MICROSAR Etm

Technical Reference

Ethernet Testability Module

Version 5.1.0

Authors	Jens Bauer
Status	Released

Document Information

History

Author	Date	Version	Remarks
Jens Bauer	2015-12-04	2.00.xx	Update to Etm protocol (MSR4-R14)
Jens Bauer	2015-12-21	4.00.xx	Rename module to Etm
Jens Bauer	2016-04-14	4.01.xx	Adapt to implementation
Jens Bauer	2017-02-01	4.02.xx	Describe Etm event behavior
Jens Bauer	2017-05-02	5.00.xx	Added ClearNdpCache/Lock
Jens Bauer	2017-05-17	5.01.xx	Code restructuring

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_PRS_TestabilityProtocolAndServicePrimitives.pdf	V1.1
[2]	AUTOSAR	AUTOSAR_SWS_DET.pdf	V4.0.3
[3]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	V4.2.1

Scope of the Document

This technical reference describes the general use of the Ethernet Testability Module (Etm) basis software module for interaction with external testers.

Caution

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

Contents

Illustrations

Tables

1 Component History

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

Component Version	New Features
2.00.xx	Update to Etm protocol (MSR4-R13)
3.00.xx	Update to Etm protocol (MSR4-R14)
4.00.xx	Renaming to Etm
4.01.xx	Improved UserScript configuration
4.02.xx	Internal improvements to prevent stack overflows
5.00.xx	Added ClearNdpCache Service Primitive and persistent lock functionality
5.01.xx	Code restructuring and integration of release findings

Table 1-1 Component history

2 Introduction

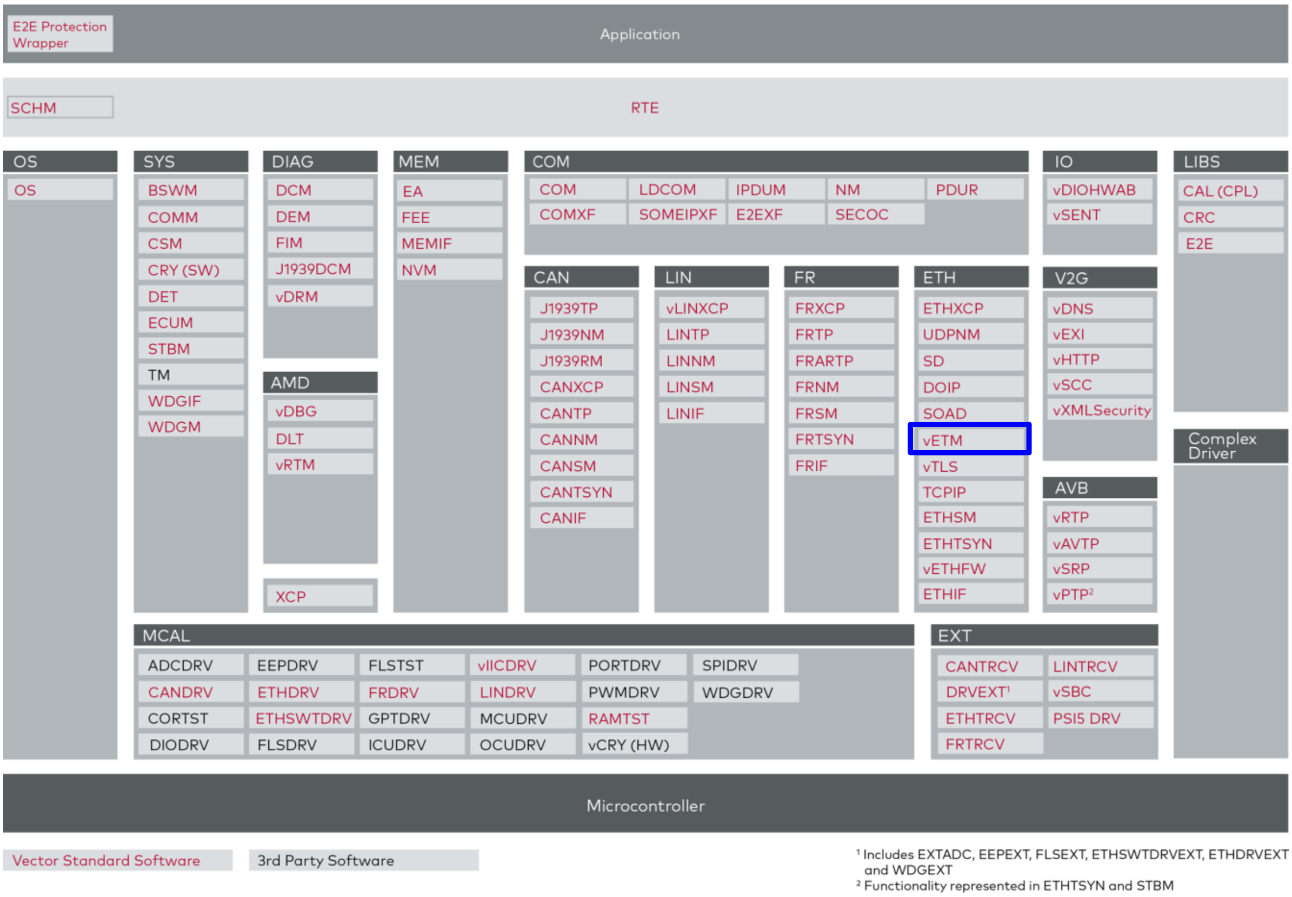
This document describes the functionality, API and configuration of the AUTOSAR BSW module Etm as specified in [1].

