MICROSAR XCP on TcpIp

Technical Reference

TcpIp Transport Layer

Version 1.04.00

Status	Released

1 Document Information

1.1 History

Date	Version	Remarks
2008-11-10	1.00.00	Creation
2012-06-12	1.01.00	Adaptation to SoAd
2014-07-11	1.02.00	Minor rework of document for AR4
2015-04-23	1.03.00	ESCAN00080791: Support of Resume Mode ESCAN00077236: AR3-2679: Description BCD-coded return-value of TcpIpXcp_GetVersionInfo() in TechRef
2015-10-21	1.03.01	ESCAN00085995: Describe usage of TCPIPXCP_ENABLE_PDUMODE switch
2016-09-22	1.04.00	ESCAN00091918: FEAT-1980: Add Multi Client / Multi Connection support

Table 1-1 History of the document

1.2 Reference Documents

No.	Title	Version
[1]	ASAM_XCP_Part3-Transport-Layer-Specification_XCPonEthernet(TCP_IP&UDP_IP)_V1-1-0.pdf	V1.1
[2]	AUTOSAR_SWS_DET.pdf	V2.2.1
[3]	AUTOSAR_SWS_DEM.pdf	V2.2.0
[4]	ASAM_XCP_Part2-Protocol-Layer-Specification_V1-1-0.pdf	V1.1
[5]	TechnicalReference_XCP_Protocol_Layer.pdf	-

Table 1-2 Reference documents

1.3 Scope of the Document

This technical reference describes the general use of the AUTOSAR XCP on TcpIp Transport Layer.

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

2 Component History

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

Component Version	New Features
1.0	First Version
2.0	Adaptation to SoAd

Table 2-1 Component history

3 Introduction

This document describes the features, API, configuration and integration of the XCP Transport Layer for TcpIp. The XCP Protocol Layer, which is already described within a separate document [5], is not covered by this document.

Please note that in this document the term Application is not used strictly for the user software but also for any higher software layer, like e.g. a Communication Control Layer. Therefore, Application refers to any of the software components using XCP on TcpIp.

The API of the functions is described in a separate chapter at the end of this document.

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	XCP on TcpIp_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	XCP on TcpIp_MODULE_ID	255 decimal

* For the precise AUTOSAR Release 4.x please see the release specific documentation.

The XCP on TcpIp Transport Layer is responsible for bus abstraction and can be used in combination with the Vector TcpIp Stack for TcpIp communication.

3.1 Architecture Overview

The following figure shows the interfaces to adjacent modules of the XCP on TcpIp. These interfaces are described in chapter 6.

