
Sync/Async	Synchronous

Table 107 Specification for `Eth_17_GEthMac_TxDmaCh0IrqHdlr` API (continued)

ASIL Level	QM	
Re-entrancy	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in)	CtrlIdx	Controller Index
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	None.
Description	IRQ handler for the frame transmission interrupt from transmits DMA channel-0 for the controller with ID passed as the input parameter. *Note: The ETH driver does not handle any error-related interrupts.*	
Source	IFX	
Error handling	ETH_17_GETHMAC_E_NOT_INITIALIZED, ETH_17_GETHMAC_E_INV_CTRL_IDX	
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 4.2.2.	

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_GETHMAC_E_CORE_NOT_CONFIGURED: ETH controller not configured to the core.	IFX	0x64	DET	NA	NA
ETH_E_ACCESS: ETH controller access failure	AUTOSAR	Value Assigned by DEM	DEM	NA	NA

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
ETH_E_ALIGNMENT: Frame alignment error	AUTOSAR	Value Assigned by DEM	DEM	NA	NA
ETH_E_CRC: CRC failure	AUTOSAR	Value Assigned by DEM	DEM	NA	NA
ETH_E_LATECOLLISION: Late frame collision	AUTOSAR	Value Assigned by DEM	DEM	NA	NA
ETH_E_MULTIPLECOLLISION: Multiple frame collision	AUTOSAR	Value Assigned by DEM	DEM	NA	NA
ETH_E_OVERSIZEFRAME : Frame size overflow	AUTOSAR	Value Assigned by DEM	DEM	NA	NA
ETH_E_RX_FRAMES_LOST: ETH frames lost	AUTOSAR	Value Assigned by DEM	DEM	NA	NA
ETH_E_SINGLECOLLISION: Single fame collision	AUTOSAR	Value Assigned by DEM	DEM	NA	NA
ETH_E_UNDERSIZEFRAME: Frame size underflow	AUTOSAR	Value Assigned by DEM	DEM	NA	NA
ETH_17_GETHMAC_E_INV_CTRL_IDX: Invalid controller index. Note: This DET is reported if the controller index is invalid/if the controller is not allocated to the current core.	AUTOSAR	0x01	DET	NA	NA
ETH_17_GETHMAC_E_NOT_INITIALIZED: ETH driver and controller is not initialized. Note: This DET is reported if Eth_17_GEthMac_Init() API is not called before invoking runtime APIs and if controller is not configured to the current core.	AUTOSAR	0x02	DET	NA	NA
ETH_17_GETHMAC_E_PARAM_POINTER: Invalid pointer in parameter list.	AUTOSAR	0x03	DET	NA	NA
ETH_17_GETHMAC_E_INV_PARAM: Invalid parameter.	AUTOSAR	0x04	DET	NA	NA

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
ETH_17_GETHMAC_E_INV_MODE: Invalid controller mode.	AUTOSAR	0x05	DET	NA	NA
ETH_17_GETHMAC_E_INIT_FAILED: Invalid configuration set selection.	AUTOSAR	0x20	DET	NA	NA

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 108 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].
AUTOSAR requirement - SWS_Eth_00226	In AUTOSAR specification, as per the syntax of the `Eth_17_GEthMac_GetDropCount()` API, the datatype of `CountValues` parameter shall be `uint8`. However, this parameter is mentioned as `InOut` in the AUTOSAR specification. Since the parameter is not a pointer type, it cannot be used as `Out` parameter. Hence this parameter is used only as `In` parameter in the Ethernet driver.
AUTOSAR requirement - SWS_EthIf_00085	The Ethernet driver calls `EthIf_RxIndication()` to indicate a successful reception. `LenByte` provided by Ethernet driver to `EthIf_RxIndication()` will be the length of payload. But as per [SWS_EthIf_00085], this should be length of the received frame bytes which includes frame header length of 14 bytes (including target MAC address, source MAC address,

Table 108 Known deviations (continued)