Supported AUTOSAR Release:	4.2.1	
Supported Configuration Variants:	pre-complie	
Vendor ID:	ETM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	ETM_MODULE_ID	170 decimal (according to ref. [3]) (TcpIp-Stack Modul ID)

The primary objective of the Etm module is to validate a remote TCP stack's adherence to standards as specified in AUTOSAR standard specifications for TCP. In order to trigger a TCP stack to send/receive TCP packets (which can be validated for compliance). So, to summarize, Etm module is an agent who resides on the TCP stack and can communicate with a remote tester over a command channel. The primary job of the Etm module is to get specific command from the test tool, interpret them and using socket API instruct the local TCP stack under test to take TCP actions to generate corresponding TCP packets.

2.1 Architecture Overview

The following figure shows where the Etm is located in the AUTOSAR architecture.



[1] Includes EXTADC, EEPEXT, FLSEXT, ETHSWTDRVEXT, ETHDRVEXT and WDGEXT

[2] Functionality represented in ETHTSYN and STBM

Figure 2-1 AUTOSAR 4.2 Architecture Overview

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

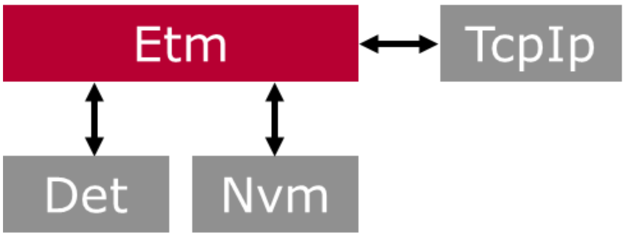


Figure 2-2 Interfaces to adjacent modules of the Etm

Applications do not access the services of the BSW modules directly. They use the service ports provided by the BSW modules via the RTE. The Etm module does not support any service ports.

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for the Etm.

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

- Table 3-1 Supported AUTOSAR standard conform features
- Table 3-2 Not supported AUTOSAR standard conform features

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Supported Service Groups: GENERAL, UDP, TCP

Table 3-1 Supported AUTOSAR standard conform features

3.1.1 Deviations

The following features specified in [1] are not supported:

Not Supported AUTOSAR Standard Conform Features
Service Primitive: SHUTDOWN

Table 3-2 Not supported AUTOSAR standard conform features

3.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
Supported Service Group NDP (0x0A) with Service Primitive ClearNdpCache (0xFF) command
Supported Persistent lock functionality (persistent de-/activation of command processing)

Table 3-3 Features provided beyond the AUTOSAR standard

3.1.2.1 Service Primitive ClearNdpCache

The Service Primitive ClearNdpCache (0xFF) of the Service Group NDP (0x0A) starts the clearing of the IPv6 controller NDP cache that is referenced by the Etm IPv6 address in the configuration. This Service Primitive is called without parameters.

