MICROSAR Complex Device Driver

Technical Reference

DaVinci Configurator

Version 2.04.00

Authors	Safiulla Shakir, Gunnar Meiss, Markus Bart
Status	Released

Document Information

History

Author	Date	Version	Remarks
Safiulla Shakir	2012-03-23	1.00.00	Initial Version
Gunnar Meiss	2012-08-08	2.00.00	Support AUTOSAR 4
Gunnar Meiss	2013-05-13	2.00.01	performed review rework
Markus Bart	2014-02-05	2.01.00	Support J1939Rm Contribution
Markus Bart	2014-02-28	2.02.00	Support the StartOfReception API with the PduInfoType according to ASR4.1.2
Gunnar Meiss	2014-05-07	2.02.00	AR4-769: ESCAN00075414 AR4-744: Cdd shall support CddSoAdUpperLayerContribution as an extension to AR 4.0.3 (schema shall remain at AR 4.0.3)
Gunnar Meiss	2016-02-24	2.03.00	FEAT-1631: Trigger Transmit API with SduLength In/Out according to ASR4.2.2
Gunnar Meiss	2017-01-09	2.04.00	Rename TechnicalReference_Cdd.pdf to TechnicalReference_Cdd_Communication.pdf

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_TPS_ECUConfiguration.pdf	3.2.0
[2]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	1.6.0

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.

Caution
This symbol calls your attention to warnings.

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
1.00.00	First DaVinci Configurator AUTOSAR 3 Version
2.00.00	Added support of AUTOSAR 4
3.00.00	Added support for Java 7 & SoAd & DoIp
3.01.00	Added support for J1939Rm
3.02.00	Added support for StartOfReception with the PduInfoType according to ASR4.1.2 AR4-769: Support Cdd API-SERVICE-PREFIX Parameter as APIs and SNV Prefix AR4-744: Cdd shall support CddSoAdUpperLayerContribution as an extension to AR 4.0.3 (schema shall remain at AR 4.0.3)

Table 1-1 Component history

2 Introduction

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

Supported AUTOSAR Release*:	4.x	
Supported Configuration Variants:	pre-compile	
Vendor ID:	<CDD>_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	<CDD>_MODULE_ID	255 decimal (according to ref. [2])

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

Any software module which is a part of the standard AUTOSAR architecture but not a basic software module can be implemented and treated as a Complex Device Driver. This technical reference describes the configurator for complex device driver configuration.

In the AUTOSAR COM stack upper and lower layer Complex Device Drivers are allowed to access the PDUs. The PDUs that are exchanged between the CDD and the PDU router (in case the CDD is upper layer or lower layer for the PduR) or between the CDD and communication hardware abstraction layer modules (in case the CDD is lower layer) shall be configured. The contribution of the Complex Device Driver implies a reference to the global PDU and the definition of a `HandleId`. Figure 2 - 1 shows an example of a Complex Device Driver to the CANIF (lower layer) and one Complex Device Driver (upper layer) above the PDUR.

2.1 Architecture Overview

The following figure shows where the <CDD> is located in the AUTOSAR architecture.

