MICROSAR J1939 Diagnostic Communication Manager

Technical Reference

Version 4.2.0

Authors	Thomas Dedler, Amr Elazhary
Status	Released

Document Information

History

Author	Date	Version	Remarks
Thomas Dedler	2013-02-20	1.0.0	Initial document version
Thomas Dedler	2014-04-14	2.0.0	> J1939Dcm_InitMemory added > StartofReception according to AR4.1.2
Thomas Dedler	2014-08-21	2.1.0	> BETA status removed > Multiple Nodes Limitations added
Thomas Dedler	2015-05-27	3.0.0	Version update for major component release
Amr Elazhary	2016-03-01	4.0.0	> Updated the API J1939Dcm_RequestIndication > Added diagnostic messages 27, 53, 54, and 55. > Indicating the use of new DEM interface API Dem_J1939DcmReadDiagnosticReadiness1
Amr Elazhary	2016-04-27	4.1.0	> Added post-build selectable > Added post-build loadable > Added post-build loadable selectable > Added post-build deletable
Amr Elazhary	2016-06-16	4.2.0	> Added ISOBUS feature

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_SAEJ1939DiagnosticCommunicationManager.pdf	V1.0.4
[2]	AUTOSAR	AUTOSAR_SWS_DET.pdf	V3.4.0
[3]	AUTOSAR	AUTOSAR_SWS_DEM.pdf	V5.1.0
[4]	AUTOSAR	AUTOSAR_SWS_SAEJ1939RequestManager.pdf	V1.1.0
[5]	AUTOSAR	AUTOSAR_SWS_PDURouter.pdf	V4.1.0
[6]	SAE	J1939-73: Application Layer - Diagnostics	FEB2010
[7]	SAE	J1939-21: Data Link Layer	DEC2006
[8]	Vector	TechnicalReference_PostBuildLoadable.pdf	Delivery
[9]	Vector	TechnicalReference_IdentityManager.pdf	Delivery

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 Component History

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

Component Version	New Features
1.0	Initial implementation with BETA status without OBD support
2.0	Implementation of J1939Dcm_StartOfReception according to AR4.1.2
2.1	BETA Status removed Multiple Node Support
3.0	None (new major version due to incompatible J1939Rm API change)
4.0	Added support for DM27, DM53, DM54 and DM55 Added additional content to DM5
4.1	Added post-build modes
4.2	Added ISOBUS feature

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module J1939DCM as specified in [1].

Supported AUTOSAR Release*:	4.1.2	
Supported Configuration Variants:	pre-compile	
Vendor ID:	J1939DCM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	J1939DCM_MODULE_ID	58 decimal (according to ref. [1][4])

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

The J1939DCM handles the diagnostic communication in a J1939 network. SAE J1939-73 (see [6]) defines several so-called Diagnostic Messages (DMs) which are received and/or transmitted over the network by the J1939DCM.

2.1 Architecture Overview

The following figure shows where the J1939DCM is located in the AUTOSAR architecture.

