MICROSAR Classic Efficient COM for Large Data

Technical Reference

Version 3.01.00

Authors	visswa, visfrm
Status	Released

Document Information

History

Author	Date	Version	Remarks
visswa	2014-05-13	1.00.00	ESCAN00074216: AR4-665: Efficient COM for Large Data ESCAN00074486: AR4-619: Support the StartOfReception API (with the PduInfoType), TxConfirmation and RxIndication according ASR4.1.2
visfrm	2016-01-28	1.01.00	ESCAN00087873: FEAT-1631: Trigger Transmit API with SduLength In/Out according to ASR4.2.2
visfrm, visswa	2016-06-08	2.00.00	ESCAN00087732: FEAT-780: Release of FEAT-698 (Large Data COM) [AR4-987]
visfrm	2021-03-09	3.00.00	COM-2009: Remove the AUTHOR IDENTITY COM-1957: Update Doc_TechRef with the new template COM-1853: Implement Multipartition LdCom
Anna Lefarth	2022-03-01		Product name updated to MICROSAR Classic.

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	Specification of Module Efficient COM for Large Data	R4.2.2
[2]	AUTOSAR	Specification of Default Error Tracer	R4.2.2

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 Introduction

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

Supported Configuration Variants:	pre-compile, post-build-loadable, post-build-selectable	
Vendor ID:	LDCOM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	LDCOM_MODULE_ID	49 decimal

The AUTOSAR LdCom module provides an alternative Interaction Layer mechanism besides AUTOSAR Com. By focusing on spontaneous, non-cyclic communication without serializing, filtering and conversion an efficient implementation of the module without local buffers is achieved.

Main features:

- Provision of signal oriented data interface for the Rte
- Provision of received signals to Rte
- Support of large and dynamic length data types
- Support of IF- and TP-based communication
- Provision of PDU oriented data interface towards PduR

1.1 Architecture Overview

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

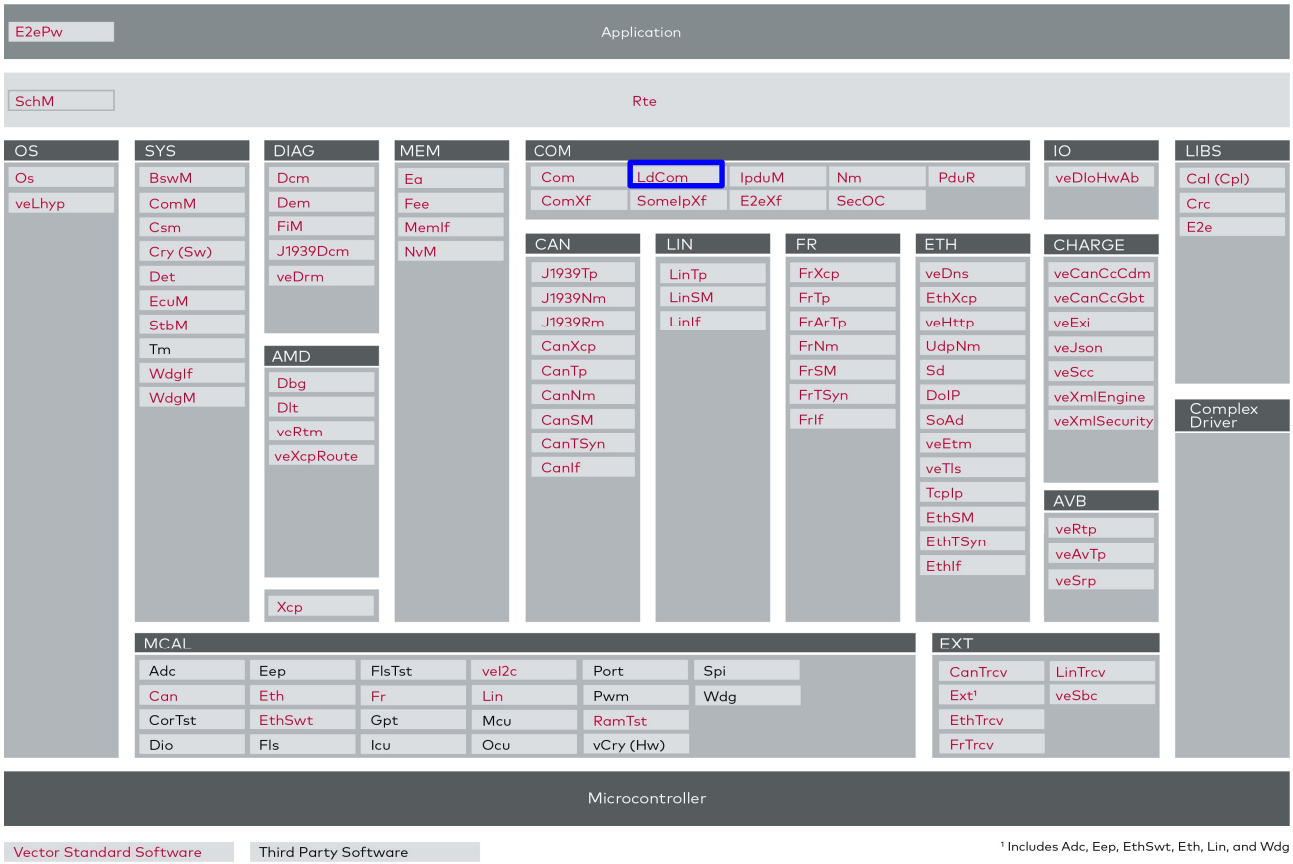

Figure 1-1 AUTOSAR 4.x Architecture Overview

2 Functional Description

2.1 Features

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

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

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

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Communication interface transmission / reception
Transport protocol transmission / reception

Table 2-1 Supported AUTOSAR standard conform features

2.1.1 Communication interface transmission / reception

For transmission of communication interface I-Pdus the upper layer (Rte) uses the API `LdCom_Transmit()`. Based on the provided I-Pdu ID the transmission request and the payload information are forwarded to the PduR module.

