MICROSAR Ethernet Transceiver Driver

Technical Reference

vVirtualTarget

Version 2.00.00

Author	David Feßler
Status	Released

1 Document Information

1.1 History

Author	Date	Version	Remarks
Alex Lunkenheimer	2008-10-02	1.0	Creation of document
David Feßler	2015-01-16	2.00.00	Renamed to VTT

Table 1-1 History of the document

1.2 Reference Documents

No.	Title	Version
[1]	AUTOSAR_SWS_EthernetTransceiver.pdf	1.3.0
[2]	AUTOSAR_SWS_DET.pdf	3.3.0
[3]	AUTOSAR_SWS_DEM.pdf	5.0.0
[4]	AUTOSAR_SWS_BSWGeneral.pdf	1.0.0

Table 1-2 Reference documents

1.3 Scope of the Document

This technical reference describes the general use of the Ethernet Transceiver Driver basis software. Please refer to your Release Notes to get a detailed description of the platform (host, compiler, transceiver) your Vector Ethernet Bundle has been configured for.

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

Contents

Illustrations

Tables

Component History

Component Version	New Features
01.00.xx	created

Table 1-3 Component history

2 Introduction

This document describes the functionality, API and configuration of the Ethernet Transceiver Driver.

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile, link-time, post-build	
Vendor ID:	ETHTRCV_30_CANOEEMU_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	ETHTRCV_30_CANOEEMU_MODULE_ID	73 decimal

* For the precise AUTOSAR Release 3.x please see the release specific documentation.

The Ethernet Transceiver Driver provides hardware independent access to control connected transceivers in a generic way. It offers the functionality to control the mode of operation of connected transceivers as well as to determine their current state, e.g. if events like link status change or bus errors happened.

The transceiver itself is a hardware device, which mainly transforms the logical I/0 signals of the Ethernet Controller to the bus compliant electrical levels, currents and timings.

2.1 Architecture Overview

The following figure shows where the Ethernet Transceiver Driver is located in the AUTOSAR architecture.