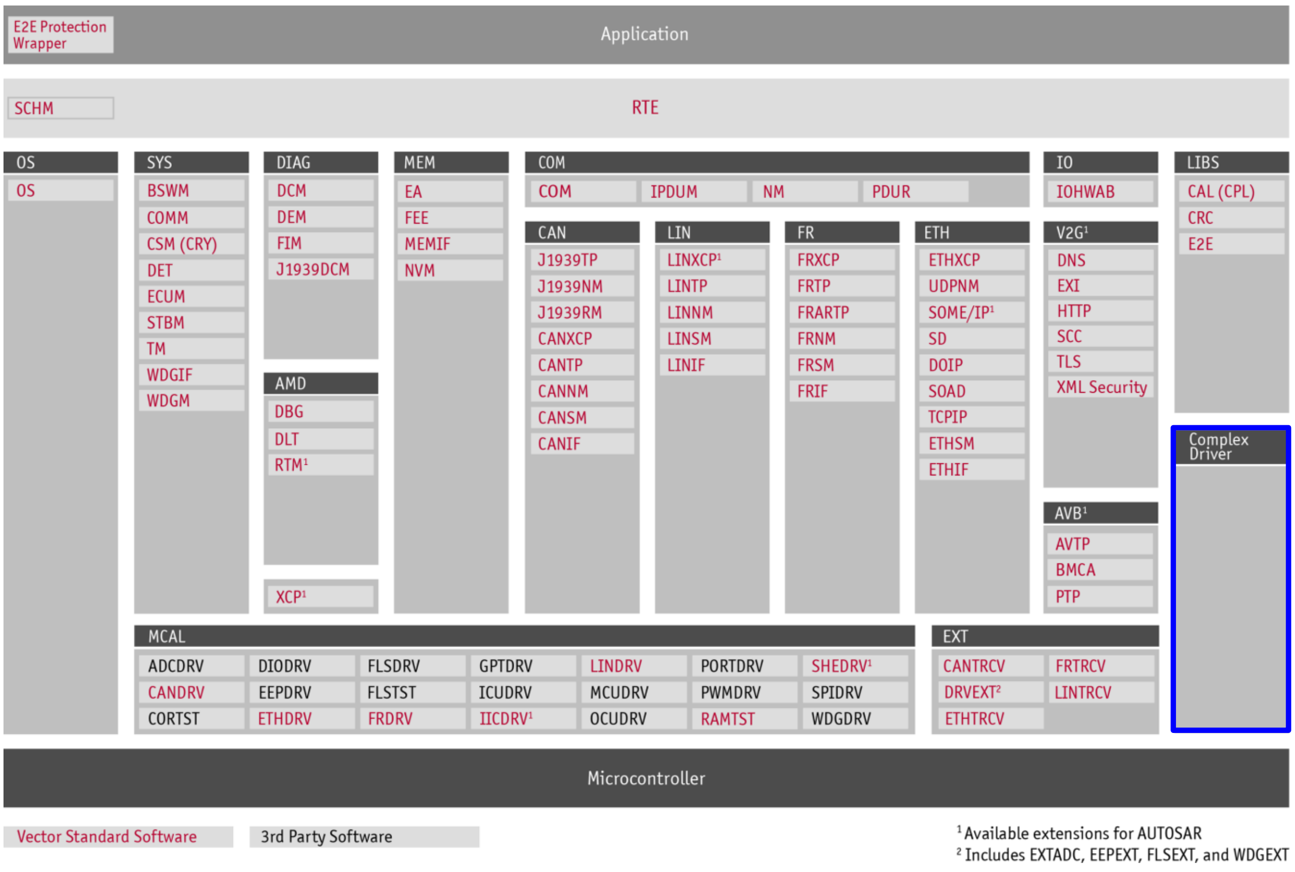

Figure 2-1 AUTOSAR 4.1 Architecture Overview

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for the <CDD>.

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

For further information of not supported features see also chapter 13.

Vector Informatik provides further <CDD> functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
CddComIfUpperLayerContribution
CddPduRLowerLayerContribution
CddPduRUpperLayerContribution
CddSoAdUpperLayerContribution

Table 3-1 Supported AUTOSAR standard conform features

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

Not Supported AUTOSAR Standard Conform Features
CddEcucPartitionInteraction
CddComMLowerLayerContribution
CddGenericNmLowerLayerContribution

Table 3-2 Not supported AUTOSAR standard conform features

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
CddJ1939RmContribution

Table 3-3 Features provided beyond the AUTOSAR standard

4 Integration

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

Note
The MSN is derived from the short name of the module configuration and not from the API-SERVICE-PREFIX of the VSMD file.

4.1 Scope of Delivery

The delivery of the <CDD> contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

The <CDD> implementation has no static files.

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool CFG5.

File Name	Description
Cdd_Cbk.h	This file is generated if the <CDD> is configured as a CddPduRUpperLayerContribution or CddComIfUpperLayerContribution.
Cdd.h	This file is generated if the <CDD> is configured as a CddPduRLowerLayerContribution

Table 4-1 Generated 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.

Edit
Update for each <CDD> instance the templates _MemMap.h and _Compiler_Cfg.h and replace “_CDD” with your <CDD> name.

5 API Description CddPduRUpperLayerContribution as IF

This chapter describes APIs to be implemented by the <CDD> if the <CDD> is configured as upper layer communication interface for the PduR.

5.1 Services used by <CDD>

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

Component	API
PduR	PduR_<CDD>Transmit

Table 5-1 Services used by the <CDD>

5.2 Callback Functions

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

5.2.1 <CDD>_RxIndication

Prototype	
`void` **`<CDD>_RxIndication`** `(PduIdType RxPduId, PduInfoType* PduInfoPtr)`	
Parameter	
`RxPduId`	id of the CddPduRUpperLayerRxPdu.
`PduInfoPtr`	Payload information of the received I-PDU (pointer to data and data length).
Return code	
`void`	
Functional Description	
The function is called to indicate the complete reception of a RX I-PDU.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_RxIndication call for the same RxPduId.	