Reference	Deviation
	and type) and payload length, but this is not meaningful since the `DataPtr` points to the payload. AUTOSAR Bugzilla is created to change the description of `LenByte`. Refer https://www.autosar.org/bugzilla/show_bug.cgi?id=76835. (This change is incorporated in AUTOSAR 4.4.0 specification.)
AUTOSAR header file inclusion requirement for ETH module	As per the header file structure in AUTOSAR specification, `Dem.h` shall be included in the `Eth_17_GEthMac.c` file. However, the Ethernet module configuration structure defined in `Eth_17_GEthMac.h` file refers the data type `Dem_EventIdType` from DEM module. Hence, `Dem.h` is included in `Eth_17_GEthMac.h` file.
AUTOSAR requirements - SWS_Eth_00058, SWS_Eth_00064	As per AUTOSAR specification, the return type of `Eth_17_GEthMac_WriteMii()` and `Eth_17_GEthMac_ReadMii()` APIs is mentioned as `Std_ReturnType`. However, the Autosar specification includes a third return value `ETH_E_NO_ACCESS` which is not part of `Std_ReturnType`. Hence, the return values mentioned in the specification are implemented using `Eth_ReturnType` type for these APIs in the Ethernet driver.

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 109 Violations reported by VSMD checker tool for EB03

Rule ID:	EB03
VSMD Node(s):	/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

Table 109 Violations reported by VSMD checker tool for EB03 (continued)

	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_MULTIPLECOLLISION /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthDemEventParameterRefs/ETH_E_OVERSIZEFRAM /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 110 Violations reported by VSMD checker tool for EcucSws_1014

Rule ID:	EcucSws_1014
VSMD Node(s):	/AURIX2G/EcucDefs/Eth /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig /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 111 Violations reported by VSMD checker tool for EcucSws_1035

Rule ID:	EcucSws_1035
VSMD Node(s):	/AURIX2G/EcucDefs/Eth /AURIX2G/EcucDefs/Eth/EthConfigSet /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlEnableMii /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlEnableRxInterrupt /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlEnableTxInterrupt /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlIdx

Table 111 Violations reported by VSMD checker tool for EcucSws_1035 (continued)

	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlPhyAddress
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlRxBufLenByte
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlTxBufLenByte
	/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
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthRxBufTotal
	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthTxBufTotal
	/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

Table 111 Violations reported by VSMD checker tool for EcucSws_1035 (continued)

	/AURIX2G/EcucDefs/Eth/EthGeneral/EthCtrlOffloading/EthCtrlEnableOffloadChecksumUDP /AURIX2G/EcucDefs/Eth/EthGeneral/EthDevErrorDetect /AURIX2G/EcucDefs/Eth/EthGeneral/EthGetDropCountApi /AURIX2G/EcucDefs/Eth/EthGeneral/EthGetEtherStatsApi /AURIX2G/EcucDefs/Eth/EthGeneral/EthGlobalTimeSupport /AURIX2G/EcucDefs/Eth/EthGeneral/EthIndex /AURIX2G/EcucDefs/Eth/EthGeneral/EthMainFunctionPeriod /AURIX2G/EcucDefs/Eth/EthGeneral/EthMaxCtrlsSupported /AURIX2G/EcucDefs/Eth/EthGeneral/EthUpdatePhysAddrFilter /AURIX2G/EcucDefs/Eth/EthGeneral/EthVersionInfoApi
Description:	For Containers, Parameters and References elements UUID must be unique (also between StMD and VSMD).
Additional Information:	-

Table 112 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 113 Violations reported by VSMD checker tool for TpsEcuc_06051_ASR41

Rule ID:	TpsEcuc_06051_ASR41
VSMD Node(s):	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlRxBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlTxBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthRxBufTotal /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthTxBufTotal
Description:	The implementationConfigClass of an EcucParameterDef or EcucAbstractReferenceDef in

Table 113 Violations reported by VSMD checker tool for TpsEcuc_06051_ASR41 (continued)

	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 for Rx and Tx. 2. These parameters are used for generating pre-compile macro for total size allocated to Rx and Tx buffers. These generated pre-compile macros are further used in code for memory allocation and hence the implementationConfigClass is set to PreCompile.