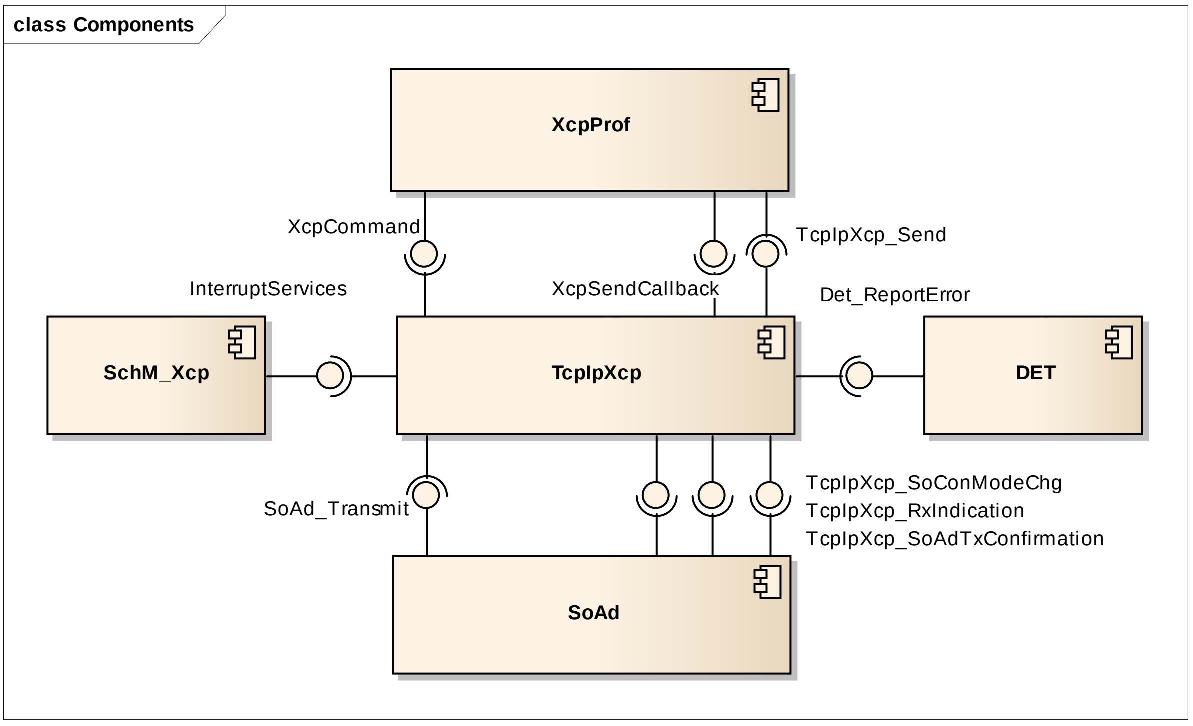

Figure 3-1 Interfaces to adjacent modules of the XCP on TcpIp

4 Functional Description

4.1 Initialization

Initialization is done by calling `TcpIpXcp_Init` with a pointer to the configuration data `TcpIpXcp_Config` as parameter.

4.2 States

The following figure shows the states the XCP on TcpIp can enter.

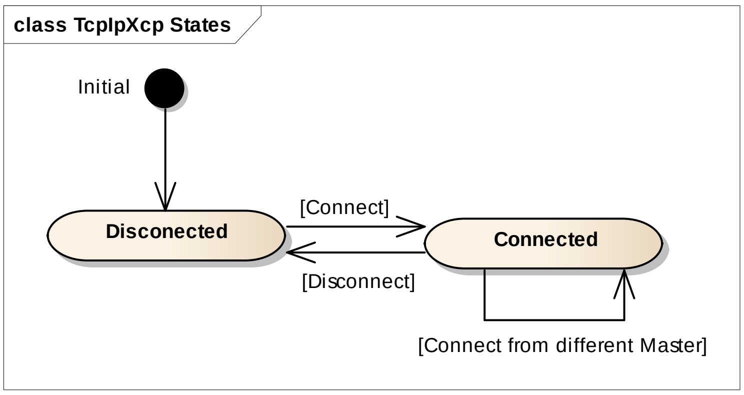

Figure 4-1 XCP on TcpIp States

After Initialization the XCP is in state Disconnected. With the first received connect, the Connected state is entered. In this state a connection from a different Master will interrupt an existing connection. Please keep this in mind.

4.3 Main Functions

The XCP on TcpIp Transport Layer provides one main function `TcpIpXcp_MainFunction` which has to be called cyclically. This is usually done by the SchM.

4.4 Critical Sections / Exclusive Areas

The XCP makes use of interrupt locking to guarantee atomic operation of critical sections. For this purpose one exclusive area is defined

- TCPIPXCP_EXCLUSIVE_AREA_0

The exclusive area must be mapped to interrupt lock and unlock functions which can be called nested. The exclusive areas are used in the following cases:

4.4.1 TCPIPXCP_EXCLUSIVE_AREA_0

This area is used whenever the services Xcp_Event, Xcp_SendCallBack, Xcp_MainFunction and Xcp_Command can interrupt each other.
Please read the Technical Reference XCP Protocol Layer [5] for further information.

4.5 Error Handling

4.5.1 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 `TCPIPXCP_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 XCP on TcpIp ID is 255.

The reported service IDs identify the services which are described in 6.2. The following table presents the service IDs and the related services:

Service ID	Service
0	`TCPIPXCP_RXINDICATION_SERVICE_ID`
1	`TCPIPXCP_TXCONFIRMATION_SERVICE_ID`
2	`TCPIPXCP_SOCONMODECHG_SERVICE_ID`
3	`TCPIPXCP_SEND_SERVICE_ID`
4	`TCPIPXCP_SENDFLUSH_SERVICE_ID`
5	`TCPIPXCP_TLSERVICE_SERVICE_ID`
6	`TCPIPXCP_MAINFUNCTION_SERVICE_ID`
7	`TCPIPXCP_INIT_SERVICE_ID`
8	`TCPIPXCP_GETVERSIONINFO_SERVICE_ID`
11	`TCPIPXCP_DAQRESUMEGET_ID`
12	`TCPIPXCP_DAQRESUMESTORE_ID`

Table 4-1 Mapping of service IDs to services

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

Error Code		Description
0	`TCPIPXCP_E_NULL_POINTER`	A null pointer has been used
1	`TCPIPXCP_E_INV_SOCK_IDX`	A socket could not be acquired
3	`TCPIPXCP_E_NOT_INITIALIZED`	The component has not been initialized. Please call TcpIpXcp_Init service first.

