

MICROSAR CCP

Technical Reference

Version 1.01.00

Status	Released
--------	----------

Document Information

History

Date	Version	Remarks
2012-01-16	1.00.00	Creation of document
2012-04-16	1.00.01	ESCAN00058365 Fix ranges in TechRef to match GENy
2014-04-24	1.00.02	ESCAN00075168 Add description of template _Ccp_Appl.c ESCAN00077229 AR3-2679: Description BCD-coded return-value of <u>Ccp_GetVersionInfo()</u> in <u>TechRef</u> ESCAN00076196 Broadcast address feature not configurable
2017-01-19	1.01.00	Support of Microsar 4

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_DET.pdf	3.1.0
[2]	AUTOSAR	AUTOSAR_SWS_DEM.pdf	4.1.0
[3]	AUTOSAR	AUTOSAR_BasicSoftwareModules.pdf	V1.0.0
[4]	ASAM	CCP21.pdf	2.1
[5]	AUTOSAR	AUTOSAR_SWS_CANInterface.pdf	4.1.0

Scope of the Document

This technical reference describes the specific use of the CCP driver software.

**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

1	Component History	8
2	Introduction.....	9
2.1	Architecture Overview	9
3	Functional Description	11
3.1	Initialization.....	11
3.2	States	11
3.3	Main Functions	11
3.4	Error Handling.....	12
3.4.1	Development Error Reporting.....	12
3.4.1.1	Parameter Checking.....	13
3.4.2	Production Code Error Reporting	13
4	Integration	14
4.1	Scope of Delivery	14
4.1.1	Static Files	14
4.1.2	Dynamic Files	14
4.2	Include Structure	15
4.3	Compiler Abstraction and Memory Mapping.....	15
4.4	Critical Sections.....	16
5	API Description.....	17
5.1	Services provided by CCP	17
5.1.1	Ccp_SendCrm	17
5.1.2	Ccp_Daq.....	17
5.1.3	Ccp_MainFunction	18
5.1.4	Ccp_Command.....	18
5.1.5	Ccp_Send.....	19

5.1.6	Ccp_SendCallBack	19
5.1.7	Ccp_InitMemory	20
5.1.8	Ccp_CanInitMemory	20
5.1.9	Ccp_Init.....	21
5.1.10	Ccp_GetVersionInfo.....	21
5.2	Services used by CCP.....	22
5.3	Callback Functions.....	22
5.3.1	Ccp_RxIndication	22
5.3.2	Ccp_TxConfirmation	23
5.4	Configurable Interfaces	23
5.4.1	Callout Functions.....	23
5.4.1.1	Ccp_CheckReadAccess	24
5.4.1.2	Ccp_CheckWriteAccess	24
5.4.1.3	Ccp_GetPointer.....	25
6	Configuration	26
6.1	Configuration Variants.....	26
7	AUTOSAR Standard Compliance	27
7.1	Deviations.....	27
7.2	Additions/ Extensions	27
7.3	Limitations	27
8	Glossary and Abbreviations	28
8.1	Glossary.....	28
8.2	Abbreviations	28
9	Contact	29

Illustrations

Figure 2-1	AUTOSAR architecture	9
Figure 2-2	Interfaces to adjacent modules of the CCP	10
Figure 3-1	States of the CCP component	11
Figure 4-1	Include structure.....	15

Tables

Table 1-1	Component history.....	8
Table 3-1	Service IDs.....	12
Table 3-2	Errors reported to DET	12
Table 3-3	Development Error Reporting: Assignment of checks to services	13
Table 4-1	Static files	14
Table 4-2	Generated files	14
Table 4-3	Compiler abstraction and memory mapping	16
Table 5-1	Ccp_SendCrm	17
Table 5-2	Ccp_Daq.....	18
Table 5-3	Ccp_MainFunction	18
Table 5-4	Ccp_Command	19
Table 5-5	Ccp_Send.....	19
Table 5-6	Ccp_SendCallback	20
Table 5-7	Ccp_InitMemory	20
Table 5-8	Ccp_CanInitMemory	21
Table 5-9	Ccp_Init.....	21
Table 5-10	Ccp_GetVersionInfo	22
Table 5-11	Services used by the CCP	22
Table 5-12	Ccp_RxIndication	23
Table 5-13	Ccp_TxConfirmation.....	23