The final PduR Tx confirmation callback `LdCom_TxConfirmation()` gets transformed into an I-Pdu/Signal specific callback function configured by the upper layer. The same transformation is performed for triggerable I-Pdus using `LdCom_TriggerTransmit()`.

On reception of I-Pdus the related PduR Rx indication callback `LdCom_RxIndication()` gets transformed into an I-Pdu/Signal specific callback function configured by the upper layer.

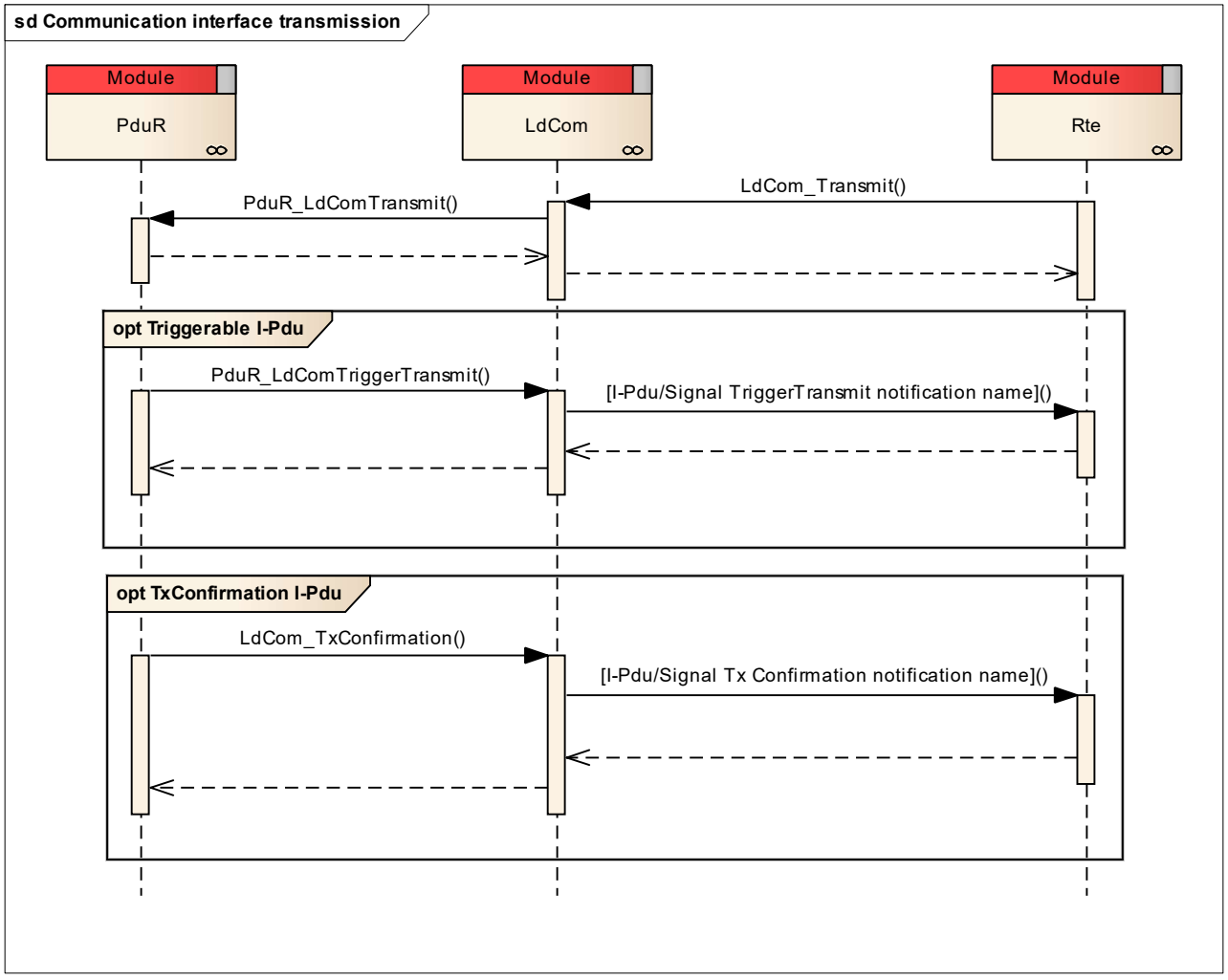

Figure 2-1 Communication interface I-Pdu transmission

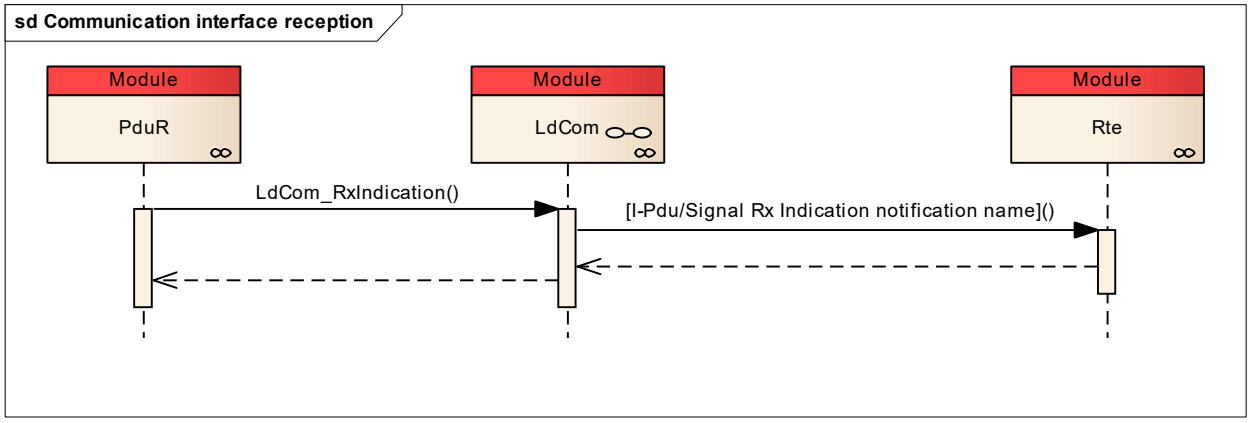

Figure 2-2 Communication interface I-Pdu reception

2.1.2 Transport protocol transmission / reception

For transmission of transport protocol I-Pdus the upper layer (Rte) uses the API `LdCom_Transmit()`. Based on the provided I-Pdu ID the transmission request is forwarded to the PduR module.

Each TP-segment specific `LdCom_CopyTxData()` callback gets transformed into an I-Pdu/Signal specific callback function configured by the upper layer. The same API transformation is performed for the final Tx confirmation `LdCom_TpTxConfirmation()`.

On reception side the same API transformation is performed for the callbacks `LdCom_StartOfReception()`, `LdCom_CopyRxData()` and `LdCom_TpRxIndication()`.