Table 4-2 Errors reported to DET

4.5.1.1 Parameter Checking

AUTOSAR requires that API functions check the validity of their parameters. The checks in Table 4-3 are internal parameter checks of the API functions. These checks are for development error reporting and can be en-/disabled separately. The configuration of en-/disabling the checks is described in chapter 7.

The following table shows which parameter checks are performed on which services:

Service \ Check	TCPIPXCP_E_NULL_POINTER	TCPIPXCP_E_INV_SOCK_IDX	TCPIPXCP_E_NOT_INITIALIZED
Xcp_SoAdRxIndication	■		■
Xcp_SoAdTxConfirmation			■
Xcp_SoConModeChg			■
TcpIpXcp_Send			■
TcpIpXcp_TLService			■
TcpIpXcp_SendFlush			■
TcpIpXcp_MainFunction			
TcpIpXcp_InitMemory			
TcpIpXcp_Init			
TcpIpXcp_GetVersionInfo	■		
TcpIpXcp_DaqResumeGet	■		■
TcpIpXcp_DaqResumeStore	■		■

Table 4-3 Development Error Reporting: Assignment of checks to services

4.5.2 Production Code Error Reporting

The XCP on TcpIP does not report any production errors.

4.6 Resume Mode

For Resume mode the connection information must be saved. This is necessary to start sending XCP frames without prior connection from an Xcp Master. This can be done with two APIs:

void **TcpIpXcp_DaqResumeGet** (const *SoAd_SockAddrIn6Type resumeData) 6.2.9

void **TcpIpXcp_DaqResumeStore** (const * const SoAd_SockAddrIn6Type resumeData) 6.2.10

The getter method is used to get the connection information which can then be saved in NVM. The setter method is used to restore the connection information after ECU reset. These APIs must be called by the user. This can best be done in the context of the resume call-backs from the XCP protocol layer. The SoAd and the TcpIpXcp must be initialized prior calling these functions.

4.7 PDU Mode

The TcpIpXcp has a feature called PDU Mode which is used to disable transmission of XCP frames when bus communication is not available. During this time the Xcp buffers data internally.

Please note that after Initialization the XCP is in PDU Mode `TCPIPXCP_SET_OFFLINE` and will not send anything in a MICROSAR 3 environment.

Transmission of XCP frames has to be enabled manually by using the following API with `TCPIPXCP_SET_ONLINE` as parameter:

```
void TcpIpXcp_SetPduMode ( NetworkHandleType XcpNwH,
        TcpIpXcp_PduSetModeType PduMode )
```

(6.2.8)

5 Integration

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

5.1 Scope of Delivery

The delivery of the XCP on TcpIp contains the files which are described in the chapters 5.1.1 and 5.1.2:

5.1.1 Static Files

These files are not to be modified.

File Name	Description	
TcpIpXcp.c	This is the source file of the XCP on TcpIp Transport Layer.	✗
TcpIpXcp.h	This is the header file of the XCP on TcpIp Transport Layer containing prototypes.	✗
TcpIpXcp_Cbk.h	This is the call back header file of the XCP on TcpIp Transport Layer used by lower layers.	✗
TcpIpXcp_Types.h	This is the header file of the XCP on TcpIp Transport Layer containing type definitions.	✗

Table 5-1 Static files

5.1.2 Dynamic Files

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

File Name	Description	
TcpIpXcp_Cfg.h	Pre-Compile configuration header. Can be customized to the users' needs.	✎
TcpIpXcp_Lcfg.c	Link-Time configuration file. Can be customized to the users' needs.	✎
TcpIpXcp_PBcfg.c	Post-Build configuration file. Can be customized to the users' needs.	✎