Table 5-14	Ccp_CheckReadAccess	24
Table 5-14	Ccp_CheckWriteAccess	25
Table 5-15	Ccp_GetPointer	25
Table 8-1	Glossary	28
Table 8-2	Abbreviations	28

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
[2.00.00]	CCP extended as CDD for AUTOSAR use .
[3.00.00]	CCP extended for Microsar 4 use.

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the MICROSAR BSW module **CCP** as specified in [4].

Supported AUTOSAR Release6:	-	
Supported Configuration Variants:	pre-compile	
Vendor ID:	CCP_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CCP_MODULE_ID	255 decimal (according to ref. [3])

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

The AUTOSAR CCP can be used for measurement and calibration of ECUs.

2.1 Architecture Overview

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

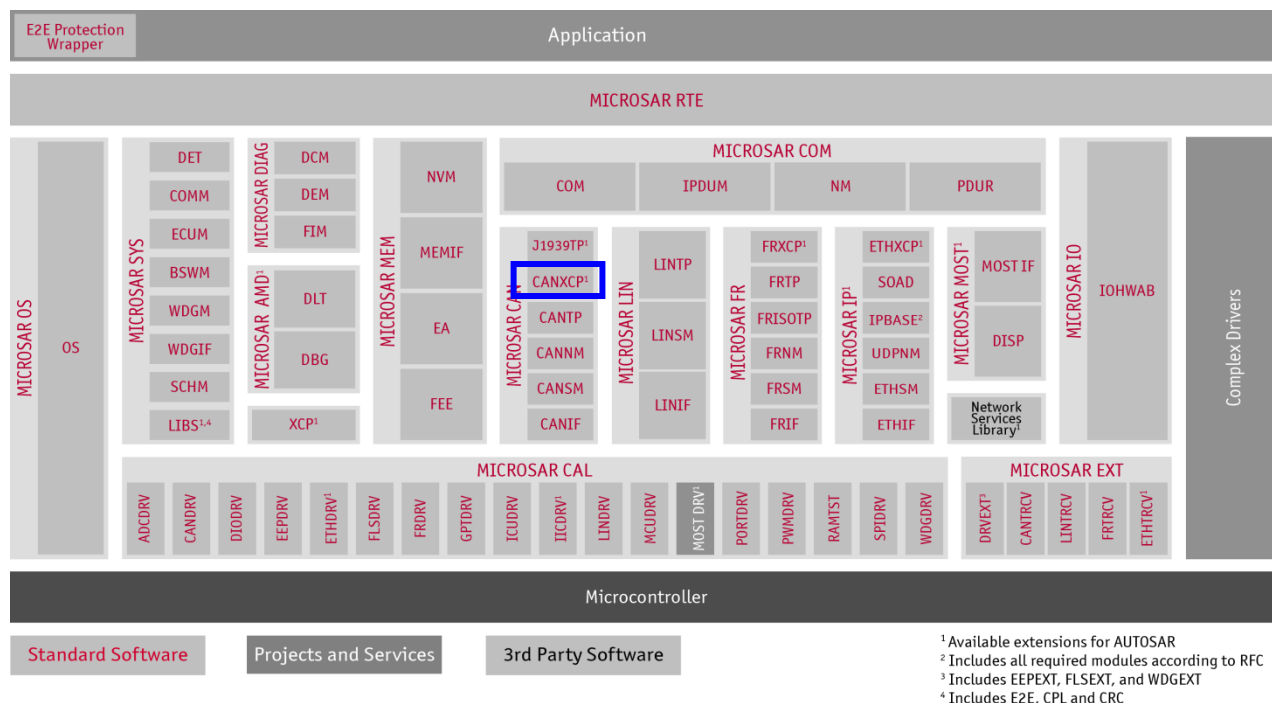


Figure 2-1 AUTOSAR architecture