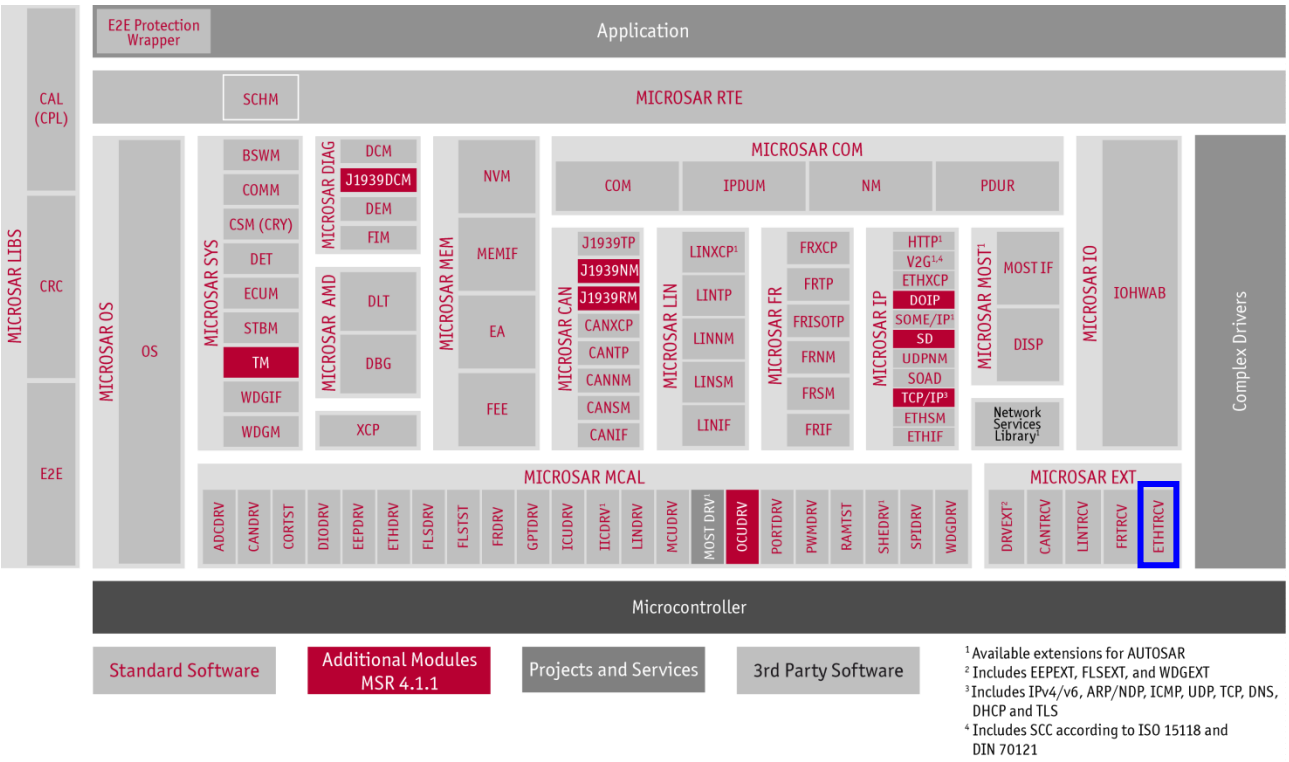

Figure 2-1 AUTOSAR architecture

The next figure shows the interfaces to adjacent modules of the Ethernet Transceiver Driver. These interfaces are described in chapter 5.

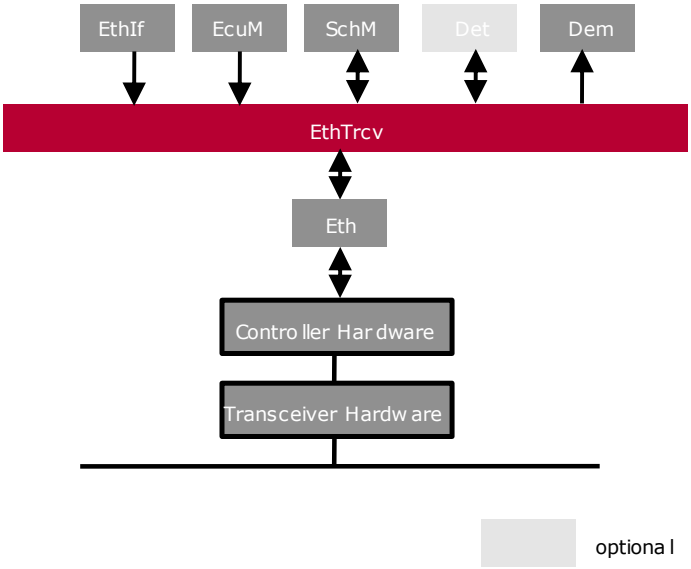

Figure 2-2 Interfaces to adjacent modules of the Ethernet Transceiver Driver

	Info The Transceiver Hardware is not directly accessible but via the Controller Hardware. Thus the Ethernet Transceiver Driver does use the Ethernet Driver to access the transceiver hardware.

Applications do not access the services of the BSW modules directly. They use the service ports provided by the BSW modules via the RTE.

3 Functional Description

3.1 Initialization

3.1.1 High-Level Initialization

The Ethernet Transceiver Driver is initialized by calling the `EthTrcv_30_Canoeemu_Init` service with the configuration as parameter.

The transceiver itself is initialized by calling the `EthTrcv_30_Canoeemu_TransceiverInit` service with the corresponding index for each transceiver.

3.2 States

The transceiver should be set to a defined state during initialization by upper layer software component (Ethernet Interface). Otherwise the initial state is undefined.

3.3 Error Handling

3.3.1 Development Error Reporting

Development errors are reported to DET using the service Det_ReportError (specified in [2]), if this feature is enabled in GENy.

The reported Ethernet Transceiver Driver ID is 73.