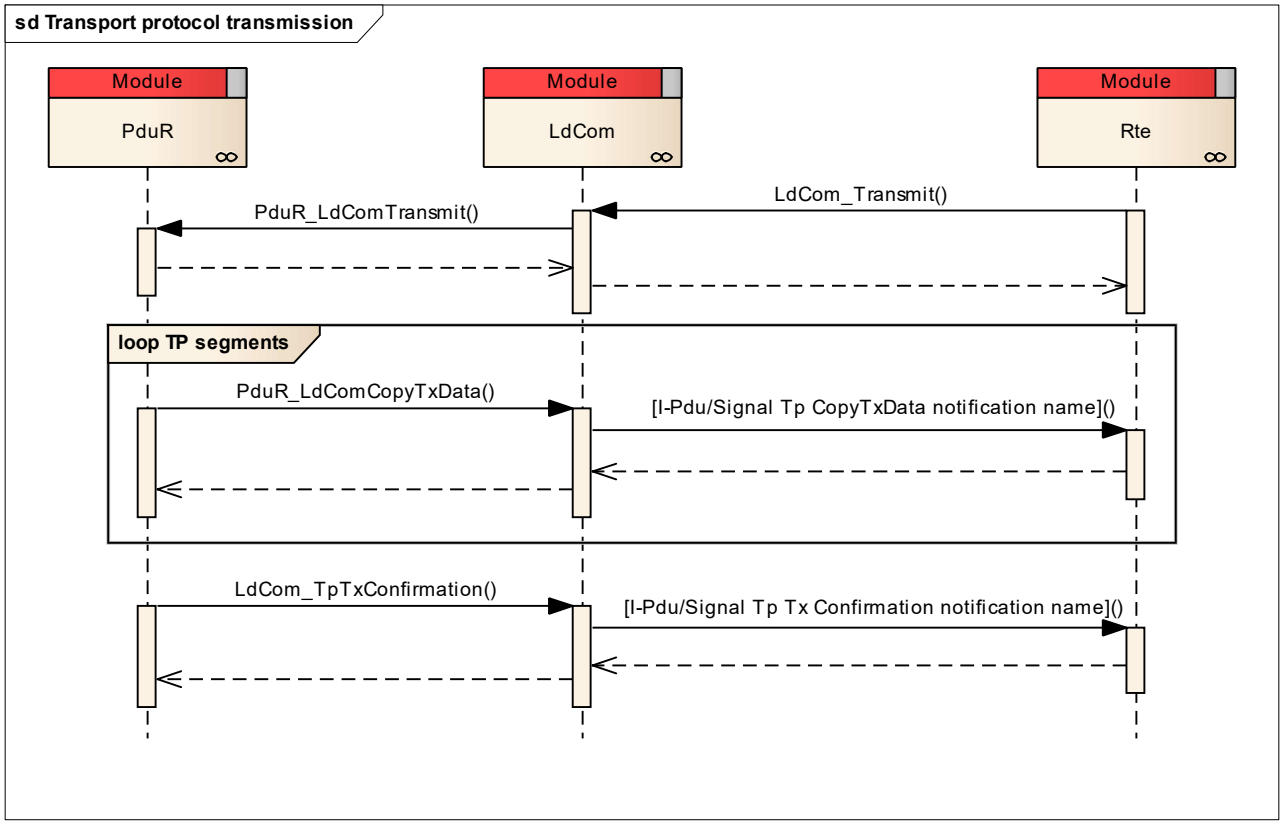

Figure 2-3 Transport protocol I-Pdu transmission

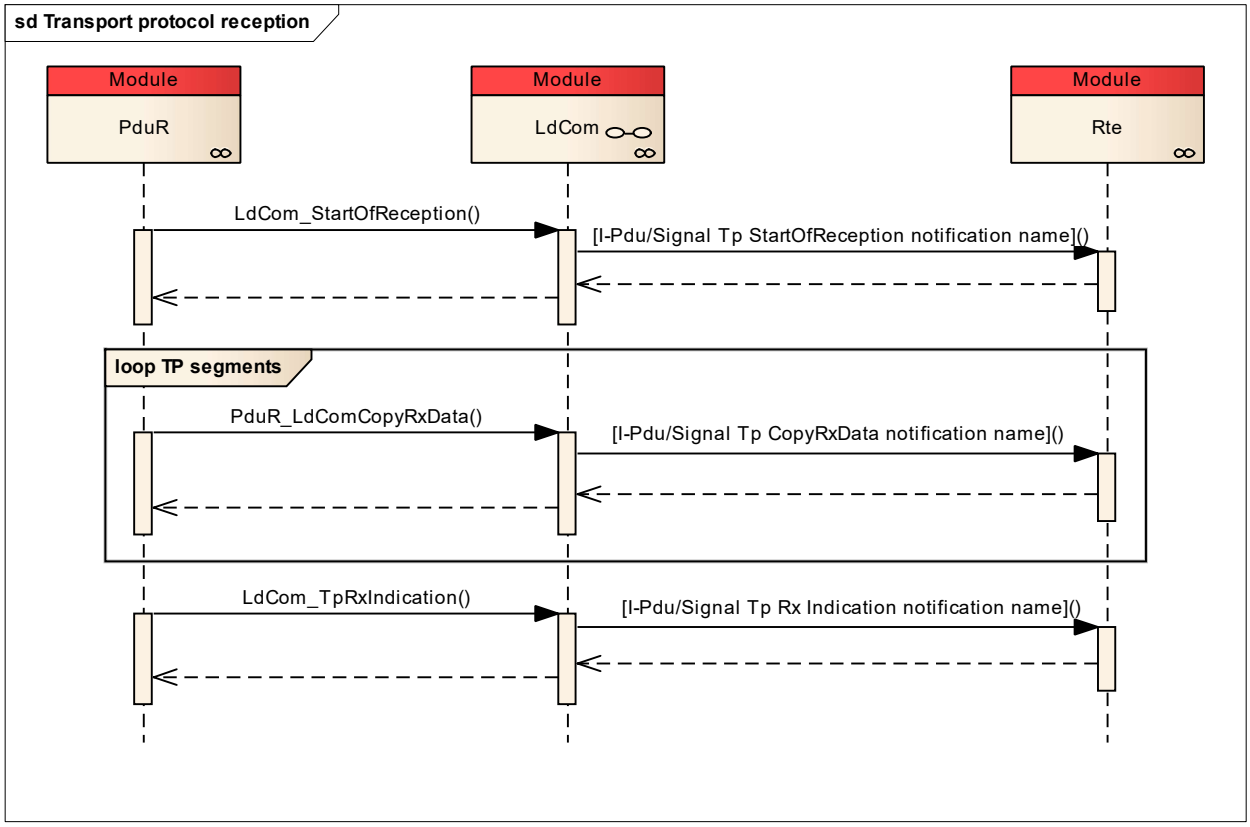

Figure 2-4 Transport protocol I-Pdu reception

2.1.3 Multipartition support

The LdCom can be used in a multi-partition/core system. As the LdCom only forwards API calls statically, the caller must ensure that it calls the LdCom in the correct partition/core context.

For the correct initialization in such a system, see chapter 2.2.

2.1.4 Deviations

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

Not Supported AUTOSAR Standard Conform Features
-

Table 2-2 Not supported AUTOSAR standard conform features

2.2 Initialization

If not already done by the startup code, the statically initialized RAM variables must be initialized by calling `LdCom_InitMemory()`.

The LdCom itself is initialized by calling `LdCom_Init()`. The module can be reset with `LdCom_DeInit()`. In the uninitialized state the only available API function is `LdCom_GetVersionInfo()`.

In a multi-partition/core system these initialization APIs must be called once on a “master” partition. After this initialization, the LdCom can be used on all partitions.