Table 5-2 Generated files

5.2 Include Structure

The following picture shows the include structure of the XCP on TcpIp component

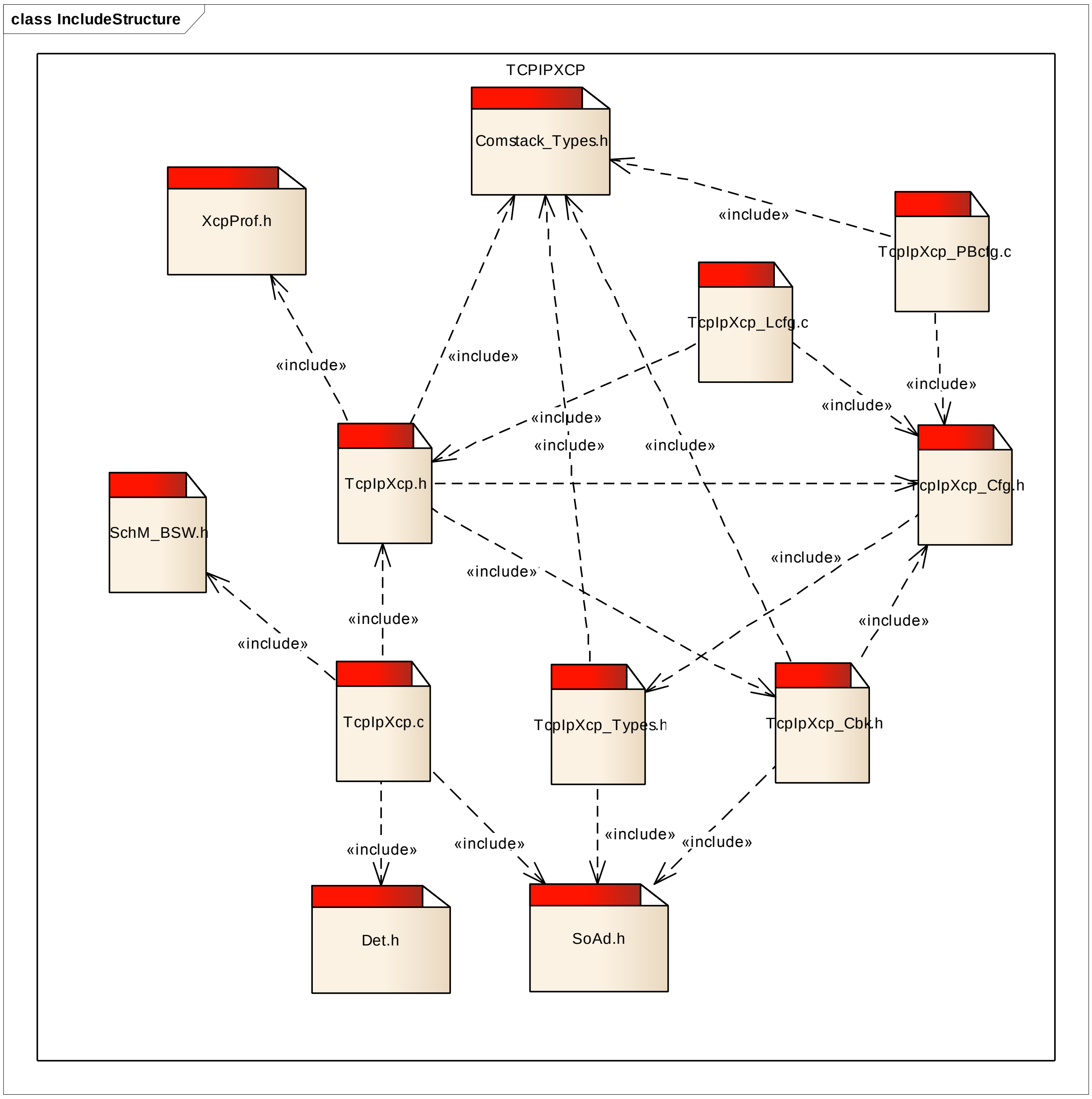

Figure 5-1 Include structure

5.3 Compiler Abstraction and Memory Mapping

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

The following table contains the memory section names and the compiler abstraction definitions for the XCP on TcpIp and illustrates their assignment among each other.

