VECTOR

MICROSAR I-PDU Multiplexer

Technical Reference

Version 2.06.00

Authors	Safiulla Shakir, Markus Bart, Gunnar Meiss
Status	Released

Document Information

History

Author	Date	Version	Remarks
Safiulla Shakir	2011-12-06	1.00.00	Initial CFG5 version derived from TechnicalReference_ASR_IpduM.pdf
Markus Bart	2012-08-23	2.00.00	ESCAN00058313 AR4-160: Support AUTOSAR 4.0.3
Markus Bart	2013-01-29	2.01.00	ESCAN00063294 AR4-197: Support BIG_ENDIAN Copy Segments in IpduM
Gunnar Meiss	2013-04-04	2.02.00	ESCAN00064368 AR4-325: Post-Build Loadable
Markus Bart	2014-11-05	2.03.00	AR4-698: Post-Build Selectable (Identity Manager)
Markus Bart	2014-12-01	2.04.00	FEAT-229: Support 16bit selector in IpduM [AR4-927]
Markus Bart	2015-08-11	2.05.00	FEAT-1315: IPDUM for CAN-FD supporting nPdu2Frame-Mapping
Gunnar Meiss	2016-02-25	2.06.00	FEAT-1631: Trigger Transmit API with SduLength In/Out according to ASR4.2.2

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_IPDUMultiplexer.pdf	2.2.0
[2]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	1.6.0
[3]	Vector	TechnicalReference_PostBuildLoadable.pdf	1.0.0

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	▶ Component in conformance with AUTOSAR 3.2.1
2.00	▶ AUTOSAR 4.0.3
2.01	▶ BIG_ENDIAN copy segments
2.02	▶ Support post-build loadable
3.00	▶ Support post-build selectable ▶ Support deleting container at post-build time
4.00	▶ Extend support for module initialization
6.00	▶ Support 16 bit selector ▶ Support selector using more than one byte (i.e. crossing byte boundaries)
6.01	▶ Support nPdu-to-Frame Mapping for CAN-FD

Table 1-1 Component history

2 Introduction

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

Supported AUTOSAR Release*:	4.x
Supported Configuration Variants:	PRE-COMPILE [SELECTABLE] POST-BUILD-LOADABLE [SELECTABLE]