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

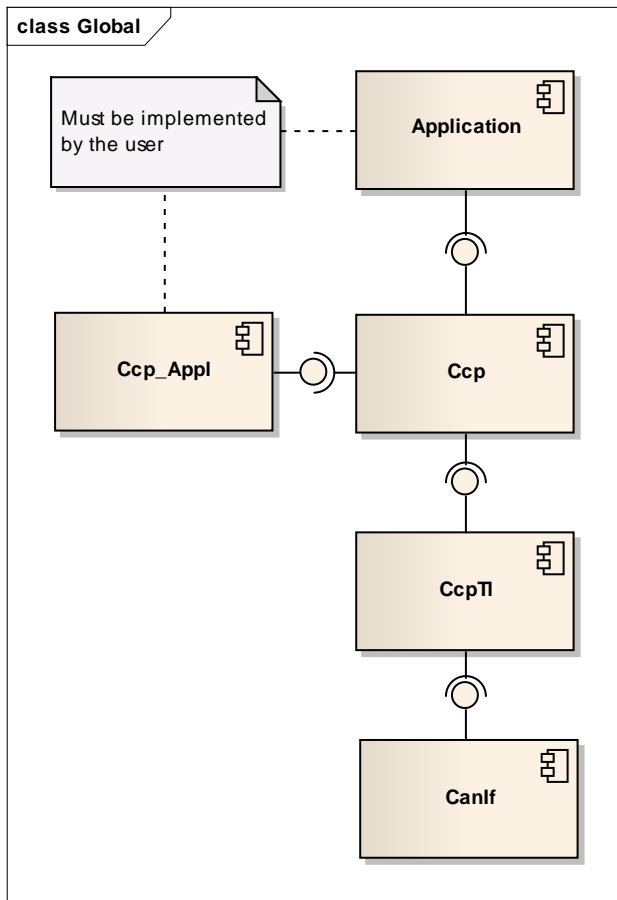


Figure 2-2 Interfaces to adjacent modules of the **CCP**

The CCP provides its Interface to be directly used by the Application. No service ports are used or provided.

3 Functional Description

3.1 Initialization

The CCP component provides two functions for initialization (see 5.1.7 for details).

FUNC(void, CCP_CODE) Ccp_InitMemory(void)

FUNC(void, CCP_CODE) Ccp_CanInitMemory(void)

The InitMemory function, which can be called when there is no startup code which performs initialization of memory with zero.

FUNC(void, CCP_CODE) Ccp_Init(void)

The Ccp_Init function which must be called to initialize the component (see 5.1.9 for details).

3.2 States

The Ccp knows two states which can be changed with the Connect/Disconnect command. This command is sent by a master tool, e.g. CANape:

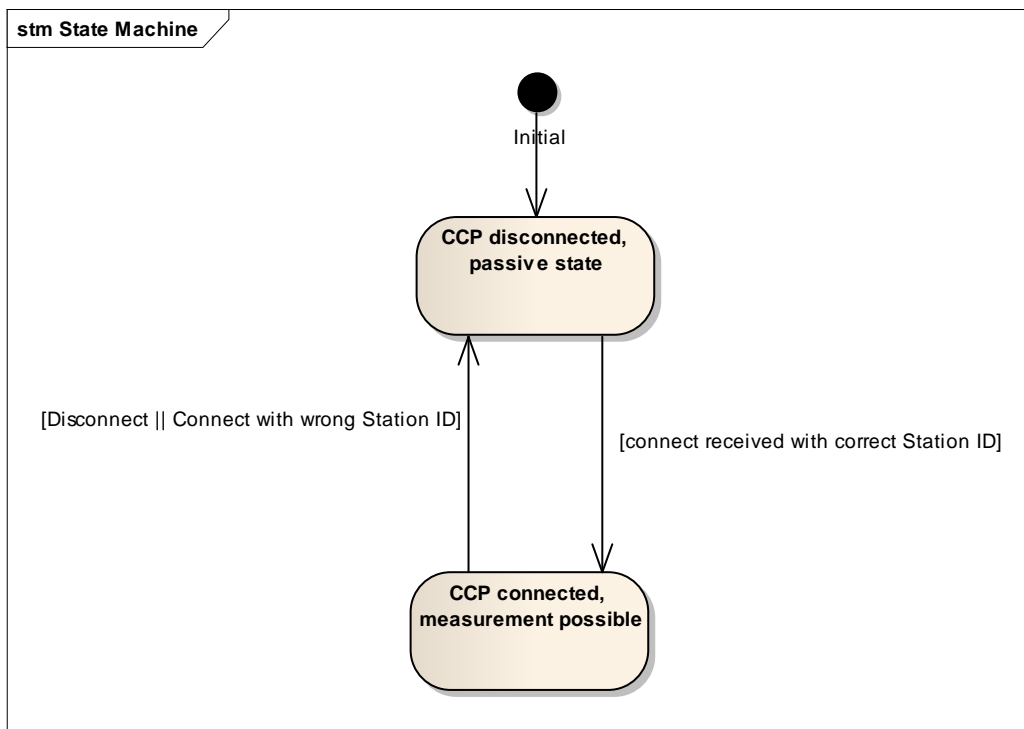


Figure 3-1 States of the CCP component

3.3 Main Functions

The Ccp provides one Mainfunction (see 5.1.3 for details):

FUNC(CCP_BYTE, CCP_CODE) Ccp_MainFunction(void)

which must be called cyclically, e.g. all 10 ms. The Mainfunction can be called with a low priority as its main purpose is to calculate CRC values of memory areas.

3.4 Error Handling

3.4.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [1], if development error reporting is enabled (i.e. pre-compile parameter **CCP_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 **CCP** ID is **255**.

The reported service IDs identify the services which are described in 5. The following table presents the service IDs and the related services:

Service ID	Service
0u	CCP_INIT_SERVICE_ID
1u	CCP_DAQ_SERVICE_ID
2u	CCP_MAINFUNCTION_SERVICE_ID
3u	CCP_COMMAND_SERVICE_ID
4u	CCP_SEND_CALLBACK_SERVICE_ID
5u	CCP_SENDCRM_SERVICE_ID
6u	CCP_QUEUEINIT_SERVICE_ID
7u	CCP_QUEUEWRITE_SERVICE_ID
8u	CCP_TXCONFIRMATION_SERVICE_ID
9u	CCP_RXINDICATION_SERVICE_ID
10u	CCP_SEND_SERVICE_ID
11u	CCP_GETVERSIONINFO_SERVICE_ID
12u	CCP_CANGETCURRENTCANID_SERVICE_ID

Table 3-1 Service IDs

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

Error Code	Description
2u	CCP_E_NOT_INITIALIZED A service was called without prior initialization of the component, i.e. the <code>Ccp_Init</code> function was not called
3u	CCP_E_INVALID_PDUID The <code>CcpTI</code> was called with an invalid PDU ID
4u	CCP_E_NULL_POINTER The <code>Ccp</code> was called with a null pointer as parameters

Table 3-2 Errors reported to DET

3.4.1.1 Parameter Checking

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

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

Service	Check	CCP_E_NOT_INITI ALIZED	CCP_E_INVALID_ PDUID	CCP_E_NULL_POI NTER
Ccp_SendCrm		■		
Ccp_QueueInit		■		
Ccp_QueueWrite		■		■
Ccp_Daq		■		
Ccp_MainFunction				
Ccp_Command		■		■
Ccp_SendCallBack		■		
Ccp_Init				■
Ccp_RxIndication		■	■	■
Ccp_TxConfirmation		■		
Ccp_Send		■		
Ccp_Reset				

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

3.4.2 Production Code Error Reporting

No production errors defined.

4 Integration

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

4.1 Scope of Delivery

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

4.1.1 Static Files





File Name	Description	
Ccp.c	This is the Ccp source file that contains the protocol layer	
Ccp.h	This is the Ccp header file which contains API definitions	
Ccp_Tl.c	This is the Ccp Transport Layer	
Ccp_Appl.c	This is the callout stubs file which must be edited by the user	

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool.