The reported service IDs identify the services which are described in 5.2. The following table presents the service IDs and the related services:

Service ID	Service
0x01	`ETHTRCV_30_CANOEEMU_API_ID_INIT`
0x02	`ETHTRCV_30_CANOEEMU_API_ID_TRANSCEIVER_INIT`
0x03	`ETHTRCV_30_CANOEEMU_API_ID_SET_TRANSCEIVER_MODE`
0x04	`ETHTRCV_30_CANOEEMU_API_ID_GET_TRANSCEIVER_MODE`
0x05	`ETHTRCV_30_CANOEEMU_API_ID_START_AUTO_NEG`
0x06	`ETHTRCV_30_CANOEEMU_API_ID_GET_LINK_STATE`
0x07	`ETHTRCV_30_CANOEEMU_API_ID_GET_BAUD_RATE`
0x08	`ETHTRCV_30_CANOEEMU_API_ID_GET_DUPLEX_MODE`
0x09	`ETHTRCV_30_CANOEEMU_API_ID_GET_VERSION_INFO`

Table 3-1 Mapping of service IDs to services

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

Error Code		Description
0x01	`ETHTRCV_30_CANOEEMU_E_INV_TRCV_IDX`	The Ethernet Transceiver Driver was called with an invalid Transceiver Index
0x02	`ETHTRCV_30_CANOEEMU_E_NOT_INITIALIZED`	An Ethernet Transceiver Driver service was called without initializing the module first by calling EthTrcv_Init

Error Code		Description
0x03	ETHTRCV_30_CANOEEMU_E_INV_POINTER	An Ethernet Transceiver Driver service was called with a zero pointer as parameter
0x04	ETHTRCV_30_CANOEEMU_E_INV_PARAM	An Ethernet Transceiver Driver service was called with an invalid parameter
0x05	ETHTRCV_30_CANOEEMU_E_INV_CONFIG	The Ethernet Transceiver Driver configuration is invalid

Table 3-2 Errors reported to DET

4 Integration

This chapter gives necessary information for the integration of the Ethernet Transceiver Driver into an application environment of an ECU.

4.1 Scope of Delivery

Depending on the delivery type of the Ethernet Transceiver Driver the static files described in chapter 4.1.1 or 4.1.2 are delivered. In both case the files described in 4.1.3 are delivered.

4.1.1 Static Files (Source Code Delivery)

The static files are not to be modified.

File Name	Description
`EthTrcv_30_Canoeemu.c`	Implementation
`EthTrcv_30_Canoeemu.h`	API declaration
`EthTrcv_30_Canoeemu_Types.h`	Data types declaration
`EthTrcv_30_Canoeemu_Priv.h`	Component local macro and variable declaration
`EthTrcv_30_Canoeemu_Lcfg.h`	Link-time parameter configuration declaration
`EthTrcv_30_Canoeemu_PBcfg.h`	Post-build time parameter configuration declaration
`EthTrcv_GeneralTypes.h`	General types header (see Common directory)

Table 4-1 Static files (source code delivery)

4.1.2 Static Files (Object Code Delivery)

The static files are not to be modified.

File Name	Description
`libEthTrcv_30_Canoeemu.a`	Implementation
`EthTrcv_30_Canoeemu.h`	API declaration
`EthTrcv_30_Canoeemu_Types.h`	Data types declaration
`EthTrcv_30_Canoeemu_Lcfg.h`	Link-time parameter configuration declaration
`EthTrcv_30_Canoeemu_PBcfg.h`	Post-build time parameter configuration declaration

Table 4-2 Static files (object code delvery)

4.1.3 Dynamic Files

The dynamic files can be modified.

File Name	Description
`EthTrcv_30_Canoeemu_Cfg.h`	Pre-compile time parameter configuration
`EthTrcv_30_Canoeemu_Lcfg.c`	Link-time parameter configuration
`EthTrcv_30_Canoeemu_PBcfg.c`	Post-build parameter configuration

Table 4-3 Dynamic files

4.2 Compiler Abstraction and Memory Mapping

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

The following table contains the memory section names and the compiler abstraction definitions which are defined for the Ethernet Transceiver Driver and illustrates their assignment among each other.