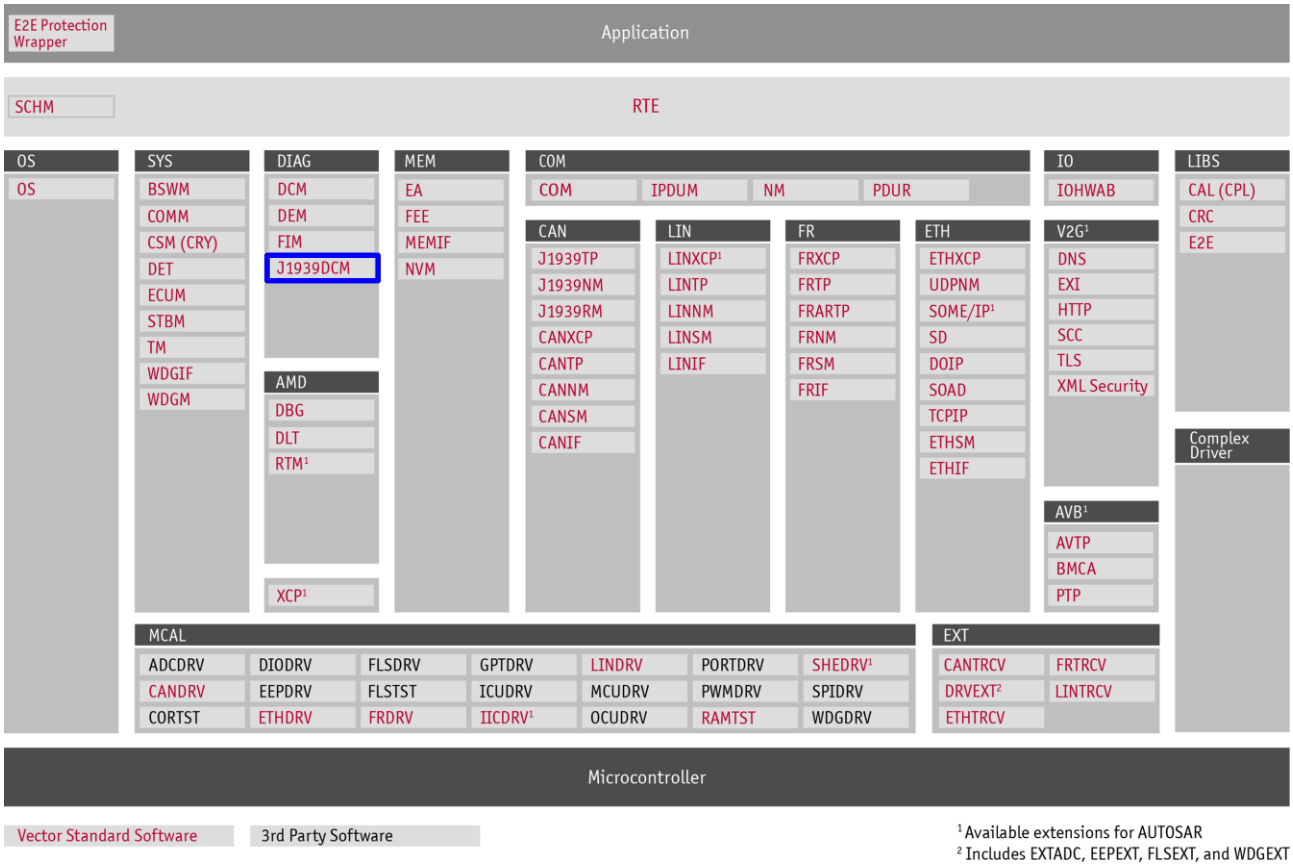

Figure 2-1 AUTOSAR 4.1 Architecture Overview

The next figure shows the interfaces to adjacent modules of the J1939DCM. These interfaces are described in chapter 4.4.