File Name	Description	
Ccp_Cfg.c	Pre-Compile time configuration source file for Ccp	
Ccp_Cfg.h	Pre-Compile time configuration header file for Ccp	
Ccp_PBcfg.c	Post Build time configuration source file for Ccp	
Ccp_PBcfg.h	Post Build time configuration header file for Ccp	

Table 4-2 Generated files

4.2 Include Structure

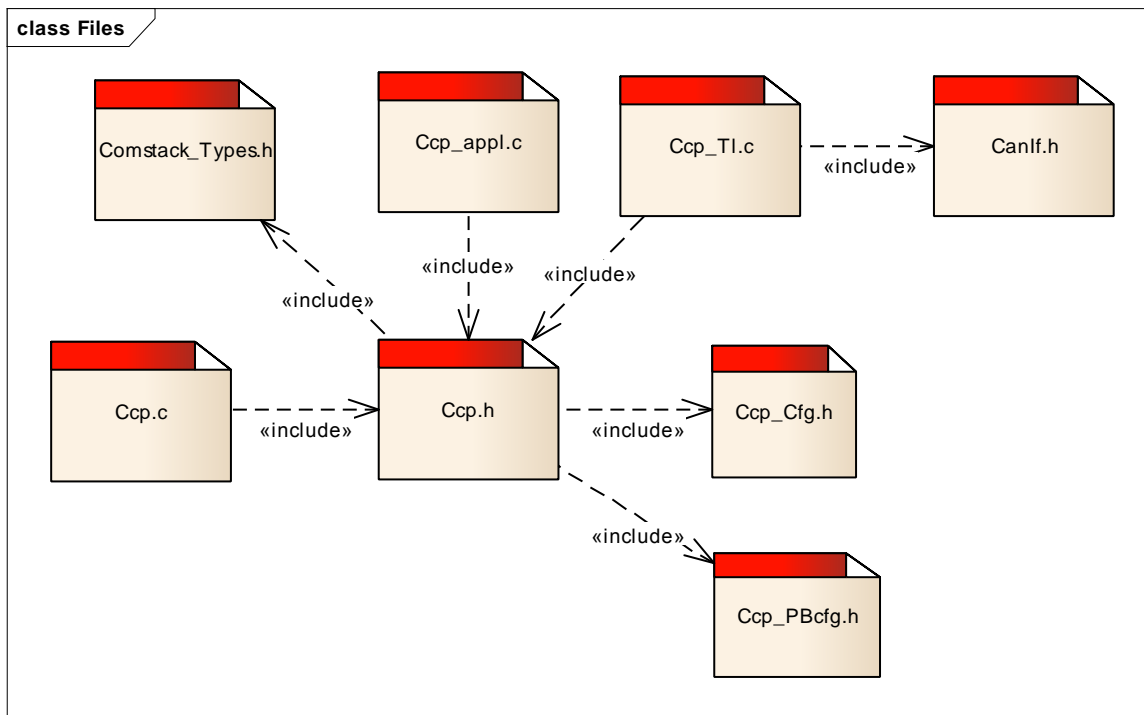


Figure 4-1 Include structure

4.3 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 of the **CCP** and illustrates their assignment among each other.

Memory Mapping Sections	Compiler Abstraction Definitions								
	CCP_CODE	CCP_VAR_ZERO_INIT	CCP_VAR_INIT	CCP_VAR_NOINIT	CCP_APPL_DATA	CCP_MTA_DATA	CCP_DAQ_DATA	CCP_PBCFG	CCP_CONST
CCP_START_SEC_CONST_8BIT CCP_STOP_SEC_CONST_8BIT									■
CCP_START_SEC_VAR_NOINIT_UNSPECIFIED CCP_STOP_SEC_VAR_NOINIT_UNSPECIFIED				■					
CCP_START_SEC_VAR_ZERO_INIT_8BIT CCP_STOP_SEC_VAR_ZERO_INIT_8BIT		■							