Memory Mapping Sections / Compiler Abstraction Definitions	TCPIPXCP_VAR_NOINIT	TCPIPXCP_CONST	TCPIPXCP_CODE	TCPIPXCP_PBCFG	TCPIPXCP_APPL_VAR
TCPIPXCP_START_SEC_VAR_NOINIT_UNSPECIFIED	■				
TCPIPXCP_START_SEC_VAR_NOINIT_16BIT	■				
TCPIPXCP_START_SEC_VAR_NOINIT_8BIT	■				
TCPIPXCP_START_SEC_PBCFG				■	
TCPIPXCP_START_SEC_CONST_8BIT		■			
TCPIPXCP_START_SEC_CODE			■		

Table 5-3 Compiler abstraction and memory mapping

6 API Description

For an interfaces overview please see Figure 3-1.

6.1 Type Definitions

No special types are defined by XCP on TcpIp.

6.2 Services provided by XCP on TcpIp

The XCP on TcpIp API consists of services, which are realized by function calls.

6.2.1 TcpIpXcp_Send

Prototype	

void **TcpIpXcp_Send**
(
  uint8 Xcp_Channel,
  uint8 len,
  P2CONST(uint8, AUTOMATIC, TCPIPXCP_APPL_DATA) msg
)	

Parameter	
Xcp_Channel	Logical channel of the protocol layer. Depending whether Multi Client is enabled this parameter will always be 0 (Multi Client disabled) or reflect the logical Xcp channel (Multi Client enabled).
len	Length of XCP frame
msg	Pointer to XCP frame
Return code	
None.	-
Functional Description	
This service is called whenever the Protocol Layer wants to transmit a XCP frame.	
Particularities and Limitations	
■ XCP must be initialized	
Expected Caller Context	
■ Task or interrupt context ■ Not re-entrant	

Table 6-1 TcpIpXcp_Send

6.2.2 TcpIpXcp_SendFlush

Prototype
void **TcpIpXcp_SendFlush** (uint8 Xcp_Channel,   uint8 XcpFlushTypeSel)

Parameter	
`Xcp_Channel`	Logical channel of the protocol layer. Depending whether Multi Client is enabled this parameter will always be 0 (Multi Client disabled) or reflect the logical Xcp channel (Multi Client enabled).
`XcpFlushTypeSel`	Select whether CTO, DTO or all frames need to be flushed
Return code	
None.	-
Functional Description	
This service is called whenever the Protocol Layer wants to finish a transmission.	
Particularities and Limitations	
■ XCP must be initialized	
Expected Caller Context	
■ Task or interrupt context ■ Not re-entrant	

Table 6-2 TcpIpXcp_SendFlush

6.2.3 TcpIpXcp_TLService

Prototype	

`uint8` **`TcpIpXcp_TLService`**
`(`
`  uint8 Xcp_Channel,`
`  P2CONST(uint8, AUTOMATIC, FRXCP_APPL_DATA) pCmd`
`)`	

Parameter	
`Xcp_Channel`	Logical channel of the protocol layer. Depending whether Multi Client is enabled this parameter will always be 0 (Multi Client disabled) or reflect the logical Xcp channel (Multi Client enabled).
`pCmd`	Pointer to the command string
Return code	
uint8	Always returns XCP_CMD_UNKNOWN as transport layer commands are not supported for TcpIpXcp.
Functional Description	
This service is called whenever the Protocol Layer has received a Transport Layer command.	
Particularities and Limitations	
■ XCP must be initialized	
Expected Caller Context	
■ Task or interrupt context ■ Not re-entrant	

Table 6-3 TcpIpXcp_TLService

6.2.4 TcpIpXcp_MainFunction

Prototype	
`void TcpIpXcp_MainFunction ( void )`	
Parameter	
None.	-
Return code	
None.	-
Functional Description	
Cyclical main function for internal processing.	
Particularities and Limitations	
■ XCP must be initialized	
Expected Caller Context	
■ Task context ■ Not re-entrant	

Table 6-4 TcpIpXcp_MainFunction

6.2.5 TcpIpXcp_InitMemory

Prototype	
`void TcpIpXcp_InitMemory ( void )`	
Parameter	
None.	-
Return code	
None.	-
Functional Description	
This service initializes the memory if this is not done by the startup code.	
Particularities and Limitations	
■ Has to be called before TcpIpXcp_Init	
Expected Caller Context	
■ Task context ■ Not re-entrant	

Table 6-5 TcpIpXcp_InitMemory

6.2.6 TcpIpXcp_Init

Prototype	
`void TcpIpXcp_Init ( void )`	
Parameter	
None.	-

Return code	
None.	-
Functional Description	
This service initializes the component.	
Particularities and Limitations	
■ None	
Expected Caller Context	
■ Task context ■ Not re-entrant	