2.1 Architecture Overview

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

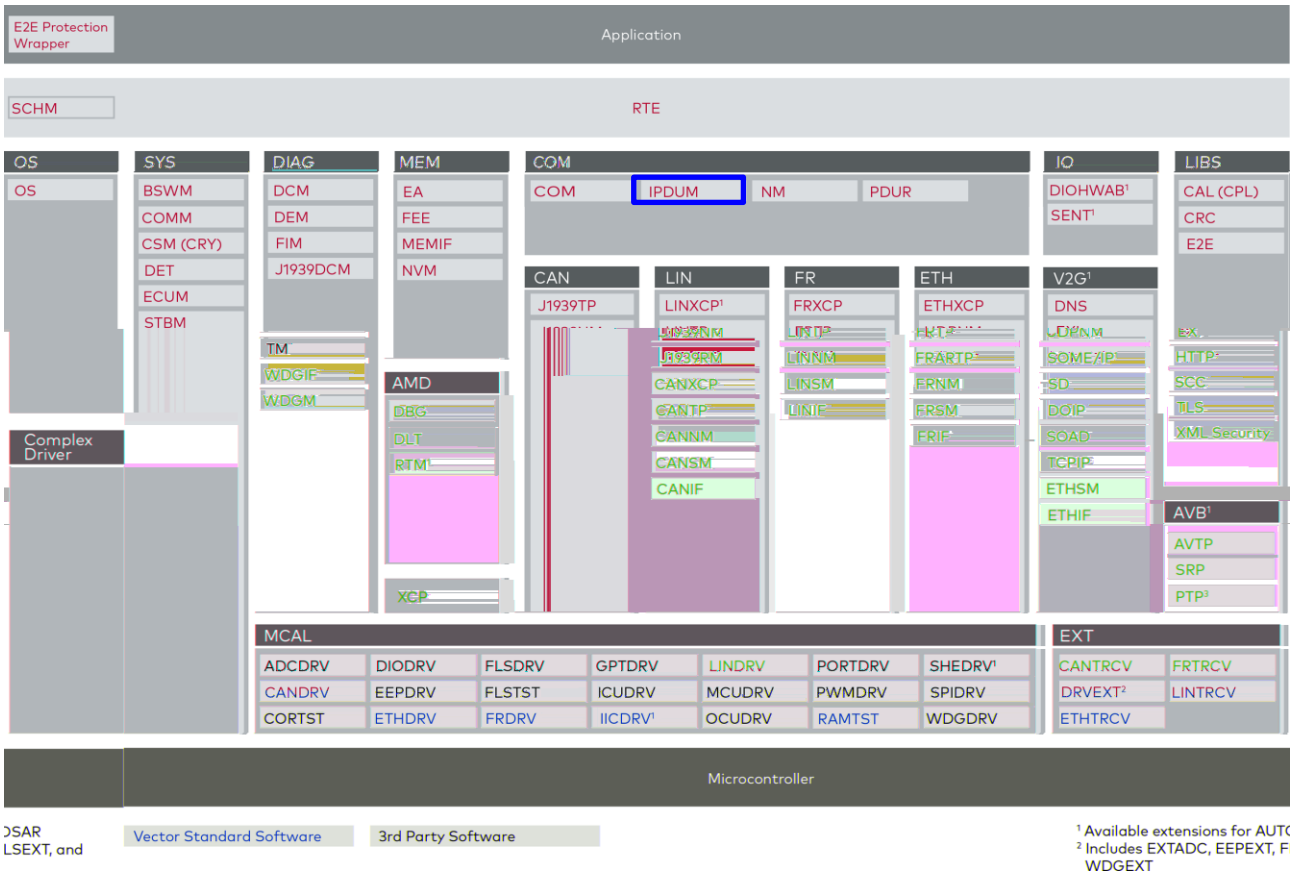

Figure 2-1 AUTOSAR 4.1 Architecture Overview

Figure 2-2 AUTOSAR architecture

The next figure shows the interfaces to adjacent modules of the IPDUM. These interfaces are described in chapter 4.2.