CCP_START_SEC_CODE	■								
CCP_STOP_SEC_CODE									
CCP_START_SEC_PBCFG								■	
CCP_STOP_SEC_PBCFG									
CCP_START_SEC_VAR_INIT_UNSPECIFIED_SAFE			■						
CCP_STOP_SEC_VAR_INIT_UNSPECIFIED_SAFE									

Table 4-3 Compiler abstraction and memory mapping

4.4 Critical Sections

The component requires one critical section to be provided by the SchM. The required functions are:

- SchM_Enter_Ccp_CCP_EXCLUSIVE_AREA_0
- SchM_Exit_Ccp_CCP_EXCLUSIVE_AREA_0

These sections are used to prevent interruption of certain critical sections that can overlap each other. The time periods are as short as possible, but note that Ccp_Send will be called with disabled interrupts.

5 API Description

For an interfaces overview please see Figure 2-2

5.1 Services provided by CCP

These services can be found in the file Ccp.c

5.1.1 Ccp_SendCrm

Prototype	
void Ccp_SendCrm (void)	
Parameter	
-	-
Return code	
-	-
Functional Description	
This service can be called by the application in case there is a pending response. This can happen if the immediate response was prevented by, e.g. an EEPROM access.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Service ID: see table '3-1' > This function is synchronous. > This function is not reentrant. 	
Expected Caller Context	
<ul style="list-style-type: none"> > This service can be called from interrupt context. 	

Table 5-1 Ccp_SendCrm

5.1.2 Ccp_Daq

Prototype	
void Ccp_Daq (CCP_BYTE eventChannel)	
Parameter	
eventChannel	Number of the event channel where a measurement cycle shall be triggered
Return code	
-	-
Functional Description	
This service triggers a DAQ measurement cycle. It must be called by the application with the respective event channel number, i.e. the event that is bound to the DAQ list by the master tool.	

Particularities and Limitations
<ul style="list-style-type: none"> > Service ID: see table '3-1' > This function is synchronous. > This function is reentrant.
Expected Caller Context
<ul style="list-style-type: none"> > This service can be called from interrupt context

Table 5-2 Ccp_Daq

5.1.3 Ccp_MainFunction

Prototype
void Ccp_MainFunction (void)
Parameter
-
Return code
-
Functional Description
This service can be called cyclically by the application. Its main purpose is to perform background tasks like CRC calculation.
Particularities and Limitations
<ul style="list-style-type: none"> > Service ID: see table '3-1' > This function is synchronous. > This function is not reentrant.
Expected Caller Context
<ul style="list-style-type: none"> > This service can be called from task context.

Table 5-3 Ccp_MainFunction

5.1.4 Ccp_Command

Prototype
void Ccp_Command (CCP_BYTEPTR com)
Parameter
com*
Pointer to command string
Return code
-

Functional Description
This function is normally called by the Transport Layer and processes the received command string. This service is usually not called by the application.
Particularities and Limitations
<ul style="list-style-type: none"> > Service ID: see table '3-1' > This function is synchronous. > This function is not reentrant.
Expected Caller Context
<ul style="list-style-type: none"> > This service can be called from interrupt context.

Table 5-4 Ccp_Command

5.1.5 Ccp_Send

Prototype	
void Ccp_Send (CCP_BYTEPTR msg)	
Parameter	
msg	Pointer to Ccp frame to be sent
Return code	
-	-
Functional Description	
This service is implemented by the Transport Layer and called by the Ccp if a frame is to be sent. This command must be acknowledged by Ccp_SendCallback once the frame is physically sent.	
Particularities and Limitations	
<ul style="list-style-type: none">> Service ID: see table '3-1'> This function is synchronous.> This function is not reentrant.	
Expected Caller Context	
<ul style="list-style-type: none">> This service can be called from interrupt context.	