Table 5-2 <CDD>_RxIndication

5.2.2 <CDD>_TxConfirmation

Prototype	
`void <CDD>_TxConfirmation (PduIdType TxPduId)`	
Parameter	
`TxPduId`	id of the CddPduRUpperLayerTxPdu.
Return code	
`void`	
Functional Description	
The function is called to confirm the transmission of an I-PDU.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_TxConfirmation call for the same TxPduId.	

Table 5-3 <CDD>_TxConfirmation

5.2.3 <CDD>_TriggerTransmit

Prototype	
`Std_ReturnType <CDD>_TriggerTransmit (PduIdType TxPduId, PduInfoType PduInfoPtr)`	
Parameter	
`TxPduId`	id of the CddPduRUpperLayerTxPdu.
`PduInfoPtr`	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Return code	
`Std_ReturnType`	E_OK SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK No data has been copied, because Cdd is not initialized or TxPduId is not valid or PduInfoPtr is NULL_PTR or SduDataPtr is NULL_PTR or SduLength is too small.
Functional Description	
The function is called to request the I-PDU for transmission.	

Particularities and Limitations
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.
Expected Caller Context
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_TriggerTransmit call for the same TxPduId.

Table 5-4 <CDD>_TriggerTransmit

6 API Description CddPduRUpperLayerContribution as TP

This chapter describes APIs to be implemented by the <CDD> if the <CDD> is configured as upper layer transport protocol for the PduR.

6.1 Services used by <CDD>

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

Component	API
PduR	PduR_<CDD>Transmit
PduR	PduR_<CDD>ChangeParameter
PduR	PduR_<CDD>CancelReceive

Table 6-1 Services used by the <CDD>

6.2 Callback Functions

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

6.2.1 <CDD>_StartOfReception

Prototype	
`BufReq_ReturnType` **`<CDD>_StartOfReception`** `(PduIdType id, PduInfoType* info, PduLengthType TpSduLength, PduLengthType* bufferSizePtr)`	
Parameter	
`id`	id of the CddPduRUpperLayerRxPdu.
`info`	Pointer to a PduInfoType structure containing the payload data (without protocol information) and payload length of the first frame or single frame of a transport protocol I-PDU reception. Depending on the global parameter MetaDataLength, additional bytes containing MetaData (e.g. the CAN ID) are appended after the payload data.
`TpSduLength`	Length of the entire TP SDU which will be received.
`bufferSizePtr`	Length of the available receive buffer in the <CDD>. This parameter is used e.g. in CanTp to calculate the Block Size (BS).
Return code	
`BufReq_ReturnType`	a BufReq_ReturnType constant of ComStackTypes.h.
Functional Description	
The function call indicates the reception start of a segmented PDU.	

Particularities and Limitations
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.
Expected Caller Context
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_StartOfReception call for the same id.

Table 6-2 <CDD>_StartOfReception

6.2.2 <CDD>_CopyRxData

Prototype	
`BufReq_ReturnType` **`<CDD>_CopyRxData`** `(PduIdType id, PduInfoType* info, PduLengthType* bufferSizePtr)`	
Parameter	
`id`	id of the CddPduRUpperLayerRxPdu.
`info`	a PduInfoType pointing to the data to be copied in the <CDD> data buffer.
`bufferSizePtr`	available receive buffer after data has been copied.
Return code	
`BufReq_ReturnType`	a BufReq_ReturnType constant of ComStackTypes.h.
Functional Description	
This function is called to trigger the copy process of a segmented PDU. The function can be called several times and each call to this function copies parts of the received data.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_CopyRxData call for the same id.	

Table 6-3 <CDD>_CopyRxData

6.2.3 <CDD>_TpRxIndication

Prototype	
`void` **`<CDD>_TpRxIndication`** `(PduIdType id, Std_ReturnType result)`	
Parameter	
`id`	id of the CddPduRUpperLayerRxPdu.
`result`	a Std_ReturnType to indicate the result of the reception.
Return code	
`void`	
Functional Description	
The function is called to indicate the complete reception of a <CDD> TP SDU or to report an error that occurred during reception.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_TpRxIndication call for the same id.	