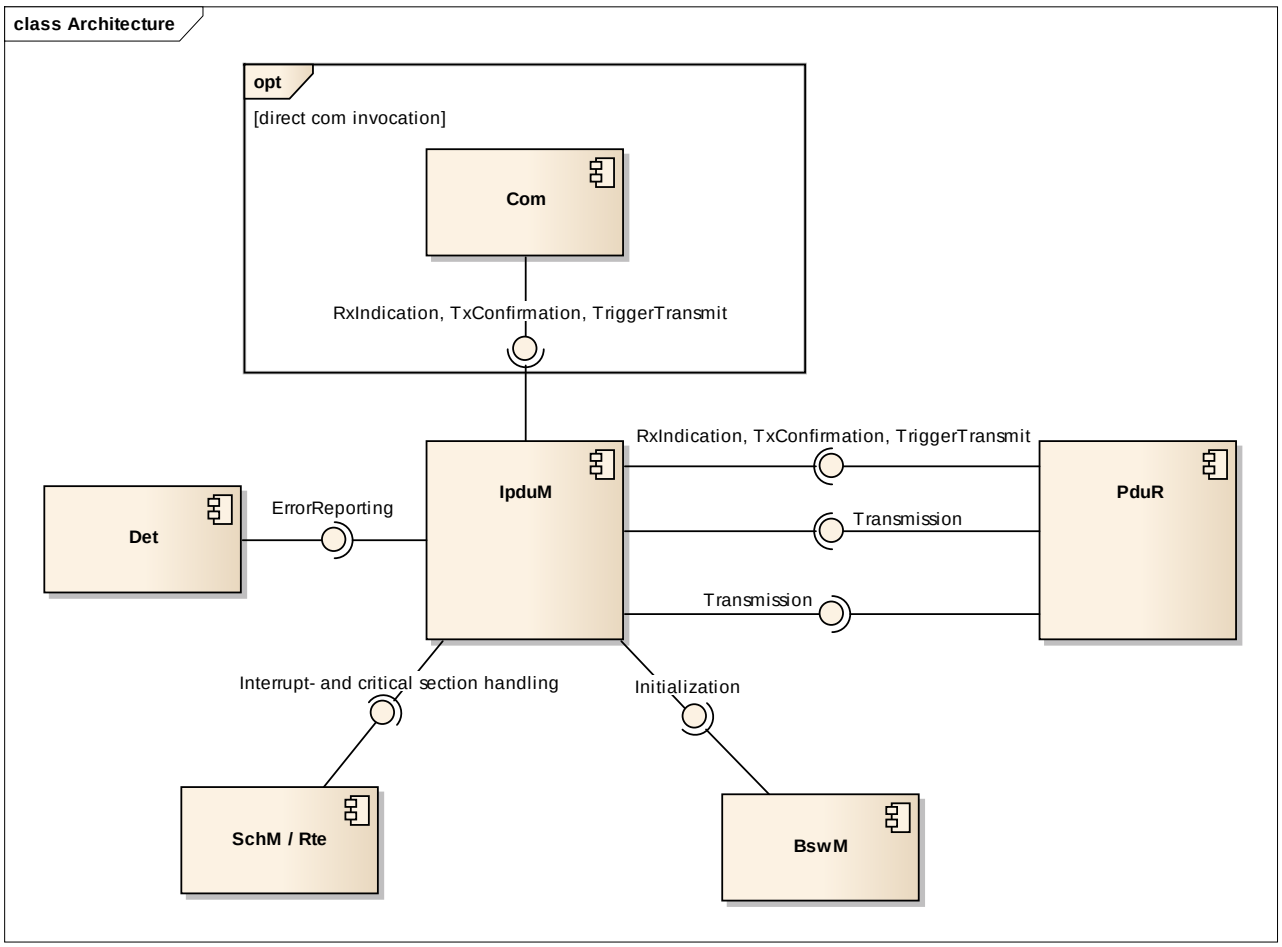

Figure 2-3 Interfaces to adjacent modules of the IPDUM

Normally the IpduM only interacts with the PduR. If required the IpduM can interface Com directly bypassing the PduR while indicating receptions and confirming transmissions.

The IpduM also uses the interfaces to the BSW Scheduler and Det.

3 Functional Description

3.1 Features

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

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
- Table 3-3 Features provided beyond the AUTOSAR standard

For further information of not supported features see also chapter 6.

The following features specified in [1] are supported:

Supported Feature
Configuration
> Multiplexing of the static and dynamic parts.
> Event base transmission triggering of the multiplexed I-PDU(s).
> Forwarding confirmations of all transmitted multiplexed I-PDUs to the BSW module AUTOSAR Com.
> Timeout handling of transmission confirmations.
> De-multiplexing and indicating receptions of the de-multiplexed dynamic parts and static part to the BSW module AUTOSAR Com.
> Static part configuration support in multiplex I-PDU(s).
> AUTOSAR EcuC format
> Initialization of transmission multiplex I-PDU buffers. This is required by e.g. FrIf in case the transmission of a multiplex I-PDU is triggered by interface layer BSW module before the transmission of their mapped dynamic and or static parts is requested.
> Just-in-Time Update of dynamic and / or static parts.
> DET error reporting.

Table 3-1 Supported AUTOSAR standard conform features

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

Not Supported AUTOSAR Standard Conform Features
> Configuring of transmission confirmations of transmitted multiplexed I-PDUs AUTOSAR Com.