Table 5-5 Ccp_Send

5.1.6 Ccp_SendCallback

Prototype	
CCP_BYTE Ccp_SendCallBack (void)	
Parameter	
-	-

Return code	
CCP_BYTE	0: Nothing pending 1: frame pending
Functional Description	
This service is normally called by the Transport Layer and acknowledges prior send requests. This service is usually not called by the application.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Service ID: see table '3-1' > This function is synchronous. > This function is not reentrant. 	
Expected Caller Context	
<ul style="list-style-type: none"> > This service can be called from interrupt context. 	

Table 5-6 Ccp_SendCallback

5.1.7 Ccp_InitMemory

Prototype	
void Ccp_InitMemory (void)	
Parameter	
-	-
Return code	
-	-
Functional Description	
This service can be called by the application and initializes the used memory.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Service ID: see table '3-1' > This function is synchronous. > This function is reentrant. 	
Expected Caller Context	
<ul style="list-style-type: none"> > This service can be called from task context. 	

Table 5-7 Ccp_InitMemory

5.1.8 Ccp_CanInitMemory

Prototype	
void Ccp_CanInitMemory (void)	

Parameter	
-	-
Return code	
-	-
Functional Description	
This service can be called by the application and initializes the used memory.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Service ID: see table '3-1' > This function is synchronous. > This function is reentrant. 	
Expected Caller Context	
<ul style="list-style-type: none"> > This service can be called from task context. 	

Table 5-8 Ccp_CanInitMemory

5.1.9 Ccp_Init

Prototype	
void Ccp_Init (void)	
Parameter	
-	-
Return code	
-	-
Functional Description	
This service must be called by the application and initializes the component	
Particularities and Limitations	
<ul style="list-style-type: none"> > Service ID: see table '3-1' > This function is synchronous. > This function is not reentrant. 	
Expected Caller Context	
<ul style="list-style-type: none"> > This service can be called from task context. 	

Table 5-9 Ccp_Init

5.1.10 Ccp_GetVersionInfo

Prototype	
void Ccp_GetVersionInfo (Std_VersionInfoType *CcpVersionInforPtr)	

Parameter	
CcpVersionInforPtr *	Pointer to version info structure
Return code	
-	-
Functional Description	
Msn_GetVersionInfo() returns version information, vendor ID and AUTOSAR module ID of the component. The versions are BCD-coded.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Service ID: see table '3-1' > This function is synchronous. > This function is reentrant. 	
Expected Caller Context	
<ul style="list-style-type: none"> > This service can be called from interrupt context. 	

Table 5-10 Ccp_GetVersionInfo

5.2 Services used by CCP

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

Component	API
DET	Det_ReportError
CanIf	CanIf_Transmit

Table 5-11 Services used by the CCP

5.3 Callback Functions

This chapter describes the callback functions that are implemented by the **CCP** and can be invoked by other modules. These services can be found in the file Ccp_Tl.c

5.3.1 Ccp_RxIndication

Prototype	
void Ccp_RxIndication (PduldType CanCanCcpRxPduld, PdulInfoType *PdulInfoPtr)	
Parameter	
CanCanCcpRxPduld	Pdu Id of the received frame
PdulInfoPtr	Pointer to pdu info struct that contains the received frame
Return code	
-	-

Functional Description
Indicates a received frame. This service is usually called by the CanIf.
Particularities and Limitations
<ul style="list-style-type: none"> > Service ID: see table '3-1' > This function is synchronous. > This function is not reentrant.
Expected Caller Context
<ul style="list-style-type: none"> > This service can be called from interrupt context.

Table 5-12 Ccp_RxIndication

5.3.2 Ccp_TxConfirmation

Prototype	
void Ccp_TxConfirmation (PduIdType CanTxPduId)	
Parameter	
CanTxPduId	The Pdu Id of the transmitted frame.
Return code	
-	-
Functional Description	
Indicates a transmitted frame. This service is usually called by the CanIf.	
Particularities and Limitations	
<ul style="list-style-type: none">> Service ID: see table '3-1'> This function is synchronous.> This function is not reentrant.	
Expected Caller Context	
<ul style="list-style-type: none">> This service can be called from interrupt context.	