2.3 States

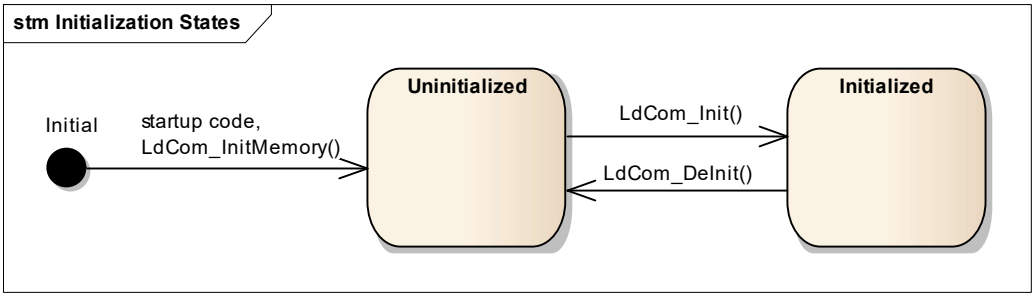

Figure 2-5 LdCom states

2.4 Main Functions

All functionalities of the LdCom are asynchronous. All functionality is performed within the requested operation. Therefore no main function exists.

2.5 Error Handling

2.5.1 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 `LDCOM_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 LdCom ID is 49.

The reported service IDs identify the services which are described in 4.1. The following table presents the service IDs and the related services:

Service ID	Service
0x01	LdCom_Init
0x02	LdCom_DeInit
0x03	LdCom_GetVersionInfo
0x05	LdCom_Transmit
0x43	LdCom_CopyTxData

Service ID	Service
0x48	LdCom_TpTxConfirmation
0x46	LdCom_StartOfReception
0x44	LdCom_CopyRxData
0x45	LdCom_TpRxIndication
0x42	LdCom_RxIndication
0x41	LdCom_TriggerTransmit
0x40	LdCom_TxConfirmation

Table 2-3 Service IDs

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

Error Code	Description
0x01	API service called with wrong parameter
0x02	API service used without module initialization
0x03	API service called with NULL_PTR as parameter
0x04	API service called with wrong Pdu ID
0x05	API service called with wrong Signal ID

Table 2-4 Errors reported to DET

3 Integration

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

3.1 Scope of Delivery

The delivery of the LdCom contains the files which are described in the chapters 3.1.1 and 3.1.2:

3.1.1 Static Files

File Name	Description
LdCom.c	Source file of the LdCom module
LdCom.h	Main header file which shall be included by modules using the LdCom

Table 3-1 Static files

3.1.2 Dynamic Files

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

File Name	Description
LdCom_Cfg.h	This file contains: > global constant macros > global function macros > global data types and structures > global data prototypes > global function prototypes of CONFIG-CLASS PRE-COMPILE data.
LdCom_Cbk.h	This is the generated header file of LdCom containing prototypes for lower layers.
LdCom_Lcfg.h	This file contains: > global constant macros > global function macros > global data types and structures > global data prototypes > global function prototypes of CONFIG-CLASS LINK data.
LdCom_Lcfg.c	This file contains: > local constant macros > local function macros > local data types and structures > local data prototypes > local data > global data of CONFIG-CLASS LINK data.

File Name	Description
LdCom_PBcfg.h	This file contains: > global constant macros > global function macros > global data types and structures > global data prototypes > global function prototypes of CONFIG-CLASS POST-BUILD data.
LdCom_PBcfg.c	This file contains: > local constant macros > local function macros > local data types and structures > local data prototypes > local data > global data of CONFIG-CLASS POST-BUILD data.

Table 3-2 Generated files

4 API Description

4.1 Services provided by LdCom

4.1.1 LdCom_Init

Prototype	
`void LdCom_Init (const LdCom_ConfigType *config)`	
Parameter	
config	NULL_PTR if LDCOM_USE_INIT_POINTER is STD_OFF. Pointer to the LdCom configuration data if LDCOM_USE_INIT_POINTER is STD_ON.
Return Code	
void	none
Functional Description	
This service initializes internal and external interfaces and variables of the AUTOSAR LdCom layer for the further processing.	
Particularities and Limitations	
The function is used by the Ecu State Manager	
Caution LdCom_Init() shall not pre-empt any LdCom function. The rest of the system must guarantee that LdCom_Init() is not called in such a way.	
Pre-Conditions	
LdCom_InitMemory() has to be executed previously, if the startup code does not initialize variables.	
Call Context	
The function must be called on task level and must not be interrupted by other administrative function calls.	

Table 4-1 LdCom_Init

4.1.2 LdCom_InitMemory

Prototype	
`void LdCom_InitMemory (void)`	
Parameter	
void	none
Return Code	
void	none
Functional Description	
The function initializes variables, which cannot be initialized with the startup code.	
Particularities and Limitations	
The function is used by the application.	
Pre-Conditions	

LdCom_Init() is not called yet.
Call Context
The function must be called on task level.

Table 4-2 LdCom_InitMemory

4.1.3 LdCom_DeInit

Prototype	
`void LdCom_DeInit (void)`	
Parameter	
void	none
Return Code	
void	none
Functional Description	
This service stops the LdCom. The module is put into an uninitialized state.	
Particularities and Limitations	
The function is used by the application.	
Caution LdCom_DeInit() shall not pre-empt any LdCom function. The rest of the system must guarantee that LdCom_DeInit() is not called in such a way.	
Pre-Conditions	
none	
Call Context	
The function must be called on task level and must not be interrupted by other administrative function calls.	