Memory Mapping Sections \ Compiler Abstraction Definitions	ETHTRCV_CONST	ETHTRCV_VAR	ETHTRCV_CODE
`ETHTRCV_30_CANOEEMU_START_SEC_CONST_UNSPECIFIED`	■		
`ETHTRCV_30_CANOEEMU_START_SEC_CONST_32BIT`	■		
`ETHTRCV_30_CANOEEMU_START_SEC_CONST_16BIT`	■		
`ETHTRCV_30_CANOEEMU_START_SEC_CONST_8BIT`	■		
`ETHTRCV_30_CANOEEMU_START_SEC_VAR_NOINIT_UNSPECIFIED`		■	
`ETHTRCV_30_CANOEEMU_START_SEC_VAR_NOINIT_32BIT`		■	
`ETHTRCV_30_CANOEEMU_START_SEC_VAR_NOINIT_16BIT`		■	
`ETHTRCV_30_CANOEEMU_START_SEC_VAR_NOINIT_8BIT`		■	
`ETHTRCV_30_CANOEEMU_START_SEC_CODE`			■

Table 4-4 Compiler abstraction and memory mapping

4.3 Data Consistency

To ensure data consistency and a correct function of the Ethernet Driver the exclusive area `ETHTRCV_30_CANOEEMU_EXCLUSIVE_AREA_0` has to be provided during the integration.

Considering the timing behavior of your system (e.g. depending on the CPU load of your system, priorities and interruptibility of interrupts and OS tasks and their jitter and delay

times) the integrator has to choose and configure a critical section solution in such way that it is ensured that the API functions do not interrupt each other.

It is recommended to use an AUTOSAR OS Resource for `ETHTRCV_30_CANOEEMU_EXCLUSIVE_AREA_0`.

5 API Description

5.1 Interfaces Overview

The Ethernet Transceiver Driver provides the following services:

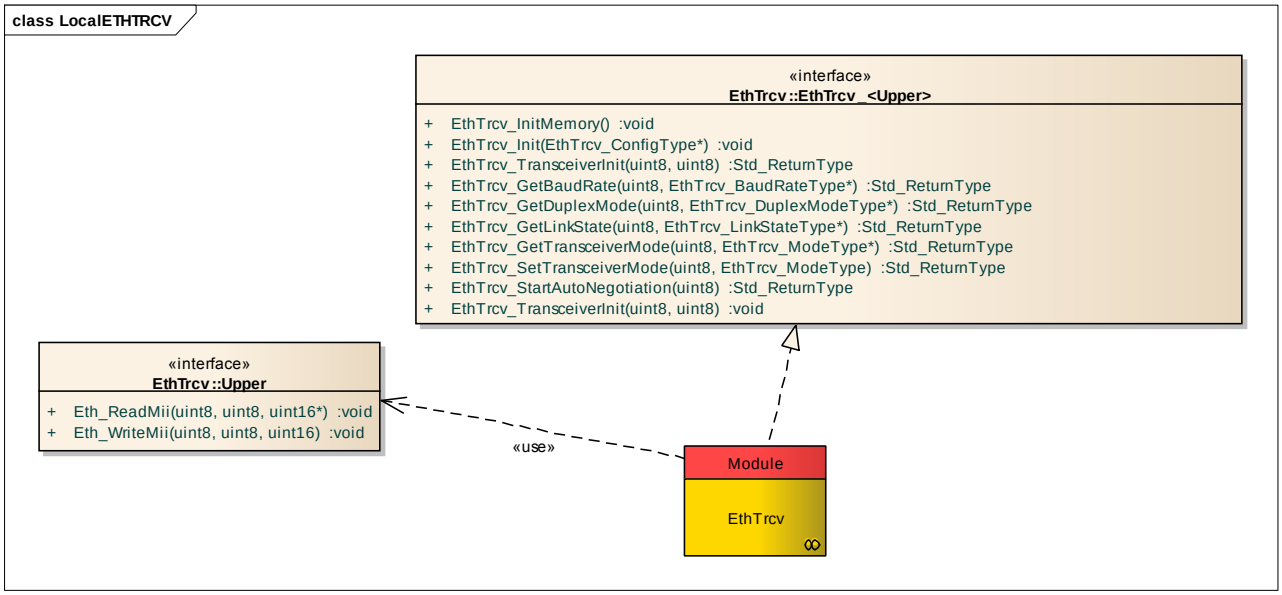

Figure 5-1 Ethernet Transceiver Driver API

5.2 Type Definitions

Type Name	C-Type	Description	Value Range
EthTrcv_ConfigType	void	Transceiver configuration	NULL_PTR Transceiver uses Link-time configuration
			CFG_PTR Post-build configuration
EthTrcv_ModeType	uint8	Defines all possible transceiver modes	ETHTRCV_MODE_DOWN Transceiver inactive
			ETHTRCV_MODE_ACTIVE Normal operation mode
EthTrcv_LinkStateType	uint8	Defines all possible transceiver link states	ETHTRCV_LINK_STATE_DOWN Transceiver link disconnected
			ETHTRCV_LINK_STATE_ACTIVE Transceiver link connected
EthTrcv_BaudRateType	uint8	Defines all possible transceiver baud rates	ETHTRCV_BAUD_RATE_10MBIT 10MBit baud rate
			ETHTRCV_BAUD_RATE_100MBIT 100MBit baud rate
			ETHTRCV_BAUD_RATE_1000MBIT 1000MBit baud rate

Type Name	C-Type	Description	Value Range
EthTrcv_DuplexModeType	uint8	Defines all possible transceiver duplex modes	ETHTRCV_DUPLEX_MODE_HALF Half duplex connection
			ETHTRCV_DUPLEX_MODE_FULL Full duplex connection
EthTrcv_StateType	uint8	Defines all possible transceiver states	ETHTRCV_STATE_UNINIT Ethernet Transceiver Driver not initialized
			ETHTRCV_STATE_INIT Ethernet Transceiver Driver initialized
			ETHTRCV_STATE_ACTIVE Ethernet Transceiver Driver active
			ETHTRCV_STATE_DOWN Ethernet Transceiver Driver down

Table 5-1 Type definitions

5.3 Services provided by Ethernet Transceiver Driver

5.3.1 EthTrcv_30_Canoeemu_InitMemory

Prototype	
void **EthTrcv_30_Canoeemu_InitMemory** (void)	
Parameter	
void	
Return Code	
void	void
Functional Description	
Initializes global variables.	
Particularities and Limitations	
Re-entrant, synchronous	
Caution Has to be called before usage of the module	
Pre-Conditions	
Call Context	
Initialization	

Table 5-2 EthTrcv_30_Canoeemu_InitMemory

5.3.2 EthTrcv_30_Canoeemu_Init

Prototype	
`void EthTrcv_30_Canoeemu_Init (const EthTrcv_30_Canoeemu_ConfigType *CfgPtr)`	
Parameter	
CfgPtr	Pointer to module configuration
Return Code	
void	void
Functional Description	
Stores the start address of the post build time configuration of the module and may be used to initialize the data structures.	
Particularities and Limitations	
Re-entrant, synchronous	
Caution Has to be called before usage of the module	
Pre-Conditions	
Call Context	
Initialization	

Table 5-3 EthTrcv_30_Canoeemu_Init

5.3.3 EthTrcv_30_Canoeemu_TransceiverInit

Prototype	
`Std_ReturnType EthTrcv_30_Canoeemu_TransceiverInit (uint8 TrcvIdx, uint8 CfgIdx)`	
Parameter	
TrcvIdx	Zero based index of the transceiver
CfgIdx	Configuration index
Return Code	
Std_ReturnType	> E_OK : Transceiver configured > E_NOT_OK : Transceiver configuration failed
Functional Description	
Initializes an Ethernet transceiver (register configuration).	
Particularities and Limitations	
- Re-entrant, synchronous - If API optimization is enabled, parameter TrcvIdx is void	