3.1.2.2 Persistent lock functionality

The APIs `Etm_DeactivateProcessing()` and `Etm_ActivateProcessing()` can be called (i.e. by a diagnostic service) to stop and start the Etm processing. If deactivated Etm does not receive further Service Primitives and stop further command processing until Etm gets activated. The processing state will be saved persistent to the NVM and is persistent over an ECU reboot. To use this functionality create the `/MICROSAR/Etm/EtmNvmBlock` container in the configuration. The referenced NVM block descriptor should be configured with the following settings:

- Block Length: 1 byte

  (e.g. /MICROSAR/NvM/NvMBlockDescriptor/NvMNvBlockLength)
- Ram Block Data: Etm_NvmBlock_Ram

  (e.g. /MICROSAR/NvM/NvMBlockDescriptor/NvMRamBlockDataAddress)
- Rom Block Data: Etm_NvmBlock_Rom (default value)

  (e.g. /MICROSAR/NvM/NvMBlockDescriptor/NvMRomBlockDataAddress)
- Select Block For ReadAll: TRUE

  (e.g. /MICROSAR/NvM/NvMBlockDescriptor/NvMSelectBlockForReadAll)
- Select Block For WriteAll: TRUE

  (e.g. /MICROSAR/NvM/NvMBlockDescriptor/NvMSelectBlockForWriteAll)

C-Type	Description	Value Range
uint8	Etm Processing States	`0x00u` ETM_PROCESSING_INVALID
		`0x01u` ETM_PROCESSING_ACTIVE
		`0x02u` ETM_PROCESSING_INACTIVE

Table 3-4 Etm Processing State

Caution
Deactivate processing after usage and set default value (NVM rom block) to inactive to avoid safety and security risks.

3.2 Initialization

The Etm is initialized by calling the `Etm_InitMemory()` service followed by `Etm_Init()`. The current version does not support link-time or post-build configuration. Hence, `Etm_Init()` can be called with a `NULL_PTR`.

3.3 Main Functions

The Etm module has a main function that needs to be called periodically. This is normally done by the RTE. If no RTE is used call the `Etm_MainFunction()` manually. The tasks are connection establishment on command channel and command processing.

3.4 ReceiveAndForward Handling

Referring to the ReceiveAndForward Use Cases specified in the AUTOSAR Etm protocol specification[2] for all send or received data on ECU side an Etm event with the received and forwarded data are expected by the test system. Due to that TCP is a stream protocol and can collect, merge and split the send or received data it should not be expected that for every single data package an Etm event is received – especially if there are a lot of send calls triggered within a short period of time.

The common ReceiveAndForward behavior for received UDP data with the Vector TcpIp is that the forwarded UDP data are cut off if a full UDP datagram is received and the whole data could not be forwarded within one datagram.

The common ReceiveAndForward behavior for received TCP data with the Vector TcpIp is that the forwarded TCP data may be split from the TcpIp in several small data chunks. This would happen if the received data in the receive ring buffer are copied over the buffer's wraparound. In this case the TcpIp generate two separate RxInidications for which the Etm generate an own ReceiveAndForward event for each of the RxIndications. In this cases the Tester should not wait for an fix amount of Etm event messages but should read out the event field with the amount of received data bytes until all the data are received.

3.5 Error Handling

3.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `ETM_DEV_ERROR_DETECT==STD_ON`).

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

The reported Etm ID is 170. This is also the TcpIp ID! The reported Etm instance ID is 5.

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
0x00U	Etm_Init
0x01U	Etm_GetVersionInfo
0x02U	Etm_Mainfunction
0x03U	Etm_RxIndication
0x04U	Etm_TxConfirmation (not yet used)
0x05U	Etm_TcpAccepted (not yet used)
0x06U	Etm_TcpConnected (not yet used)
0x07U	Etm_TcpIpEvent (not yet used)
0x08U	Etm_ActivateProcessing
0x09U	Etm_DeactivateProcessing

Service ID	Service
0xFFU	For internal Etm services

Table 3-5 Service IDs

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

Error Code	Description
0x01U	ETM_E_NOT_INITIALIZED
0x02U	ETM_E_ALREADY_INITIALIZED
0x03U	ETM_E_INV_CONFIG
0x04U	ETM_E_NULL_POINTER
0x05U	ETM_E_INV_POINTER
0x06U	ETM_E_INV_PARAM
0x07U	ETM_E_INV_SOCK_HND
0x08U	ETM_E_INV_SOCK_ADDR_FAMILY
0x09U	ETM_E_MSGSIZE
0x10U	ETM_E_UNKNOWN_MESSAGE
0x11U	ETM_E_NOT_INIT_LISTEN
0x12U	ETM_E_NOT_LOCAL_BUFFER_OVERFLOW

Table 3-6 Errors reported to DET

4 Integration

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

4.1 Scope of Delivery

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

4.1.1 Static Files

File Name	Description
Etm.c	This is the source file of the Etm module
Etm.h	API declaration (public)
Etm_Cbk.h	API declaration (callbacks)
Etm_Types.h	Data types declaration

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator Pro.