Table 4-3 LdCom_DeInit

4.1.4 LdCom_GetVersionInfo

Prototype	
`void LdCom_GetVersionInfo (Std_VersionInfoType *versioninfo)`	
Parameter	
versioninfo	Pointer to where to store the version information of this module.
Return Code	
void	none
Functional Description	
Returns the version information of the LdCom module.	
Particularities and Limitations	
none	
Pre-Conditions	

none
Call Context
The function can be called on interrupt and task level.

Table 4-4 LdCom_GetVersionInfo

4.1.5 LdCom_Transmit

Prototype	
`Std_ReturnType` **`LdCom_Transmit`** `(PduIdType Id, const PduInfoType *PduInfoPtr)`	
Parameter	
Id	ID of the LdCom I-PDU/Signal to be transmitted
PduInfoPtr	Payload information of the I-PDU/Signal (data pointer and data length)
Return Code	
Std_ReturnType	> E_OK: The request was accepted by the LdCom and by the destination layer. > E_NOT_OK: The LdCom is not initialized or the ID is not valid or the ID is not forwarded in this predefined variant or the PduInfoPtr parameter is not valid or the request was not accepted by the destination module.
Functional Description	
The function serves to request the transmission of an I-PDU/Signal. LdCom evaluates the I-PDU/Signal handle and identifies the destination I-PDU. The call is routed to the PduR module using the appropriate I-PDU handle of the PduR.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
LdCom is initialized.	
Call Context	
This function can be called on interrupt and task level and has not to be interrupted by other LdCom_Transmit() calls for the same ID.	

Table 4-5 LdCom_Transmit

4.2 Services used by LdCom

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

Component	API
Rte	IPdu specific configurable callback/notification functions
PduR	PduR_LdComTransmit
DET	Det_ReportError

Table 4-6 Services used by the LdCom

4.3 Callback Functions

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

4.3.1 LdCom_RxIndication

Prototype	
`void LdCom_RxIndication (PduIdType RxPduId, const PduInfoType *PduInfoPtr)`	
Parameter	
RxPduId	ID of the LdCom I-PDU/Signal that has been received.
PduInfoPtr	Payload information of the received I-PDU/Signal (pointer to data and data length).
Return Code	
void	none
Functional Description	
This function is called by the lower layer when an LdCom IF I-PDU has been received. LdCom evaluates the I-PDU/Signal handle and identifies the destination Signal. The call is routed to the upper layer module using the appropriate API.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
LdCom is initialized.	
Call Context	
The function can be called on interrupt and task level.	

Table 4-7 LdCom_RxIndication

4.3.2 LdCom_TxConfirmation

Prototype
`void LdCom_TxConfirmation (PduIdType TxPduId)`

Parameter	
TxPduId	ID of the LdCom I-PDU/Signal that has been transmitted.
Return Code	
void	none
Functional Description	
This function is called by the lower layer after an Ldcom IF I-PDU has been transmitted. LdCom evaluates the I-PDU/Signal handle and identifies the destination Signal. The call is routed to the upper layer module using the appropriate API.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
LdCom is initialized.	
Call Context	
The function can be called on interrupt and task level.	

Table 4-8 LdCom_TxConfirmation

4.3.3 LdCom_TriggerTransmit

Prototype	
`Std_ReturnType` **`LdCom_TriggerTransmit`** `(PduIdType TxPduId, PduInfoType *PduInfoPtr)`	
Parameter	
TxPduId	ID of the LdCom I-PDU/Signal that is requested to be transmitted.
PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLength. On return, the service will indicate the length of the copied SDU data in SduLength.
Return Code	
Std_ReturnType	> E_OK: The SDU has been copied and the SduLength indicates the number of copied bytes. > E_NOT_OK: A Det error occurred and the SDU has not been copied and the SduLength has not been set.
Functional Description	
This function is called by the lower layer when an LdCom IF I-PDU shall be transmitted. LdCom evaluates the I-PDU/Signal handle and identifies the destination Signal. The call is routed to the upper layer module using the appropriate API.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
LdCom is initialized.	
Call Context	
The function can be called on interrupt and task level.	

Table 4-9 LdCom_TriggerTransmit

4.3.4 LdCom_StartOfReception

Prototype	
`BufReq_ReturnType LdCom_StartOfReception (PduIdType id, const PduInfoType *info, PduLengthType TpSduLength, PduLengthType *bufferSizePtr)`	
Parameter	
id	ID of LdCom I-PDU/Signal that shall be received.
info	Payload and MetaData information of the received I-PDU (pointer to data/MetaData and data length).
TpSduLength	Total length of the TP I-PDU to be received.
bufferSizePtr	The LdCom returns in this value the remaining TP buffer size to the lower layer.
Return Code	
BufReq_ReturnType	> BUFREQ_OK: Connection has been accepted. RxBufferSizePtr indicates the available receive buffer. > BUFREQ_E_NOT_OK: Connection has been rejected. RxBufferSizePtr remains unchanged. > BUFREQ_E_OVFL: In case the related upper layer module has rejected the reception. > BUFREQ_E_BUSY: The case the related upper layer module is currently busy.
Functional Description	
This function is called by the lower layer to indicate the start of an incoming TP connection. LdCom evaluates the I-PDU/Signal handle and identifies the destination Signal. The call is routed to the upper layer module using the appropriate API.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
LdCom is initialized.	
Call Context	
The function can be called on interrupt and task level.	