	Caution Has to be called before usage of the module
Pre-Conditions	
Call Context	
Initialization	

Table 5-4 EthTrcv_30_Canoeemu_TransceiverInit

5.3.4 EthTrcv_30_Canoeemu_SetTransceiverMode

Prototype	
`Std_ReturnType` **`EthTrcv_30_Canoeemu_SetTransceiverMode`** `(uint8 TrcvIdx, EthTrcv_ModeType TrcvMode)`	
Parameter	
TrcvIdx	Zero based index of the transceiver
TrcvMode	Transceiver mode
Return Code	
Std_ReturnType	> E_OK : Transceiver mode changed > E_NOT_OK : Transceiver mode change failed
Functional Description	
Set transceiver mode.	
Particularities and Limitations	
- Re-entrant, synchronous - If API optimization is enabled, parameter TrcvIdx is void	
Pre-Conditions	
Init and TransceiverInit must be called before, otherwise a DET is thrown (if enabled)	
Call Context	
Interrupt or task level	

Table 5-5 EthTrcv_30_Canoeemu_SetTransceiverMode

5.3.5 EthTrcv_30_Canoeemu_GetTransceiverMode

Prototype	
`Std_ReturnType` **`EthTrcv_30_Canoeemu_GetTransceiverMode`** `(uint8 TrcvIdx, EthTrcv_ModeType *TrcvModePtr)`	
Parameter	
TrcvIdx	Zero based index of the transceiver
TrcvModePtr	Pointer for transceiver mode

Return Code	
Std_ReturnType	> E_OK : Transceiver mode evaluated > E_NOT_OK : Transceiver mode evaluation failed
Functional Description	
Get transceiver mode.	
Particularities and Limitations	
- Re-entrant, synchronous - If API optimization is enabled, parameter TrcvIdx is void	
Pre-Conditions	
Init and TransceiverInit must be called before, otherwise a DET is thrown (if enabled)	
Call Context	
Interrupt or task level	

Table 5-6 EthTrcv_30_Canoeemu_GetTransceiverMode

5.3.6 EthTrcv_30_Canoeemu_StartAutoNegotiation

Prototype	
`Std_ReturnType EthTrcv_30_Canoeemu_StartAutoNegotiation (uint8 TrcvIdx)`	
Parameter	
TrcvIdx	Zero based index of the transceiver
Return Code	
Std_ReturnType	> E_OK : Auto negotiation started > E_NOT_OK : Auto negotiation start failed
Functional Description	
Start automatic mode negotiation (10/100MBit, Full/Half-Duplex).	
Particularities and Limitations	
- Re-entrant, synchronous - If API optimization is enabled, parameter TrcvIdx is void	
Pre-Conditions	
Init and TransceiverInit must be called before, otherwise a DET is thrown (if enabled)	
Call Context	
Interrupt or task level	

Table 5-7 EthTrcv_30_Canoeemu_StartAutoNegotiation

5.3.7 EthTrcv_30_Canoeemu_GetLinkState

Prototype	
`Std_ReturnType` **`EthTrcv_30_Canoeemu_GetLinkState`** `(uint8 TrcvIdx, EthTrcv_LinkStateType *LinkStatePtr)`	
Parameter	
TrcvIdx	Zero based index of the transceiver
LinkStatePtr	Pointer for link state value
Return Code	
Std_ReturnType	> E_OK : Link state read > E_NOT_OK : Link state read failed
Functional Description	
Get transceiver link state.	
Particularities and Limitations	
- Re-entrant, synchronous - If API optimization is enabled, parameter TrcvIdx is void	
Pre-Conditions	
Init and TransceiverInit must be called before, otherwise a DET is thrown (if enabled)	
Call Context	
Interrupt or task level	