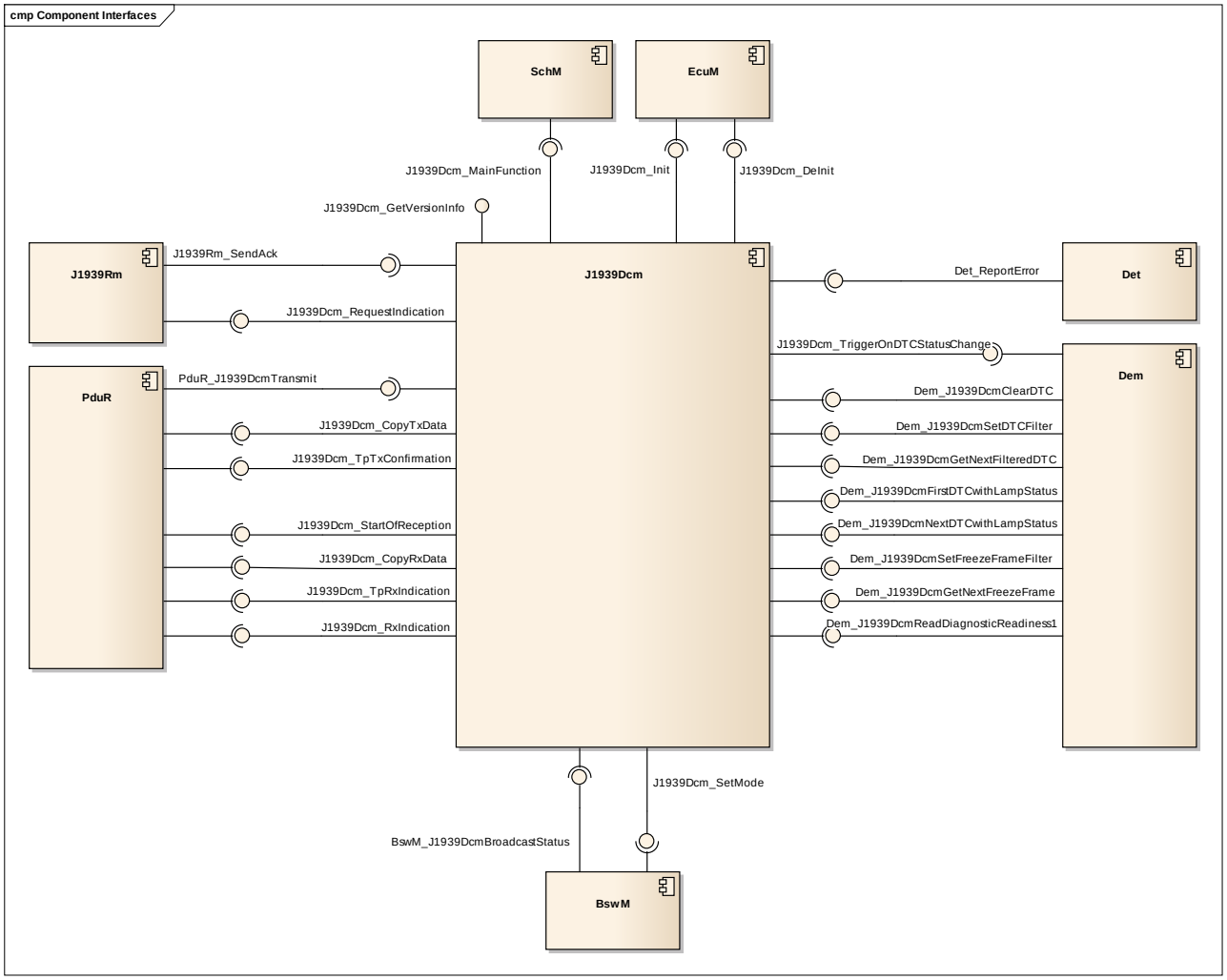

Figure 2-2 Interfaces to adjacent modules of the J1939DCM

3 Functional Description

3.1 Features

The following table gives an overview over the diagnostic messages which are defined by AUTOSAR in [1] and which are supported by the J1939DCM.

Especially OBD related DMs are not yet supported, while the memory access messages are implemented according to the SAE specification ([6]) but not yet defined by AUTOSAR.

Diagnostic Messages	Specified by AUTOSAR	Implemented by J1939DCM
DM1 - Active Diagnostic Trouble Codes	■	■
DM2 - Previously Active Diagnostic Trouble Codes	■	■
DM3 - Diagnostic Data Clear/Reset Of Previously Active DTCs	■	■
DM4 - Freeze Frame Parameters	■	■
DM5 - Diagnostic Readiness 1	■	■
DM6 - Pending DTCs	■	
DM11 - Diagnostic Data Clear/Reset For Active DTCs	■	■
DM12 - Emissions-Related Active Diagnostic Trouble Codes	■	
DM13 - Stop Start Broadcast	■	■
DM14 - Memory Access Request		■
DM15 - Memory Access Response		■
DM16 - Binary Data Transfer		■
DM17 - Boot Load Data		■
DM18 - Security Data		■
DM19 - Calibration Information	■	
DM20 - Monitor Performance Ratio	■	
DM21 - Diagnostic Readiness 2	■	
DM23 - Previously Active Emission Related Faults	■	
DM24 - SPN Support	■	■
DM25 - Expanded Freeze Frame	■	■
DM26 - Diagnostic Readiness 3	■	
DM27 – All Pending DTCs		■
DM28 - Permanent DTCs	■	
DM29 - DTC Counts	■	
DM31 - DTC To Lamp Association	■	■
DM35 - Immediate Fault Status	■	■
DM53 – Active Service Only DTCs		■
DM54 – Previously Active Service Only DTCs		■