Table 4-10 LdCom_StartOfReception

4.3.5 LdCom_CopyRxData

Prototype	
`BufReq_ReturnType LdCom_CopyRxData (PduIdType id, const PduInfoType *info, PduLengthType *bufferSizePtr)`	
Parameter	
id	ID of LdCom I-PDU/Signal that shall be received.
info	Payload information of the received TP segment (pointer to data and data length).
bufferSizePtr	Remaining receive buffer size after completion of this call.

Return Code	
BufReq_ReturnType	> BUFREQ_OK: Data successfully copied. RxBufferSizePtr indicates the remaining receive buffer size. > BUFREQ_E_NOT_OK: Data was not copied because an error occurred.
Functional Description	
This function is called by the lower layer once upon reception of each TP segment. LdCom evaluates the I-PDU/Signal handle and identifies the destination Signal. The call is routed to the upper layer module using the appropriate API.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
LdCom is initialized.	
Call Context	
The function can be called on interrupt and task level.	

Table 4-11 LdCom_CopyRxData

4.3.6 LdCom_TpRxIndication

Prototype	
`void LdCom_TpRxIndication (PduIdType id, Std_ReturnType result)`	
Parameter	
id	ID of LdCom I-PDU/Signal that has been received.
result	Indicates the reception result of the TP I-PDU.
Return Code	
void	none
Functional Description	
This function is called by the lower layer after an LdCom TP I-PDU has been received. LdCom evaluates the I-PDU/Signal handle and identifies the destination Signal. The call is routed to the upper layer module using the appropriate API.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
LdCom is initialized.	
Call Context	
The function can be called on interrupt and task level.	

Table 4-12 LdCom_TpRxIndication

4.3.7 LdCom_CopyTxData

Prototype	
`BufReq_ReturnType LdCom_CopyTxData (PduIdType id, const PduInfoType *info, RetryInfoType *retry, PduLengthType *availableDataPtr)`	
Parameter	
id	ID of LdCom I-PDU/Signal that shall be transmitted.
info	Provides the destination buffer and the number of bytes to be copied. An SduLength of 0 is possible in order to poll the available transmit data count. In this case no data has to be copied and SduDataPtr might be invalid.
retry	This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems.
availableDataPtr	Remaining transmit buffer size after completion of this call.
Return Code	
BufReq_ReturnType	> BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. > BUFREQ_E_BUSY: The transmission buffer is actually not available (implementation specific). > BUFREQ_E_NOT_OK: Data has not been copied.
Functional Description	
This function is called by the lower layer once upon transmission of each TP segment. LdCom evaluates the I-PDU/Signal handle and identifies the destination Signal. The call is routed to the upper layer module using the appropriate API.	
Particularities and Limitations	
The function is called by the lower layer.	
Pre-Conditions	
LdCom is initialized.	
Call Context	
The function can be called on interrupt and task level.	

Table 4-13 LdCom_CopyTxData

4.3.8 LdCom_TpTxConfirmation

Prototype	
`void LdCom_TpTxConfirmation (PduIdType id, Std_ReturnType result)`	
Parameter	
id	ID of LdCom I-PDU/Signal that has been transmitted.
result	Indicates the transmission result of the TP I-PDU.
Return Code	
void	none

Functional Description
This function is called by the lower layer after a TP I-PDU has been transmitted. LdCom evaluates the I-PDU/Signal handle and identifies the destination Signal. The call is routed to the upper layer module using the appropriate API.
Particularities and Limitations
The function is called by the lower layer.
Pre-Conditions
LdCom is initialized.
Call Context
The function can be called on interrupt and task level.

Table 4-14 LdCom_TpTxConfirmation

4.4 Configurable Interfaces

4.4.1 Notifications

At its configurable interfaces the LdCom defines notifications that can be mapped to callback functions provided by other modules. The mapping is not statically defined by the LdCom but can be performed at configuration time. The function prototypes that can be used for the configuration have to match the appropriate function prototype signatures, which are described in the following sub-chapters.

4.4.1.1 I-Pdu/Signal Rx Indication

Prototype	
`void` **`[I-Pdu/Signal Rx Indication name]`** `(const PduInfoType *Info)`	
Parameter	
`Info`	Payload information of the received I-PDU/Signal (pointer to data and data length).
Return code	
`void`	none
Functional Description	
This function is called when an LdCom IF I-PDU has been received.	
Particularities and Limitations	
none	
Call context	
The function is called on interrupt and task level.	

Table 4-15 I-Pdu/Signal Rx Indication

4.4.1.2 I-Pdu/Signal Tx Confirmation

Prototype	
`void` **`[I-Pdu/Signal Tx Confirmation name]`** `(void)`	
Parameter	
`void`	none
Return code	
`void`	none
Functional Description	
This function is called when an LdCom IF I-PDU has been transmitted.	
Particularities and Limitations	
none	
Call context	
The function is called on interrupt and task level.	

Table 4-16 I-Pdu/Signal Tx Confirmation

4.4.1.3 I-Pdu/Signal TriggerTransmit

Prototype	
`Std_ReturnType` **`[I-Pdu/Signal TriggerTransmit name]`** `(PduInfoType *PduInfoPtr)`	
Parameter	
`PduInfoPtr`	Payload information of the received I-PDU/Signal (pointer to data and data length).
Return code	
`Std_ReturnType`	> E_OK: The SDU has been copied and the SduLength indicates the number of copied bytes. > E_NOT_OK: A Det error occurred and the SDU has not been copied and the SduLength has not been set.
Functional Description	
This function is called when an LdCom IF I-PDU shall be transmitted.	
Particularities and Limitations	
none	
Call context	
The function is called on interrupt and task level.	