Table 3-2 Not supported AUTOSAR standard conform features

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
> nPdu-to-Frame mapping for CAN-FD

Table 3-3 Features provided beyond the AUTOSAR standard

3.2 Initialization

The IpduM is initialized by calling the API `IpduM_Init()`. This is done by the AUTOSAR BSW module EcuM. If this is not the case, an adequate and suitable solution has to be provided.

The API initializes all the transmission multiplex I-PDU(s) buffer(s) and all runtime relevant module variables. The successful execution of the API initializes the IpduM.

Caution
No IpduM function should pre-empt `IpduM_Init()`. The function should be called on task level and should not to be interrupted by other administrative function calls.

3.3 States

The IpduM has the states: uninitialized and initialized. By calling IpduM_Init(), the state is changed from uninitialized to initialized.

3.4 Main Functions

IpduM provides all functions described in [1]. The function `IpduM_MainFunction()` should be called cyclically by the Basic Software Scheduler or a similar component. The `IpduM_MainFunction()` primarily takes care of the timeout handling of the multiplexed I-PDU transmission confirmations.

3.5 Error Handling

3.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [1], if development error reporting is enabled (i.e. pre-compile parameter `IPDUM_DEV_ERROR_DETECT==STD_ON`). The DET reporting can be enabled during pre-compile time.

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 IPDUM ID is 52.

The reported services IDs identify the IPDUM services as specified by [1]. For the service IDs and their related service names see section 8.3 of [1].

The errors reported to DET are described in section 7.6 of [1].

3.6 nPdu-to-Frame mapping for CAN-FD

This feature was backported from AUTOSAR 4.2.1. It can be used to collect multiple PDUs and transfer them in one CAN-FD frame, for saving bandwidth or tunneling several classic CAN busses over one CAN-FD bus. The IpduM module handles packing and unpacking of those frames similar to the multiplexing feature.

Several “contained” PDUs can be put into a single “container” PDU in arbitrary order and unpacked in the same order on the receiver side. Transmission is triggered by one of these conditions:

- Contained PDU send timeout runs out.
- Container PDU send timeout runs out.
- Size threshold of container PDU is reached.
- Contained PDU is configured for immediate transmission.
- Container PDU is configured for immediate transmission.

On the receiver side, the contained PDUs are retrieved from the container PDU again and RxIndications are created for every single contained PDU in the same order they were put into the container PDU. For even greater flexibility, a container PDU on the receiver side can be configured to accept any contained PDU instead of just the ones that are configured.

4 Integration

This chapter provides information necessary for the integration of the MICROSAR IPDUM into an application environment of an ECU.

4.1 Scope of Delivery

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

4.1.1 Static Files

File Name	Source Code Delivery	Object Code Delivery	Description
IpduM.c	■		This is the source file of the IPDUM
IpduM.h	■	■	This is the header file of IPDUM
IpduM_Cbk.h	■	■	This is the callback ile 2

File Name	Description
	> local data prototypes > local data > global data of CONFIG-CLASS LINK and PRE-COMPILE data.
IpduM_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.
IpduM_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 4-2 Generated files

4.2 Critical Sections

The critical section IPDUM_EXCLUSIVE_AREA_0 has to lock global interrupts to protect common critical sections.

5 API Description

For an interfaces overview please see Figure 2-3.

5.1 Services provided by IPDUM

5.1.1 IpduM_InitMemory

Prototype	
`void (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	
> IpduM_Init() is not called yet. > The function is called by the Application.	
Expected Caller Context	
> The function must be called on task level.	