File Name	Description
Etm_Cfg.h	Parameter configuration
Etm_Lcfg.(c|h)	Parameter configuration

Table 4-2 Generated files

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

The types defined by the Etm are described in this chapter.

Type Name	C-Type	Description	Value Range
Etm_ConfigType	uint8*	Configuration pointer	
Etm_CmdChanStateType	uint8	Command Channel State	`0 = INACTIVE`
			`1 = DO_GETSOCKET`
			`2 = DO_BIND`
			`3 = DO_LISTEN`
			`4 = DO_BUFFER_RX`
			`5 = DO_BUFFER_TX`
			`6 = ACTIVE`

Table 5-1 Type definitions

Etm_HeaderType

Struct Element Name	C-Type	Description	Value Range
Sid	uint16	SID – Service ID	
Evb	boolean	EVB – Event Bit	
Gid	uint8	GID – Group ID	
Pid	uint8	PID – Service Primitive ID	
Len	uint32	LEN – Length	
ProtocolVersion	uint8	SOME/IP Protocol Version	`0x01`
InterfaceVersion	uint8	SOME/IP Interface Version	`0x01`
Tid	uint8	TID – Type ID	
Rid	uint8	RID – Result ID	
PayloadLength	uint16		
Payload	uint8*		
ReceivedSocketId	TcpIp_SocketIdType		
ReceivedRemoteAddress	Tcpip_SockAddrInXType		
UsePhysAddr	boolean	Answer UserScipts via Eth	

Table 5-2 Etm_HeaderType

Etm_CommandSocketType

Struct Element Name	C-Type	Description	Value Range
SocketId	TcpIp_SocketIdType		

Struct Element Name	C-Type	Description	Value Range
ChannelState	Etm_CmdChanStateType		

Table 5-3 Etm_CommandSocketType

Etm_TestSocketType

Struct Element Name	C-Type	Description	Value Range
SocketId	TcpIp_SocketIdType		
RecvFwdActive	boolean	ReceiveAndForward state	
RecvFwdDropCount	uint32	Dropped bytes (while inactive)	
RecvFwdMaxForward	uint16	Forward length per event	
RecvFwdMaxLength	uint16	Forward length over all	

Table 5-4 Etm_TestSocketType

5.2 Services provided by Etm

5.2.1 Etm_InitMemory

> Task.	
`void Etm_InitMemory (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Function for *_INIT_*-variable initialization. Service to initialize module global variables at power up. This function initializes the variables in *_INIT_* sections. Used in case they are not initialized by the startup code.	
Particularities and Limitations	
Module is uninitialized.	
Call context	

Table 5-5 Etm_InitMemory

5.2.2 Etm_Init

> Task.	
`void Etm_Init (ETM_P2C(Etm_ConfigType) ConfigPtr)`	
Parameter	
ConfigPtr [in]	Configuration structure for initializing the module.
Return code	
void	none

Functional Description
Initialization function. This function initializes the module Etm. It initializes all variables and sets the module state to initialized.
Particularities and Limitations
Specification of module initialization > Interrupts are disabled.Module is uninitialized.Etm_InitMemory has been called unless Etm_ModuleInitialized is initialized by start-up code.
Call context

Table 5-6 Etm_Init

5.2.3 Etm_MainFunction

> Task.	
`void` **`Etm_MainFunction`** `(void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Schedules the Etm module. (Entry point for scheduling).	
Particularities and Limitations	
Module is uninitialized.	
Call context	

Table 5-7 Etm_MainFunction

5.2.4 Etm_ActivateProcessing

> Task.	
`void` **`Etm_ActivateProcessing`** `(void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Activates the Service Primitive and command processing. Etm_ActivateProcessing() starts the message procesing. This service can, for example, be triggered by a diagnostic message.	
Particularities and Limitations	