Table 6-6 TcpIpXcp_Init

6.2.7 TcpIpXcp_GetVersionInfo

Prototype	

`void` **`TcpIpXcp_GetVersionInfo`**
`(`
`  P2VAR(Std_VersionInfoType, AUTOMATIC, TCPIPXCP_APPL_DATA) versioninfo`
`)`	

Parameter	
`versioninfo`	Pointer where version information can be stored.
Return code	
None.	-
Functional Description	
TcpIpXcp_GetVersionInfo() returns version information, vendor ID and AUTOSAR module ID of the component. The versions are BCD-coded.	
Particularities and Limitations	
■ None	
Expected Caller Context	
■ Task or interrupt context ■ Not re-entrant	

Table 6-7 TcpIpXcp_GetVersionInfo

6.2.8 TcpIpXcp_SetPduMode

Prototype	

`void` **`TcpIpXcp_SetPduMode`** `( NetworkHandleType XcpNwH,`
`TcpIpXcp_PduSetModeType PduMode )`	

Parameter	
`XcpNwH`	The Network Handle which must be currently 0 as only one channel is supported

`PduMode`	The Pdu Mode to be set. This is either TCPIPXCP_SET_ONLINE or TCPIPXCP_SET_OFFLINE
Return code	
None.	-
Functional Description	
This service can be used to disable communication if the bus is not available. By default communication is disabled.	
Particularities and Limitations	
■ None	
Expected Caller Context	
■ Task or interrupt context ■ Re-entrant	

Table 6-8 TcpIpXcp_SetPduMode

6.2.9 TcpIpXcp_DaqResumeGet

Prototype	
`void` **`TcpIpXcp_DaqResumeGet`** `(const *SoAd_SockAddrIn6Type resumeData )`	
Parameter	
`resumeData`	Pointer to location where the resume data information shall be stored
Return code	
None.	-
Functional Description	
This service is used to retrieve information needed for initialization of resume mode.	
Particularities and Limitations	
■ None	
Expected Caller Context	
■ Task or interrupt context ■ Re-entrant	

Table 6-9 TcpIpXcp_DaqResumeGet

6.2.10 TcpIpXcp_DaqResumeStore

Prototype	
`void` **`TcpIpXcp_DaqResumeStore`** `(const * const SoAd_SockAddrIn6Type resumeData )`	
Parameter	
`resumeData`	Pointer to location where the resume data information shall be retrieved from
Return code	
None.	-

Functional Description
This service is used to configure information needed for initialization of resume mode.
Particularities and Limitations
■ None
Expected Caller Context
■ Task or interrupt context ■ Re-entrant

Table 6-10 TcpIpXcp_DaqResumeStore

6.3 Services used by XCP on TcpIp

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

Component	API
DET	`Det_ReportError`
SoAd	`SoAd_IfTransmit`
XcpProf	`Xcp_Command`
	`Xcp_SetActiveTl`
	`Xcp_GetActiveTl`

Table 6-11 Services used by the XCP on TcpIp

6.4 Callback Functions

This chapter describes the callback functions that are implemented by the XCP on TcpIp and can be invoked by other modules. The prototypes of the callback functions are provided in the header file TcpIpXcp_Cbk.h by the XCP on TcpIp.

6.4.1 Xcp_SoAdRxIndication

Prototype

```
void Xcp_SoAdRxIndication
(
  PduIdType TcpIpXcpRxPduId,
  P2CONST(PduInfoType, AUTOMATIC, TCPIPXCP_APPL_DATA) PduInfoPtr
)
```

Parameter	
`TcpIpXcpRxPduId`	PDU ID of the received PDU
`PduInfoPtr`	Pointer to the PduInfoType structure
Return code	
None.	-

Functional Description
This service is called whenever a XCP frame was received.
Particularities and Limitations
■ XCP must be initialized
Expected Caller Context
■ Task or interrupt context ■ Not re-entrant

Table 6-12 Xcp_SoAdRxIndication

6.4.2 Xcp_SoAdTxConfirmation

Prototype

```
void Xcp_SoAdTxConfirmation
(
  PduIdType TcpIpXcpTxPduId
)
```

Parameter	
`TcpIpXcpTxPduId`	PDU ID of the confirmed PDU
Return code	
None.	-

Functional Description
This service is called whenever a XCP frame was transmitted.
Particularities and Limitations
■ XCP must be initialized ■ TxConfirmation was requested
Expected Caller Context
■ Task or interrupt context ■ Not re-entrant