Table 6-4 <CDD>_TpRxIndication

6.2.4 <CDD>_CopyTxData

Prototype	
`void` **`<CDD>_CopyTxData`** `(PduIdType id, PduInfoType* info, RetryInfoType retry, PduLengthType* availableDataPtr)`	
Parameter	
`id`	id of the CddPduRUpperLayerTxPdu.
`info`	a PduInfoType pointing to the destination buffer.
`retry`	NULL_PTR to indicate a successful copy process or a RetryInfoType containing a TpDataStateType constant of ComStackTypes.h.
`availableDataPtr`	Indicates the remaining number of bytes that are available in the TX buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. Iso FrTp) to determine the size of the following CFs.
Return code	
`BufReq_ReturnType`	a BufReq_ReturnType constant of ComStackTypes.h.
Functional Description	
This function is called to request transmit data of a TP CddPduRUpperLayerTxPdu. The function can be called several times and each call to this function copies the next part of the data to be transmitted.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_CopyTxData call for the same id.	

Table 6-5 <CDD>_CopyTxData

6.2.5 <CDD>_TpTxConfirmation

Prototype	
`void` **`<CDD>_TpTxConfirmation`** `(PduIdType id, Std_ReturnType result)`	
Parameter	
`id`	id of the CddPduRUpperLayerTxPdu.
`result`	a Std_ReturnType to indicate the result of the transmission.
Return code	
`BufReq_ReturnType`	a BufReq_ReturnType constant of ComStackTypes.h.
Functional Description	
The function is called to confirm a successful transmission of a TP CddPduRUpperLayerTxPdu or to report an error that occurred during transmission.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_TpTxConfirmation call for the same id.	

Table 6-6 <CDD>_TpTxConfirmation

7 API Description CddPduRLowerLayerContribution as IF

This chapter describes APIs to be implemented by the <CDD> if the <CDD> is configured as lower layer communication interface for the PduR.

7.1 Services provided by <CDD>

7.1.1 <CDD>_Transmit

Prototype		
`Std_ReturnType` **`<CDD>_Transmit`** `(PduIdType TxPduId, PduInfoType* PduInfoPtr)`		
Parameter		
`TxPduId`	id of the IF CddPduRLowerLayerTxPdu.	
`PduInfoPtr`	a PduInfoType pointing to the transmit buffer.	
Return code		
`Std_ReturnType`	E_OK	the transmission request has been accepted.
	E_NOT_OK	the transmission request has NOT been accepted.
Functional Description		
The function is called to initiate a transmission request of a TX I-PDU.		
Particularities and Limitations		
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.		
Expected Caller Context		
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_Transmit call for the same TxPduId.		

Table 7-1 <CDD>_Transmit

7.1.2 <CDD>_CancelTransmit

Prototype	
`Std_ReturnType` **`<CDD>_CancelTransmit`** `(PduIdType TxPduId)`	
Parameter	
`TxPduId`	id of the IF CddPduRLowerLayerTxPdu.
Return code	
`Std_ReturnType`	E_OK the transmission cancellation has been processed successful. E_NOT_OK the transmission cancellation has NOT been processed successful.
Functional Description	
The function is called to cancel a transmission request of a TX I-PDU.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_CancelTransmit call for the same TxPduId.	

Table 7-2 <CDD>_CancelTransmit

7.2 Services used by <CDD>

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

Component	API
PduR	PduR_<CDD>RxIndication
PduR	PduR_<CDD>TxConfirmation
PduR	PduR_<CDD>TriggerTransmit

Table 7-3 Services used by the <CDD>

8 API Description CddPduRLowerLayerContribution as TP

This chapter describes APIs to be implemented by the <CDD> if the <CDD> is configured as lower layer transport protocol for the PduR.

8.1 Services provided by <CDD>

8.1.1 <CDD>_Transmit

Prototype	
`Std_ReturnType` **`<CDD>_Transmit`** `(PduIdType TxPduId, PduInfoType* PduInfoPtr)`	
Parameter	
`TxPduId`	id of the IF CddPduRLowerLayerTxPdu.
`PduInfoPtr`	a PduInfoType pointing to the transmit buffer.
Return code	
`Std_ReturnType`	E_OK the transmission request has been accepted. E_NOT_OK the transmission request has NOT been accepted.
Functional Description	
The function is called to initiate a transmission request of a TX I-PDU.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_Transmit call for the same TxPduId.	

Table 8-1 <CDD>_Transmit

8.1.2 <CDD>_CancelTransmit

Prototype	
`Std_ReturnType` **`<CDD>_CancelTransmit`** `(PduIdType id)`	
Parameter	
`id`	id of the IF CddPduRLowerLayerTxPdu.
Return code	
`Std_ReturnType`	E_OK the transmission cancellation has been processed successful. E_NOT_OK the transmission cancellation has NOT been processed successful.
Functional Description	
The function is called to cancel a transmission request of a TX I-PDU.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_CancelTransmit call for the same id.	