Table 4-17 I-Pdu/Signal TriggerTransmit

4.4.1.4 I-Pdu/Signal Tp StartOfReception

Prototype	
`BufReq_ReturnType` **`[I-Pdu/Signal Tp StartOfReception name]`** `(const PduInfoType *SduInfoPtr, PduLengthType SduLength, PduLengthType *RxBufferSizePtr)`	
Parameter	
`SduInfoPtr`	Payload and MetaData information of the received I-PDU (pointer to data/MetaData and data length).
`SduLength`	Total length of the TP I-PDU to be received.
`RxBufferSizePtr`	Returned remaining TP buffer size to the lower layer.
Return code	
`BufReq_ReturnType`	> BUFREQ_OK: Connection has been accepted. RxBufferSizePtr indicates the available receive buffer. > BUFREQ_E_NOT_OK: Connection has been rejected. RxBufferSizePtr remains unchanged. > BUFREQ_E_OVFL: In case the reception was rejected. > BUFREQ_E_BUSY: In the receiver is currently busy.
Functional Description	
This function is called to indicate the start of an incoming TP connection.	
Particularities and Limitations	
none	

Call context
The function is called on interrupt and task level.

Table 4-18 I-Pdu/Signal StartOfReception

4.4.1.5 I-Pdu/Signal Tp CopyRxData

Prototype	
`BufReq_ReturnType` **`[I-Pdu/Signal Tp CopyRxData name]`** `(const PduInfoType *SduInfoPtr, PduLengthType *RxBufferSizePtr)`	
Parameter	
`SduInfoPtr`	Payload information of the received TP segment (pointer to data and data length).
`RxBufferSizePtr`	Remaining receive buffer size after completion of this call.
Return code	
`BufReq_ReturnType`	> BUFREQ_OK: Data successfully copied. RxBufferSizePtr indicates the remaining receive buffer size. > BUFREQ_E_NOT_OK: Data was not copied because an error occurred.
Functional Description	
This function is called once upon reception of each TP segment.	
Particularities and Limitations	
none	
Call context	
The function is called on interrupt and task level.	

Table 4-19 I-Pdu/Signal CopyRxData

4.4.1.6 I-Pdu/Signal Tp Rx Indication

Prototype	
`void` **`[I-Pdu/Signal Tp Rx Indication name]`** `(Std_ReturnType Result)`	
Parameter	
`Result`	Indicates the reception result of the TP I-PDU.
Return code	
`void`	none
Functional Description	
This function is called after an LdCom TP I-PDU has been received.	
Particularities and Limitations	
none	
Call context	
The function is called on interrupt and task level.	

Table 4-20 I-Pdu/Signal Tp Rx Indication

4.4.1.7 I-Pdu/Signal Tp CopyTxData

Prototype	
`BufReq_ReturnType` **`[I-Pdu/Signal Tp CopyTxData name]`** `(const PduInfoType *SduInfoPtr, RetryInfoType *RetryInfoPtr, PduLengthType *TxDataCntPtr)`	
Parameter	
`SduInfoPtr`	Provides the destination buffer and the number of bytes to be copied. An SduLength of 0 is possible in order to poll the available transmit data count. In this case no data has to be copied and SduDataPtr might be invalid.
`RetryInfoPtr`	This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems.
`TxDataCntPtr`	Remaining transmit buffer size after completion of this call.
Return code	
`BufReq_ReturnType`	> BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. > BUFREQ_E_BUSY: The transmission buffer is actually not available (implementation specific). > BUFREQ_E_NOT_OK: Data has not been copied.
Functional Description	
This function is called by the lower layer once upon transmission of each TP segment.	
Particularities and Limitations	
none	
Call context	
The function is called on interrupt and task level.	

Table 4-21 I-Pdu/Signal CopyTxData

4.4.1.8 I-Pdu/Signal Tp Tx Confirmation

Prototype	
`void` **`[I-Pdu/Signal Tp Tx Confirmation name]`** `(Std_ReturnType result)`	
Parameter	
`result`	Indicates the transmission result of the TP I-PDU.
Return code	
`Void`	none
Functional Description	
This function is called by the lower layer after a TP I-PDU has been transmitted.	
Particularities and Limitations	
none	

Call context
The function is called on interrupt and task level.

Table 4-22 I-Pdu/Signal Tp Tx Confirmation

5 Configuration

5.1 Configuration Variants

The LdCom supports the configuration variants

- VARIANT-PRE-COMPILE
- VARIANT-POST-BUILD-LOADABLE
- VARIANT-POST-BUILD-SELECTABLE

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

6 Glossary and Abbreviations

6.1 Glossary

Term	Description
Cfg5	DaVinci Configurator

Table 6-1 Glossary

6.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
EAD	Embedded Architecture Designer
ECU	Electronic Control Unit
HIS	Hersteller Initiative Software
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
PPORT	Provide Port
RPORT	Require Port
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 6-2 Abbreviations

7 Contact

Visit our website for more information on

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

www.vector.com