Table 114 Violations reported by VSMD checker tool for TpsEcuc_08032

Rule ID:	TpsEcuc_08032
VSMD Node(s):	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlRxBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlTxBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthRxBufTotal /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthTxBufTotal
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.
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 for Rx and Tx. 2. These parameters are used for generating pre-compile macro for total size allocated to Rx and Tx buffers.

Table 114 Violations reported by VSMD checker tool for TpsEcuc_08032 (continued)

	These generated pre-compile macros are further used in code for memory allocation and hence the postBuildVariantValue is set to FALSE.

Table 115 Violations reported by VSMD checker tool for TpsEcuc_08033

Rule ID:	TpsEcuc_08033
VSMD Node(s):	/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 physical address in the EthCtrlPhyAddress parameter. Hence, the container multiplicity is fixed to 1 and hence the post build multiplicity (0..1) cannot be supported.

Table 116 Violations reported by VSMD checker tool for TpsEcuc_08038

Rule ID:	TpsEcuc_08038
VSMD Node(s):	/AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlRxBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthCtrlTxBufLenByte /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthRxBufTotal /AURIX2G/EcucDefs/Eth/EthConfigSet/EthCtrlConfig/EthTxBufTotal
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 StMDwith 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 for Rx and Tx.

Table 116 Violations reported by VSMD checker tool for TpsEcuc_08038 (continued)

	2. These parameters are used for generating pre-compile macro for total size allocated to Rx and Tx buffers. These generated pre-compile macros are further used in code for memory allocation and hence should be pre-compile time.

1.3.9.2 Limitations

The section describes the limitations of the ETH driver.

Table 117 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 on 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.
`Eth_17_GEthMac_Transmit()` API does not work if `BufIdx` parameter is not passed in same sequence as it is provided	The `Eth_17_GEthMac_Transmit()` API is designed to work in such a way that `BufIdx` parameter passed to this API shall be in same sequence as `BufIdx` is allocated by calling `Eth_17_GEthMac_ProvideTxBuffer()` API . An example for the right usage of this API is given below Step1- Invoke `Eth_17_GEthMac_ProvideTxBuffer()` API - Assume that BufIdx1 is allocated to application from this API Step2- Invoke `Eth_17_GEthMac_ProvideTxBuffer()` API - Assume that BufIdx2 is allocated to application from this API Step3- Invoke `Eth_17_GEthMac_Transmit()` API with parameter as BufIdx1 Step4- Invoke `Eth_17_GEthMac_Transmit()` API with parameter as BufIdx2 In this case the limitation is, if Step4 is followed before Step3, then the `Eth_17_GEthMac_Transmit()` API does not work. The `Eth_17_GEthMac_Transmit()` API is designed in this way to make use of a feature from Ethernet controller hardware that it automatically manages linked list circular buffer. Hence Ethernet driver performance is increased by avoiding this feature in software.
Usage of Compiler library in Ethernet driver for compilation of global time APIs	If the pre-compile configuration parameter `EthGlobalTimeSupport` is enabled in Ethernet driver configuration, then the Ethernet driver will use the compiler's

Table 117 Known limitations (continued)

Reference	Limitation
	floating point library for implementation of double precision floating point in global time APIs. Infineon has not validated the compiler's floating library used by the Ethernet driver.

Revision history

Table 118 Revision History

Date	Version	Description
2020-11-26	1.0	Released
2020-11-25	0.1	- Initial Version - ETH driver chapter moved from MC-ISAR_TC3xx_UM_COM-E to this document - Configuration parameter EthMDCClockFrequency is added - Default value of EthMaxCtrlsSupported is changed

Trademarks

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

Edition 2020-11-26

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.