Call context

Table 5-8 Etm_ActivateProcessing

5.2.5 Etm_DeactivateProcessing

> Task.	
`void Etm_DeactivateProcessing (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Deactivates the Service Primitive and command processing. Etm_DeactivateProcessing() stops the message procesing. This service can, for example, be triggered by a diagnostic message.	
Particularities and Limitations	
Call context	

Table 5-9 Etm_DeactivateProcessing

5.3 Services used by Etm

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

1 Component	API
DET	Det_ReportError
TcpIp	TcpIp_Bind
TcpIp	TcpIp_ChangeParameter
TcpIp	TcpIp_ClearARCache
TcpIp	TcpIp_Close
TcpIp	TcpIp_EtmGetSocket
TcpIp	TcpIp_TcpConnect
TcpIp	TcpIp_TcpListen
TcpIp	TcpIp_TcpReceived
TcpIp	TcpIp_TcpTransmit
TcpIp	TcpIp_UdpTransmit
TcpIp (IpV4)	IPV4_HTONL
TcpIp (IpV4)	IPV4_HTONS
TcpIp (IpV4)	IPV4_UINT8_HTONS

1 Component	API
TcpIp (IpV6)	IPV6_HTONL
TcpIp (IpV6)	IPV6_HTONS
TcpIp (IpV6)	IPV6_UINT8_HTONS

Table 5-10 Services used by the Etm

5.4 Callback Functions

This chapter describes the callback functions that are implemented by the Etm and can be invoked by other modules. The prototypes of the callback functions are provided in the header file `Etm_Cbk.h` by the Etm.

5.4.1 Etm_RxIndication

> INTERRUPT	
`void` **`Etm_RxIndication`** `(const TcpIp_SocketIdType SocketId, const TcpIp_SockAddrType *RemoteAddrPtr, const uint8 *BufPtr, const uint16 BufLength)`	
Parameter	
SocketId [in]	Socket identifier of the related local socket resource.
RemoteAddrPtr [in]	Pointer to memory containing IP address and port of the remote host which sent the data.
BufPtr [in]	Pointer to the received data.
BufLength [in]	Data length of the received TCP segment or UDP datagram.
Return code	
void	none
Functional Description	
The TCP/IP stack calls this function after the reception of data on a socket.	
Particularities and Limitations	
Module is initialized.	
!	**Caution** The frame buffer has to be released later.
Call context	

Table 5-11 Etm_RxIndication

5.4.2 Etm_TcpAccepted

> INTERRUPT	
`Std_ReturnType` **`Etm_TcpAccepted`** `(const TcpIp_SocketIdType SocketId, const TcpIp_SocketIdType SocketIdConnected, const TcpIp_SockAddrType *RemoteAddrPtr)`	
Parameter	
SocketId [in]	Socket identifier of the related local socket resource which has been used at TcpIp_Bind().

SocketIdConnected [in]	Socket identifier of the local socket resource used for the established connection.
RemoteAddrPtr [in]	IP address and port of the remote host.
Return code	
Std_ReturnType	E_OK Accepts the established connection.
Std_ReturnType	E_NOT_OK Refuses the established connection, TcpIp stack shall close the connection.
Functional Description	
This function gets called if the stack put a socket into the listen mode before (as server) and a peer connected to it (as client).	
Particularities and Limitations	
Module is initialized. The TCP/IP stack calls this function after a socket was set into the listen state with TcpIp_TcpListen() and a TCP connection is requested by the peer.	
Call context	

Table 5-12 Etm_TcpAccepted

5.4.3 Etm_TcpPreAccepted

> INTERRUPT	
`Std_ReturnType` **`Etm_TcpPreAccepted`** `(const TcpIp_SocketIdType SocketId, const TcpIp_SocketIdType SocketIdConnected, const TcpIp_SockAddrType *RemoteAddrPtr)`	
Parameter	
SocketId [in]	Socket identifier of the related local socket resource which has been used at TcpIp_Bind().
SocketIdConnected [in]	Socket identifier of the local socket resource used for the established connection.
RemoteAddrPtr [in]	IP address and port of the remote host.
Return code	
Std_ReturnType	E_OK Return always E_OK.
Functional Description	
This function is similar to 'Etm_TcpAccepted'.	
Particularities and Limitations	
Module is initialized. This function is similar to 'Etm_TcpAccepted'. This function is called right after receiving a 'SYN' on a TCP listen socket and reports the listen socket ID and the ID of the socket that will accept the connection request. These sockets are the same ones that are reported in the offical call 'Etm_TcpAccepted' later on. The reported socket IDs may only be used for analysis purposes, the accepting socket may not be used in any way before it is fully connected (reported by call of 'Etm_TcpAccepted'). This function is needed if the socket IDs are needed in the TCP states SYN-RECV or SYN-SEND.	
Call context	