Diagnostic Messages	Specified by AUTOSAR	Implemented by J1939DCM
DM55 – Diagnostic Data Clear/Reset for All Service Only DTCs		■

Table 3-1 Implemented Diagnostic Messages

Additional Features which are not related to a specific DM:

Additional Features
Communication State Handling

Table 3-2 Additional Features

3.1.1 Limitations

3.1.1.1 DM1 Transmission on DTC status change

Only the first status change of any DTC within a 1s period will lead to a DM1 transmission. This is a limitation to the AUTOSAR and SAE specifications ([1] and [6]), where a DM1 shall be transmitted for the first status change of each DTC.

3.1.1.2 No OBD Support

OBD related diagnostic messages are not yet supported (see Table 3-1).

DM5, which is also related to OBD, is nevertheless supported, as it contains the OBD Compliance (SPN 1220; see [6]). The OBD Compliance is set to “No OBD”. The number of active and the number of previously active DTCs are imported from DEM. The rest of the parameters are set to zero.

3.1.1.3 Parallel Message Processing and Multiple Nodes

Generally, the J1939DCM is only able to process one message request at a time for each node. Different nodes are handled independently, so it is possible to request a DM simultaneously for two nodes.

However, parallel message processing on different nodes is limited by the fact, that the DEM (which provides the data for most DMs) can only handle one request at a time. If an access to the DEM takes longer than one task cycle, all further requests which require DEM access are rejected.

In case of a directed request, the NACK “Cannot Respond” will be sent. According to the specification of the Acknowledgment PGN 59392 in [7], the data must then be re-requested at a later time.

For broadcast requests, where all configured nodes need to respond simultaneously, this may have the effect that only the first node is able to respond.

3.1.1.4 DM13 and Multiple Nodes

DM13 only affects the configured channels, its implementation is completely independent to the nodes.

3.2 Initialization

If not already done by the startup code, the statically initialized RAM variables must be initialized by calling `J1939Dcm_InitMemory`.

The J1939DCM itself is initialized by calling `J1939Dcm_Init`. The module can be reset with the `J1939Dcm_DeInit`. In uninitialized state, only the `J1939Dcm_GetVersionInfo` API is available.

3.3 States

After it is initialized, the J1939DCM is by default in the state `J1939DCM_OFFLINE`. In this state, requests can neither be received nor are messages transmitted. The state is changed to `J1939DCM_ONLINE` by the API `J1939Dcm_SetState`.