Table 8-2 <CDD>_CancelTransmit

8.1.3 <CDD>_CancelReceive

Prototype		
`Std_ReturnType` **`<CDD>_CancelReceive`** `(PduIdType id)`		
Parameter		
`id`	id of the TP CddPduRLowerLayerRxPdu.	
Return code		
`Std_ReturnType`	E_OK	the reception cancellation has been processed successful.
	E_NOT_OK	the reception cancellation has NOT been processed successful.
Functional Description		
The function is called to cancel a reception of a RX I-PDU.		
Particularities and Limitations		
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.		
Expected Caller Context		
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_CancelReceive call for the same id.		

Table 8-3 <CDD>_CancelReceive

8.1.4 <CDD>_ChangeParameter

Prototype	
`Std_ReturnType` **`<CDD>_ChangeParameter`** `(PduIdType id, TPParameterType parameter, uint16 value)`	
Parameter	
`id`	id of the TP CddPduRLowerLayerRxPdu.
`parameter`	a TPParameterType enumeration of ComStackTypes.h.
`value`	the new value of the parameter
Return code	
`Std_ReturnType`	E_OK the parameter change has been processed successful. E_NOT_OK the parameter change has NOT been processed successful.
Functional Description	
The function is called to change a transport protocol parameter (e.g. block size) of a RX I-PDU	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_CancelParameter call for the same id.	

Table 8-4 <CDD>_ChangeParameter

8.2 Services used by <CDD>

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

Component	API
PduR	PduR_<CDD>StartOfReception
PduR	PduR_<CDD>CopyRxData
PduR	PduR_<CDD>RxIndication
PduR	PduR_<CDD>CopyTxData
PduR	PduR_<CDD>TxConfirmation

Table 8-5 Services used by the <CDD>

9 API Description CddComIfUpperLayerContribution

This chapter describes APIs to be implemented by the <CDD> if the <CDD> is configured as upper layer communication interface for a communication hardware abstraction layer.

9.1 Services used by <CDD>

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

Component	API
CanIf, LinIf, FrIf	CanIf_Transmit, LinIf_Transmit, FrIf_Transmit

Table 9-1 Services used by the <CDD>

9.2 Callback Functions

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

Note
The names of the callbacks can be defined completely in the communication hardware abstraction layer. The postfix _RxIndication, _TxConfirmation and _TriggerTransmit is used in this chapter to eplain the usage of the function.

9.2.1 <CDD>_RxIndication

Prototype	
`void` **`<CDD>_RxIndication`** `(PduIdType RxPduId, PduInfoType* PduInfoPtr)`	
Parameter	
`RxPduId`	id of the CddComIfUpperLayerRxPdu.
`PduInfoPtr`	Payload information of the received I-PDU (pointer to data and data length).
Return code	
`void`	
Functional Description	
The function is called to indicate the complete reception of a RX I-PDU.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by a communication hardware abstraction layer.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_RxIndication call for the same RxPduId.	

Table 9-2 <CDD>_RxIndication

9.2.2 <CDD>_TxConfirmation

Prototype	
`void` **`<CDD>_TxConfirmation`** `(PduIdType TxPduId)`	
Parameter	
`TxPduId`	id of the CddComIfUpperLayerTxPdu.
Return code	
`void`	
Functional Description	
The function is called to confirm the transmission of an I-PDU.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by a communication hardware abstraction layer.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_TxConfirmation call for the same TxPduId.	

Table 9-3 <CDD>_RxIndication

9.2.3 <CDD>_TriggerTransmit

Prototype	
`Std_ReturnType` **`<CDD>_TriggerTransmit`** `(PduIdType TxPduId, PduInfoType PduInfoPtr)`	
Parameter	
`TxPduId`	id of the CddComIfUpperLayerTxPdu.
`PduInfoPtr`	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Return code	
`Std_ReturnType`	E_OK SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK No data has been copied, because Cdd is not initialized or TxPduId is not valid or PduInfoPtr is NULL_PTR or SduDataPtr is NULL_PTR or SduLength is too small.
Functional Description	
The function is called to request the I-PDU for transmission.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the PduR.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_TriggerTransmit call for the same TxPduId.	