Table 5-13 Etm_TcpPreAccepted

5.4.4 Etm_TcpIpEvent

> INTERRUPT	
`void **Etm_TcpIpEvent** (const TcpIp_SocketIdType SocketId, const IpBase_TcpIpEventType Event)`	
Parameter	
SocketId [in]	Socket identifier of the related local socket resource.
Event [in]	This parameter contains a description of the event just encountered.
Return code	
void	none
Functional Description	
This function gets called if the stack encounters a condition described by the values in TcpIpEvent.	
Particularities and Limitations	
Module is initialized.	
Call context	

Table 5-14 Etm_TcpIpEvent

5.4.5 Etm_LocalIpAddrAssignmentChg

> INTERRUPT	
`void **Etm_LocalIpAddrAssignmentChg** (const TcpIp_LocalAddrIdType IpAddrId, const TcpIp_IpAddrStateType State)`	
Parameter	
LocalAddrId [in]	IP address Identifier, representing an IP address specified in the TcpIp module configuraiton (e.g. static IPv4 address on EthIf controller 0).
State [in]	State of IP address assignment.
Return code	
void	none
Functional Description	
This function gets called by the TCP/IP stack if an IP address assignment changes (i.e. new address assigned or assigned address becomes invalid).	
Particularities and Limitations	
Module is initialized.	
Call context	

Table 5-15 Etm_LocalIpAddrAssignmentChg

5.4.6 Etm_CopyTxData

> INTERRUPT	
`BufReq_ReturnType Etm_CopyTxData (const TcpIp_SocketIdType SocketId, uint8 *BufPtr, const uint16 BufLength)`	
Parameter	
SocketId [in]	Socket identifier of the related local socket resource.
BufPtr [in]	Pointer to buffer for transmission data.
BufLength [in]	Length of provided data buffer.
Return code	
BufReq_ReturnType	BUFREQ_OK Data has been copied to the transmit buffer completely as requested.
BufReq_ReturnType	BUFREQ_E_NOT_OK Data has not been copied. Request failed. (No further action for TcpIp required. Later the upper layer might either close the socket or retry the transmit request)
Functional Description	
This function requests to copy data for transmission to the buffer indicated. This call is triggered by TcpIp_Transmit(). Note: The call to Etm_CopyTxData() may happen in the context of TcpIp_Transmit().	
Particularities and Limitations	
Module is initialized.	
Call context	

Table 5-16 Etm_CopyTxData

5.5 Configurable Interfaces

5.5.1 Etm_GetVersionInfo

> Task.	
`void Etm_GetVersionInfo (ETM_P2V(Std_VersionInfoType) VersionInfoPtr)`	
Parameter	
VersionInfoPtr [out]	Pointer to where to store the version information. Parameter must not be NULL.
Return code	
void	none
Functional Description	
Returns the version information. Etm_GetVersionInfo() returns version information, vendor ID and AUTOSAR module ID of the component.	
Particularities and Limitations	
none	
Call context	

Table 5-17 Etm_GetVersionInfo

6 Configuration

In the Etm the attributes can be configured with the tool DaVinci Configurator Pro.

6.1 Configuration Variants

The Etm supports the configuration variants

- `VARIANT_PRECOMPILE`

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

6.2 Configuration with DaVinci Configurator Pro

For a detailed description of the configuration parameters and possible values see the help texts in the configuration tool. The help texts are extracted from the module description file (bswmd) and therefore can be found there, too.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
DaVinci Configurator Pro	Generation tool for MICROSAR components (only AUTOSAR v4)

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
BSWMD	Basis Software Module Description
DET	Development Error Tracer
ECU	Electronic Control Unit
ETM	Ethernet Testability Module for TCP/IP for interaction with external testers
HIS	Hersteller Initiative Software
IP	Internet Protocol
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Table 7-2 Abbreviations

8

9

10

11

12 Contact

Visit our website for more information on

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

www.vector.com