Table 5-8 EthTrcv_30_Canoeemu_GetLinkState

5.3.8 EthTrcv_30_Canoeemu_GetBaudRate

Prototype	
`Std_ReturnType` **`EthTrcv_30_Canoeemu_GetBaudRate`** `(uint8 TrcvIdx, EthTrcv_BaudRateType *BaudRatePtr)`	
Parameter	
TrcvIdx	Zero based index of the transceiver
BaudRatePtr	pointer for baud rate value
Return Code	
Std_ReturnType	> E_OK : Baud rate read > E_NOT_OK : Baud rate read failed
Functional Description	
Get transceiver baud rate.	
Particularities and Limitations	
- Re-entrant, synchronous - If API optimization is enabled, parameter TrcvIdx is void	
Pre-Conditions	

Init and TransceiverInit must be called before, otherwise a DET is thrown (if enabled)
Call Context
Interrupt or task level

Table 5-9 EthTrcv_30_Canoeemu_GetBaudRate

5.3.9 EthTrcv_30_Canoeemu_GetDuplexMode

Prototype	
`Std_ReturnType` **`EthTrcv_30_Canoeemu_GetDuplexMode`** `(uint8 TrcvIdx, EthTrcv_DuplexModeType *DuplexModePtr)`	
Parameter	
TrcvIdx	Zero based index of the transceiver
DuplexModePtr	Pointer for duplex mode value
Return Code	
Std_ReturnType	> E_OK : Duplex mode read > E_NOT_OK : Duplex mode read failed
Functional Description	
Get transceiver duplex mode.	
Particularities and Limitations	
- Re-entrant, synchronous - If API optimization is enabled, parameter TrcvIdx is void	
Pre-Conditions	
Init and TransceiverInit must be called before, otherwise a DET is thrown (if enabled)	
Call Context	
Interrupt or task level	

Table 5-10 EthTrcv_30_Canoeemu_GetDuplexMode

5.3.10 EthTrcv_30_Canoeemu_GetVersionInfo

Prototype
`void` **`EthTrcv_30_Canoeemu_GetVersionInfo`** `(Std_VersionInfoType *VersionInfoPtr)`

Parameter	
VersionInfoPtr	Returns the following version information: - Vendor ID - Module ID - Software major version - Software minor version - Software patch version
Return Code	
void	void
Functional Description	
Get driver version.	
Particularities and Limitations	
Re-entrant, synchronous	
Pre-Conditions	
Call Context	
Interrupt or task level	

Table 5-11 EthTrcv_30_Canoeemu_GetVersionInfo

5.4 Services used by Ethernet Transceiver Driver

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

Component	API
DET (optional)	`Det_ReportError`
Dem	`Dem_ReportErrorStatus`
Eth	`Eth_WriteMii` `Eth_ReadMii`

Table 5-12 Services used by the Ethernet Transceiver Driver

5.5 Callback Functions

The Ethernet Transceiver Driver does not provide callback functions.

6 AUTOSAR Standard Compliance

6.1 Deviations

No deviation.

6.2 Additions/ Extensions

Not relevant.

6.3 Limitations

6.3.1 L002: Multiple Configuration

The current version of the Ethernet Transceiver Driver supports only one configuration.

6.3.2 L003: Transceiver Access

The current version of the Ethernet Transceiver Driver for CANoe is not accessing the physical transceiver but provides the interface of the Ethernet Transceiver Driver only.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
EAD	Embedded Architecture Designer; generation tool for MICROSAR components
GENy	Generation tool for CANbedded and MICROSAR components

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DIO	Digital Input Output
EAD	Embedded Architecture Designer
ECU	Electronic Control Unit
Eth	Ethernet Controller Driver
EthIf	Ethernet Interface
EthTrcv	Ethernet Transceiver Driver
HIS	Hersteller Initiative Software
ICU	Input Capture Unit
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
Platform	Hardware including host and communication controller (might also be integrated in host) on which the communication stack is implemented.
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

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

www.vector-informatik.com