Table 9-4 <CDD>_TriggerTransmit

10 API Description CddJ1939RmContribution

This chapter describes APIs to be implemented by the <CDD> if the <CDD> is configured as upper layer of the J1939Rm module.

10.1 Services used by <CDD>

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

Component	API

Table 10-1 Services used by the <CDD>

10.2 Callback Functions

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

10.2.1 <CDD>_RequestIndication

Prototype	
`void` **`<CDD>_RequestIndication`** `(uint8 node, NetworkHandleType channel, uint32 requestedPgn, uint8 sourceAddress, uint8 destAddress, uint8 priority)`	
Parameter	
`node`	Node by which the request was received.
`channel`	Channel on which the request was received.
`requestedPgn`	PGN of the requested PG.
`sourceAddress`	Address of the node that sent the Request PG.
`destAddress`	Address of this node or 0xFF for broadcast.
`priority`	Priority of the Request PG.
Return code	
`void`	
Functional Description	
The RequestIndication provides information about a received J1939 RQST message which affects a J1939Rm user that references this CDD.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_RequestIndication call for the same node.	

Table 10-2 <CDD>_RequestIndication

10.2.2 <CDD>_AckIndication

Prototype	
`void` **`<CDD>_AckIndication`** `(uint8 node, NetworkHandleType channel, uint32 ackPgn, uint8 ackCode, uint8 ackAddress, uint8 sourceAddress, uint8 priority)`	
Parameter	
`node`	Node by which the acknowledgement was received.
`channel`	Channel on which the acknowledgement was received.
`ackPgn`	Acknowledged PGN.
`ackCode`	Type of acknowledgement, see definition of J1939Rm_AckCode for available codes.
`ackAddress`	Address of this node.
`sourceAddress`	Address of the node that sent the Acknowledgement PG.
`priority`	Priority of the Acknowledgement PG.
Return code	
`void`	

Functional Description
The AckIndication provides information about a received J1939 ACKM message which affects a J1939Rm user that references this CDD.
Particularities and Limitations
> Service ID: N.a. > The **<CDD>** is initialized and active.
Expected Caller Context
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_AckIndication call for the same node.

Table 10-3 <CDD>_AckIndication

10.2.3 <CDD>_RequestTimeoutIndication

Prototype	
`void` **`<CDD>_RequestTimeoutIndication`** `(uint8 node, NetworkHandleType channel, uint32 requestedPgn, uint8 destAddress)`	
Parameter	
`node`	Node by which the request was sent.
`channel`	Channel on which the request was sent.
`requestedPgn`	PGN of the requested PG.
`destAddress`	Address of the destination node or 0xFF for broadcast.
Return code	
`void`	
Functional Description	
The RequestTimeoutIndication is called after time out of a transmitted J1939 RQST message.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_RequestTimeoutIndication call for the same node.	

Table 10-4 <CDD>_RequestTimeoutIndication

11 API Description CddSoAdUpperLayerContribution

This chapter describes APIs to be implemented by the <CDD> if the <CDD> is configured as upper layer interface in the SoAd.

Note
The caller and type infixes of the APIs are configurable in the SoAdBswModules.

11.1 Services used by <CDD>

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

Component	API
SoAd	SoAd_IfTransmit
SoAd	SoAd_TpTransmit
SoAd	SoAd_TpChangeParameter
SoAd	SoAd_TpCancelReceive
SoAd	SoAd_TpCancelTransmit

Table 11-1 Services used by the <CDD>

11.2 Communication Interface Callback Functions

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

11.2.1 <CDD>_[SoAd][If]RxIndication

Prototype	
`void` **`<CDD>_[SoAd][If]RxIndication`** `(PduIdType RxPduId, PduInfoType* PduInfoPtr)`	
Parameter	
`RxPduId`	id of the CddSoAdUpperLayerRxPdu.
`PduInfoPtr`	Payload information of the received I-PDU (pointer to data and data length).
Return code	
`void`	
Functional Description	
The function is called to indicate the complete reception of a RX I-PDU.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the SoAD.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_[SoAd][If]RxIndication call for the same RxPduId.	

Table 11-2 <CDD>_[SoAd][If]RxIndication

11.2.2 <CDD>_[SoAd][If]TxConfirmation

Prototype	
`void` **`<CDD>_[SoAd][If]TxConfirmation`** `(PduIdType TxPduId)`	
Parameter	
`TxPduId`	id of the CddSoAdUpperLayerTxPdu.
Return code	
`void`	
Functional Description	
The function is called to confirm the transmission of an I-PDU.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the SoAd.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_[SoAd][If]TxConfirmation call for the same TxPduId.	