Table 5-1 IpduM_InitMemory

5.1.2 IpduM_Init

Prototype	
`void (const IpduM_ConfigType* config)`	
Parameter	
`config`	> NULL_PTR in the IPDUM_CONFIGURATION_VARIANT_PRECOMPILE > Pointer to the IpduM configuration data in the IPDUM_CONFIGURATION_VARIANT_POSTBUILD_LOADABLE
Return code	
void	none
Functional Description	
This function initializes the IpduM and performs configuration consistency checks. If the initialization is performed successfully the IpduM is initialized.	
Particularities and Limitations	
> IpduM_InitMemory() has been executed, if the startup code does not initialize variables. The IpduM is in the state uninitialized. > No IpduM function shall not pre-empt IpduM_Init()	
Expected Caller Context	

> The function is used by the Ecu State Manager.

Table 5-2 IpduM_Init

5.1.3 IpduM_Transmit

Prototype	
`StdReturnType (PduIdType PdumTxPduId, const PduInfoType* PduInfoPtr)`	
Parameter	
`PdumTxPduId`	ID of transmit request (static or dynamic part). Identifies the payload.
`PduInfoPtr`	Payload information of the I-PDU (pointer to data and data length).
Return code	
`StdReturnType`	E_OK: The request was accepted by the IpduM and by the destination layer. E_NOT_OK: The request was rejected because: the IpduM is not initialized or the provided PduInfoPtr is NULL or the SduDataPtr of the PduInfoPtr is NULL or the PdumTxPduId is not in the expected range or the destination layer did not accept the transmission request.
Functional Description	
The function serves to request the transmission of a static or dynamic part.	
Particularities and Limitations	
> The IpduM_Init() has been executed successfully. > The function is non-reentrant for the same PdumTxPduId.	
Expected Caller Context	
> The function is called by the PduR. > The function is called in an interrupt context and at the task level	

Table 5-3 IpduM_Transmit

5.1.4 IpduM_MainFunction

Prototype	
`void (void)`	
Parameter	
`void`	none
Return code	
`void`	none
Functional Description	
This function shall perform the processing of the IpduM processing which is not directly initiated by calls from the PduR. A call to IpduM_MainFunction returns simply if IpduM was not previously initialized with a call to IpduM_Init.	
Particularities and Limitations	
> The IpduM_Init() has been executed successfully. > The function is called cyclically by the BSW scheduler	
Expected Caller Context	
> The function is called on task level	

Table 5-4 IpduM_MainFunction

5.1.5 IpduM_GetVersionInfo

Prototype	
`void (Std_VersionInfoType* versioninfo)`	
Parameter	
`versioninfo`	Pointer to the structure to write the version info to.
Return code	

5.2 Services used by IPDUM

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

Component	API
Det	Det_ReportError
PduR	PduR_IpduMTransmit
PduR	PduR_IpduMTxConfirmation
PduR	PduR_IpduMRxIndication
PduR	PduR_IpduMTriggerTransmit
Com	Com_RxIndication
Com	Com_TxConfirmation
Com	Com_TriggerTransmit
BSW Scheduler	SchM_Enter_IpduM
BSW Scheduler	SchM_Exit_IpduM

Table 5-6 Services used by the IPDUM

5.3 Callback Functions

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

5.3.1 IpduM_RxIndication

Prototype	
`void (PduIdType RxPduId, const PduInfoType* PduInfoPtr)`	
Parameter	
`RxPduId`	ID of received multiplexed I-PDU. Identifies the payload.
`PduInfoPtr`	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Return code	
void	none
Functional Description	
This service indicates the complete reception of an IpduM I-PDUs. It de-multiplexes the received multiplexed I-PDU.	
Particularities and Limitations	
> The IpduM_Init() has been executed successfully > The function is non-reentrant for the same PdumRxPduId.	
Expected Caller Context	
> The function is called by the PduR. The function is called in an interrupt context or at the task level	

Table 5-7 IpduM_RxIndication

5.3.2 IpduM_TxConfirmation

Prototype	
`void (PduIdType TxPduId)`	
Parameter	
`TxPduId`	ID of transmitted multiplexed IPDU.
Return code	
`void`	none
Functional Description	
This service confirms the transmissions of t s	