Table 5-13 Ccp_TxConfirmation

5.4 Configurable Interfaces

5.4.1 Callout Functions

At its configurable interfaces the **CCP** defines callout functions. The declarations of the callout functions are provided by the BSW module, i.e. the **CCP**. It is the integrator's task to provide the corresponding function definitions. The definitions of the callouts can be adjusted to the system's needs and stubs can be found in Ccp_Appl.c. The **CCP** callout function declarations are described in the following tables:

5.4.1.1 Ccp_CheckReadAccess

Prototype	
CCP_BYTE Ccp_CheckReadAccess (CCP_MTABYTEPTR a, CCP_BYTE size)	
Parameter	
a	Address of memory area to read
size	Size of memory area to read
Return code	
CCP_BYTE	CCP_READ_DENIED if the memory area is protected, CCP_READ_OK if read access is granted.
Functional Description	
This service is notified if there was a CCP read request. This function must check if the memory address matches a protected area. When the memory area is protected a CCP_READ_DENIED must be returned. Otherwise CCP_READ_OK is returned.	
Particularities and Limitations	
> None	
Call context	
> interrupt or task context > is called in case of a Ccp read request or if a WRITE_MTA is performed.	

Table 5-14 Ccp_CheckReadAccess

5.4.1.2 Ccp_CheckWriteAccess

Prototype	
CCP_BYTE Ccp_CheckWriteAccess (CCP_MTABYTEPTR a, CCP_BYTE size)	
Parameter	
a	Address of memory area to write
size	Size of memory area to write
Return code	
CCP_BYTE	CCP_WRITE_DENIED if the memory area is protected, CCP_WRITE_OK if write access is granted.
Functional Description	
This service is notified if there was a CCP write request. This function must check if the memory address matches a protected area. When the memory area is protected a CCP_WRITE_DENIED must be returned. Otherwise CCP_WRITE_OK is returned.	
Particularities and Limitations	
> None	

Call context
<ul style="list-style-type: none">> interrupt or task context> is called in case of a Ccp write request.

Table 5-15 Ccp_CheckWriteAccess

5.4.1.3 Ccp_GetPointer

Prototype	
CCP_MTABYTEPTR Ccp_GetPointer (CCP_BYTE addr_ext, CCP_DWORD addr)	
Parameter	
addr_ext	Address extension to be converted to local address.
addr	Address to be converted to local address.
Return code	
CCP_MTABYTEPTR	Local data pointer.
Functional Description	
This service is called to convert a received address to an ECU internal address. In most cases the parameter addr is simply returned without conversation.	
Particularities and Limitations	
<ul style="list-style-type: none">> None	
Call context	
<ul style="list-style-type: none">> interrupt or task context> is called whenever a Tool address must be converted to a local address.	

Table 5-16 Ccp_GetPointer

6 Configuration

In the **CCP** the attributes can be configured according to/ with the following methods/ tools:

- > Configuration in CFG5.

6.1 Configuration Variants

The **CCP** supports the configuration variants

- > VARIANT-PRE-COMPILE
- > VARIANT-POST-BUILD

7 AUTOSAR Standard Compliance

7.1 Deviations

None.

7.2 Additions/ Extensions

The Ccp is realized as a Complex Device Driver (CDD) that sits atop the CanIf.

7.3 Limitations

- All DAQ lists contain an equal number of ODTs (CCP_MAX_ODT).
- Dynamic or static assignment of different CAN identifiers to DAQ lists is not supported.
- For performance reasons, the response to the CCP commands DNLOAD and UPLOAD does not contain the incremented MTA value.
- The resume feature is not implemented.

8 Glossary and Abbreviations

8.1 Glossary

Term	Description
CCP	Can Calibration Protocol
GENy	Generation tool for CANbedded and MICROSAR components

Table 8-1 Glossary

8.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DAQ	Data Acquisition
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 8-2 Abbreviations

9 Contact

Visit our website for more information on

- > News
 - > Products
 - > Demo software
 - > Support
- Training

29 / 29