Table 11-3 <CDD>_[SoAd][If]TxConfirmation

11.2.3 <CDD>_[SoAd][If]TriggerTransmit

Prototype	
`Std_ReturnType` **`<CDD>_[SoAd][If]TriggerTransmit`** `(PduIdType TxPduId, PduInfoType PduInfoPtr)`	
Parameter	
`TxPduId`	id of the CddSoAdUpperLayerTxPdu.
`PduInfoPtr`	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Return code	
`Std_ReturnType`	E_OK SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK No data has been copied, because Cdd is not initialized or TxPduId is not valid or PduInfoPtr is NULL_PTR or SduDataPtr is NULL_PTR or SduLength is too small.
Functional Description	
The function is called to request the I-PDU for transmission.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the SoAd.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_[SoAd][If]TriggerTransmit call for the same TxPduId.	

Table 11-4 <CDD>_[SoAd][If]TriggerTransmit

11.3 Transport Protocol Callback Functions

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

11.3.1 <CDD>_[SoAd][Tp]StartOfReception

Prototype	
`BufReq_ReturnType` **`<CDD>_[SoAd][Tp]StartOfReception`** `(PduIdType id, PduInfoType* info, PduLengthType TpSduLength, PduLengthType* bufferSizePtr)`	
Parameter	
`id`	id of the CddSoAdUpperLayerRxPdu.
`info`	Pointer to a PduInfoType structure containing the payload data (without protocol information) and payload length of the first frame or single frame of a transport protocol I-PDU reception. Depending on the global parameter MetaDataLength, additional bytes containing MetaData (e.g. the CAN ID) are appended after the payload data.
`TpSduLength`	Length of the entire TP SDU which will be received.
`bufferSizePtr`	Length of the available receive buffer in the <CDD>. This parameter is used e.g. in CanTp to calculate the Block Size (BS).
Return code	
`BufReq_ReturnType`	a BufReq_ReturnType constant of ComStackTypes.h.
Functional Description	
The function call indicates the reception start of a segmented PDU.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the SoAd.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_[SoAd][Tp]StartOfReception call for the same id.	

Table 11-5 <CDD>_[SoAd][Tp]StartOfReception

11.3.2 <CDD>_[SoAd][Tp]CopyRxData

Prototype	
`BufReq_ReturnType` **`<CDD>_[SoAd][Tp]CopyRxData`** `(PduIdType id, PduInfoType* info, PduLengthType* bufferSizePtr)`	
Parameter	
`id`	id of the CddSoAdUpperLayerRxPdu.
`info`	a PduInfoType pointing to the data to be copied in the <CDD> data buffer.
`bufferSizePtr`	available receive buffer after data has been copied.
Return code	
`BufReq_ReturnType`	a BufReq_ReturnType constant of ComStackTypes.h.
Functional Description	
This function is called to trigger the copy process of a segmented PDU. The function can be called several times and each call to this function copies parts of the received data.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the SoAd.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_[SoAd][Tp]CopyRxData call for the same id.	

Table 11-6 <CDD>_[SoAd][Tp]CopyRxData

11.3.3 <CDD>_[SoAd][Tp]RxIndication

Prototype	
`void <CDD>_[SoAd][Tp]RxIndication (PduIdType id, Std_ReturnType result)`	
Parameter	
`id`	id of the CddSoAdUpperLayerRxPdu.
`result`	a Std_ReturnType to indicate the result of the reception.
Return code	
`void`	
Functional Description	
The function is called to indicate the complete reception of a <CDD> TP SDU or to report an error that occurred during reception.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the SoAd.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_[SoAd][Tp]RxIndication call for the same id.	

Table 11-7 <CDD>_[SoAd][Tp]RxIndication

11.3.4 <CDD>_[SoAd][Tp]CopyTxData

Prototype	
`void` **`<CDD>_[SoAd][Tp]CopyTxData`** `(PduIdType id, PduInfoType* info, RetryInfoType retry, PduLengthType* availableDataPtr)`	
Parameter	
`id`	id of the CddSoAdUpperLayerTxPdu.
`info`	a PduInfoType pointing to the destination buffer.
`retry`	NULL_PTR to indicate a successful copy process or a RetryInfoType containing a TpDataStateType constant of ComStackTypes.h.
`availableDataPtr`	Indicates the remaining number of bytes that are available in the TX buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. Iso FrTp) to determine the size of the following CFs.
Return code	
`BufReq_ReturnType`	a BufReq_ReturnType constant of ComStackTypes.h.
Functional Description	
This function is called to request transmit data of a TP CddSoAdUpperLayerTxPdu. The function can be called several times and each call to this function copies the next part of the data to be transmitted.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the SoAd.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_[SoAd][Tp]CopyTxData call for the same id.	