Table 6-13 Xcp_SoAdTxConfirmation

6.4.3 Xcp_SoConModeChg

Prototype

```
void Xcp_SoConModeChg
(
  SoAd_SoConIdType SoConId,
  SoAd_SoConModeType Mode
)
```

Parameter	
`SoConId`	Connection ID
`Mode`	Mode whether it is online or offline

Return code	
None.	-
Functional Description	
This service is called whenever the cable is unplugged	
Particularities and Limitations	
■ XCP must be initialized	
Expected Caller Context	
■ Task or interrupt context ■ Not re-entrant	

Table 6-14 Xcp_SoConModeChg

7 Configuration

When no GenTool is used with XCP on TcpIp the attributes can be configured manually in the configuration files:

7.1 TcpIpXcp_Cfg.h

This config file contains the following pre-compile parameters:

Parameter	Value Range	Description
`ETHXCP_TRANSPORT_LAYER_VERSION`	0x0100	Version of Transport Layer, do not modify.
`TCPIPXCP_DEV_ERROR_DETECT`	■ STD_ON ■ STD_OFF	Development Error Detection can be enabled with this switch.
`TCPIPXCP_VERSION_INFO_API`	■ STD_ON ■ STD_OFF	The Version Info API can be enabled with this switch.
`TCPIPXCP_CONFIG_VARIANT`	1, 2, 3	Selects the configuration variant 1=Pre-Compile 2=Link-Time 3=Post-Build
`TCPIPXCP_MODE_CHG_API`	■ STD_ON ■ STD_OFF	Select whether the SoConModeChg API is used or not
`TCPIPXCP_PROTOCOL_FORMAT`	■ `TCPIPXCP_PROTOCOL_UDP` ■ `TCPIPXCP_PROTOCOL_TCP`	Select whether UDP or TCP protocol is used
`kTcpIpXcpMaxCTO`	■ 8..255	Maximum size of Command Transfer Objects (e.g. for polling).
`kTcpIpXcpMaxDTO`	■ 8..255	Maximum size of Data Transfer Objects (e.g. for DAQ).

7.2 TcpIpXcp_Lcfg.c/TcpIpXcp_PBcfg.c

Parameter	Value Range	Description
`TcpIpXcp_PduIdField`	-	Field containing Rx and Tx PDU IDs for each channel.

8 Limitations

8.1 Maximum DTO

The XCP on TcpIp component limits the maximum DTO size to 255 bytes.

8.2 TCP

Currently the TCP protocol is not supported as segmented frames are not resolved.

For reliable operation only UDP can be used. Please configure this in the SoAd accordingly.

9 Glossary and Abbreviations

9.1 Glossary

Term	Description
GENy	Generation tool for CANbedded and MICROSAR components

Table 9-1 Glossary

9.2 Abbreviations

Abbreviation	Description
API	**A**pplication **P**rogramming **I**nterface
ASAM	**A**ssociation for **S**tandardization of **A**utomation and **M**easuring Systems
AUTOSAR	**Aut**omotive **O**pen **S**ystem **Ar**chitecture
BSW	**B**asis **S**oft**w**are
CANape	Calibration and Measurement Data Acquisition for Electronic Control Systems
CTO	**C**ommand **T**ransfer **O**bject
DAQ	Synchronous **D**ata **Aq**uistion
DEM	**D**iagnostic **E**vent **M**anager
DET	**D**evelopment **E**rror **T**racer
DTO	**D**ata **T**ransfer **O**bject
ECU	**E**lectronic **C**ontrol **U**nit
HIS	**H**ersteller **I**nitiative **S**oftware
ISR	**I**nterrupt **S**ervice **R**outine
MICROSAR	**Micr**ocontroller **O**pen **S**ystem **Ar**chitecture (the Vector AUTOSAR solution)
RTE	**R**un**t**ime **E**nvironment
SRS	**S**oftware **R**equirement **S**pecification
SWC	**S**oft**w**are **C**omponent
SWS	**S**oft**w**are **S**pecification
TCP/IP	**T**ransmission **C**ontrol **P**rotocol **/** **I**nternet **P**rotocol
XCP	Universal Measurement and **C**alibration **P**rotocol

Table 9-2 Abbreviations

10 Contact

Visit our website for more information on

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

www.vector-informatik.com