Expected Caller Context
> The function is called by the PduR. The function is called in an interrupt context or at the task level

Table 5-8 IpduM_TxConfirmation

5.3.3 IpduM_TriggerTransmit

Prototype	
`Std_ReturnType (PduIdType TxPduId, PduInfoType* PduInfoPtr)`	
Parameter	
`TxPduId`	ID of transmission multiplexed IPDU.
`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 IpduM is not initialized or TxPduId is not valid or PduInfoPtr is NULL_PTR or SduDataPtr is NULL_PTR or SduLength is to small or the SDU data could not be updated because the just in time update via a trigger transmit call of the upper layer was not successful for multiplexed messages.
Functional Description	
This service copies the transmission multiplexed IPDU to the provided buffer.	
Particularities and Limitations	
> The IpduM_Init() has been executed successfully > The function is non-reentrant for the same TxPduId	
Expected Caller Context	
> The function is called by the PduR module. The function is called in an interrupt context or at the task level	

Table 5-9 IpduM_TriggerTransmit

6 Configuration

The IPDUM in the EcuC can be configured through the Vector configuration and generation tool CFG5.

6.1 Configuration of Post-Build

The configuration of post-build loadable is described in [3].

7 AUTOSAR Standard Compliance

7.1 Deviations

No deviations.

7.2 Additions/ Extensions

No additions/ extensions.

7.3 Limitations

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

8 Glossary and Abbreviations

8.1 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.
CFG5	Configurator 5 (configuration and generation tool)
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.
Event	An exclusive signal which is assigned to a certain extended task. An event can be used to send binary information to an extended task. The meaning of events is defined by the application. Any task can set an event for an extended task. The event can only be cleared by the task which is assigned to the event.
Interrupt	Processor-specific event which can interrupt the execution of a current program section.
Post-build	This type of configuration is possible after building the software module or the ECU software. The software may either receive parameters of its configuration during the download of the complete ECU software resulting from the linkage of the code, or it may receive its configuration file that can be downloaded to the ECU separately, avoiding a re-compilation and re-build of the ECU software modules. In order to make the post-build time re-configuration possible, the re-configurable parameters shall be stored at a known memory location of ECU storage area.
Scheduler	The algorithm which decides whether a task switch is to be affected and which triggers all necessary internal activities of the operating system is named scheduler. The scheduler decides whether a task switch is possible according to the implemented scheduling policy. The scheduler can be considered as a resource which can be occupied and released by tasks. Thus a task can reserve the scheduler to avoid task switch until the scheduler is released.
Signal	A signal is responsible for the logical transmission and reception of information depending on the restrictions of the physical layer. The definition of the signal contents is part of the database given by the vehicle manufacturer. Signals describe the significance of the individual data segments within a message. Typically bits, bytes or words are used

	for data segments but individual bit combinations are also possible. In the CAN data base, each data segment is assigned a symbolic name, a value range, a conversion formula and a physical unit, as well as a list of receiving nodes.

Table 8-1 Glossary

8.2 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.
Com	AUTOSAR BSW module Com
COM	Communication
DEM	AUTOSAR BSW module Diagnostic Event Manager
DET	AUTOSAR BSW module Development Error Tracer
ECU	Electronic Control Unit
EcuM	AUTOSAR BSW module ECU Manager
HIS	Hersteller Initiative Software
ID	Identifier (e.g. Identifier of a CAN message)
I-PDU	Interactive layer Protocol Data Unit
IpduM	AUTOSAR BSW module I-PDU Multiplexer
IPDUM	I-PDU Multiplexer
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR flavor)
PCI	Protocol Control Information
PDU	Protocol Data Unit
PduR	AUTOSAR BSW module Pdu-Router
RAM	Random Access Memory
ROM	Read-Only Memory
SDU	Service Data Unit
SWS	Software Specification

Table 8-2 Abbreviations

9 Contact

Visit our website for more information on

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

www.vector.com