Table 11-8 <CDD>_[SoAd][Tp]CopyTxData

11.3.5 <CDD>_[SoAd][Tp]TxConfirmation

Prototype	
`void <CDD>_[SoAd][Tp]TxConfirmation (PduIdType id, Std_ReturnType result)`	
Parameter	
`id`	id of the CddSoAdUpperLayerTxPdu.
`result`	a Std_ReturnType to indicate the result of the transmission.
Return code	
`BufReq_ReturnType`	a BufReq_ReturnType constant of ComStackTypes.h.
Functional Description	
The function is called to confirm a successful transmission of a TP CddSoAdUpperLayerTxPdu or to report an error that occurred during transmission.	
Particularities and Limitations	
> Service ID: N.a. > The **<CDD>** is initialized and active. > The function is called by the SoAd.	
Expected Caller Context	
> The function can be called in interrupt and on task level and should not be interrupted by another <CDD>_[SoAd][Tp]TxConfirmation call for the same id.	

Table 11-9 <CDD>_[SoAd][Tp]TxConfirmation

12 Configuration

12.1 Configuration Variants

The <CDD> supports the configuration variants

> `VARIANT-PRE-COMPILE`

The configuration classes of the <CDD> parameters depend on the supported configuration variants. For their definitions please see the [BSW_specific]_bswmd.arxml file.

13 AUTOSAR Standard Compliance

13.1 Deviations

No deviations.

13.2 Additions/ Extensions

See Table 3-3 Features provided beyond the AUTOSAR standard.

13.3 Limitations

See Table 3-2 Not supported AUTOSAR standard conform features.

14 Glossary and Abbreviations

14.1 Glossary

Term	Description
BSWMD	The BSWMD is a formal notation of all information belonging to a certain BSW artifact (BSW module or BSW cluster) in addition to the implementation of that artifact.
Buffer	A buffer in a memory area normally in the RAM. It is an area, that the application has reserved for data storage.
CANbedded	… is the trademark of the Vector communication stack.
CFG5	Generation tool for CANbedded and MICROSAR components
Component	CAN Driver, Network Management ... are software COMPONENTS in contrast to the expression module, which describes an ECU.
Confirmation	A service primitive defined in the ISO/OSI Reference Model (ISO 7498). With the service primitive 'confirmation' a service provider informs a service user about the result of a preceding service request of the service user. Notification by the CAN Driver on asynchronous successful transmission of a CAN message.
Electronic Control Unit	Also known as ECU. Small embedded computer system consisting of at least one CPU and corresponding periphery which is placed in one housing.
Indication	A service primitive defined in the ISO/OSI Reference Model (ISO 7498). With the service primitive 'indication' a service provider informs a service user about the occurrence of either an internal event or a service request issued by another service user. Notification of application in case of events in the Vector software components, e.g. an asynchronous reception of a CAN message.
Interrupt	Processor-specific event which can interrupt the execution of a current program section.
Interrupt service routine	The function used for direct processing of an interrupt.
Network	A network defines the assignment (1:N) between a logical communication grouping and a physical layer on which different modules are connected to (either CAN or LIN). One network consists of one channel, Y networks consists of 1..Z channel(s). We say network if we talk about more than one bus.
Transport Protocol	Some information that must be transferred over the CAN/LIN bus does not fit into individual message frames because the data length exceeds the maximum of 8 bytes. In this case, the sender must divide up the data into a number of messages. Additional information is necessary for the receiver to put the data together again.

Table 14-1 Glossary

14.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
CAN	Controller Area Network protocol originally defined for use as a communication network for control applications in vehicles.
CANIF	Controller Area Network Interface
CDD	Complex Device Driver, Complex Drivers
COM	Communication
ECU	Electronic Control Unit
FRIF	Flexray Interface
HIS	Hersteller Initiative Software
ID	Identifier (e.g. Identifier of a CAN message)
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
MSN	Module Short Name
PDU	Protocol Data Unit
SDU	Service Data Unit
SWC	Software Component
SWS	Software Specification
TP	Transport Protocol
VSMD	Vendor Specific Module Description

Table 14-2 Abbreviations

15 Contact

Visit our website for more information on

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

www.vector.com