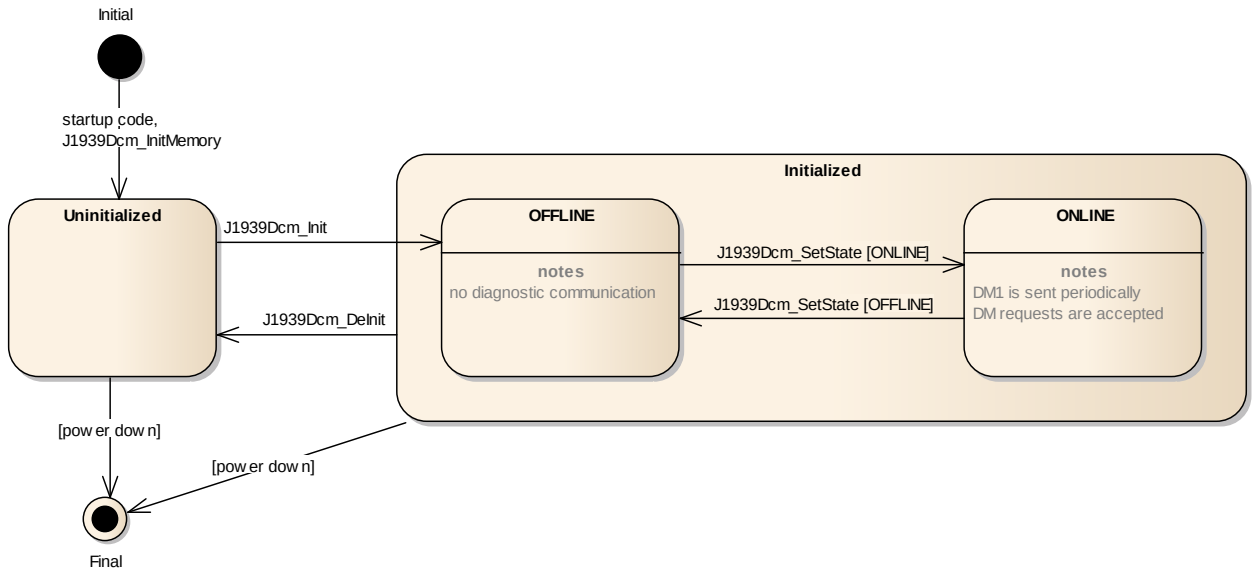

Figure 3-1 J1939DCM Module States

3.4 Main Functionality

Basically, all functionalities of the J1939DCM are asynchronous. Within the context of each API, the requested operation is queued and executed on task level. The basic functionalities of the J1939DCM can be divided into four groups:

3.4.1 Transmission of requested Diagnostic Messages

Most diagnostic messages are requested using the J1939 Request PGN (PGN = 59904). This PGN is handled by the J1939 Request Manager (see [4]) and forwarded to the J1939DCM through the API `J1939Dcm_RequestIndication`. Upon request, the J1939DCM collects the according data and starts transmission.

3.4.2 Transmission of periodic messages

DM1 (always) and DM35 (on request) are transmitted periodically. The J1939DCM regularly collects the related data and transmits the according DM over the bus as broadcast messages.

3.4.2.1 ISOBUS channel

When a specific J1939Dcm channel is configured as ISOBUS, the periodic transmission of DM1 via that very same channel is affected in such a way that the periodic transmission is triggered only when an error is active. The activation of this feature can be done via /J1939Dcm/J1939DcmConfigSet/J1939DcmChannel/J1939DcmIsobusChannel.

3.4.3 Communication State Handling

Besides the ONLINE / OFFLINE state (see 3.3), SAE also specifies the possibility to enable or disable transmission of broadcast messages. This is done through DM13, which can be received by the J1939DCM. A change of the broadcast status is then forwarded to the BswM (API `BswM_J1939DcmBroadcastStatus`).

3.4.4 Memory Access

SAE J1939-73 ([6]) defines DM14 through DM18 to allow a tester to access the memory of an ECU. These messages are supported by the J1939DCM and forwarded to the application, which have to handle the memory access operation.

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 [2], if development error reporting is enabled (i.e. pre-compile parameter `J1939DCM_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 J1939DCM ID is 58.

The reported service IDs identify the services which are described in 5.2. The following table presents the service IDs and the related services:

Service ID	Service
0x01	J1939Dcm_Init
0x02	J1939Dcm_DeInit
0x03	J1939Dcm_GetVersionInfo
0x04	J1939Dcm_MainFunction
0x05	J1939Dcm_CopyRxData
0x06	J1939Dcm_CopyTxData
0x07	J1939Dcm_StartOfReception
0x08	J1939Dcm_TpRxIndication

Service ID	Service
0x09	J1939Dcm_TpTxConfirmation
0x0A	J1939Dcm_DemTriggerOnDTCStatus
0x0B	J1939Dcm_SetState
0x0C	J1939Dcm_TxConfirmation
0x42	J1939Dcm_RxIndication
0x43	J1939Dcm_RequestIndication
0x50	J1939Dcm_MemResponseTransmit
0x51	J1939Dcm_MemDataTransmit
0x80	J1939Dcm_InitMemory

Table 3-3 Service IDs

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

Error Code		Description
0x01	J1939DCM_E_INVALID_PDU_SDU_ID	API called with invalid PDU-ID
0x02	J1939DCM_E_PARAM_POINTER	API called with invalid pointer parameter
0x06	J1939DCM_E_INVALID_STATE	API called although not allowed in current state (e.g. in OFFLINE state)
0x07	J1939DCM_E_PARAM_ID	API called with invalid parameter
0x08	J1939DCM_E_INVALID_NODE	API called with invalid node identifier
0x09	J1939DCM_E_INVALID_PGN	API called with invalid PGN
0x0A	J1939DCM_E_INVALID_PRIO	API called with invalid priority
0x0B	J1939DCM_E_INVALID_CHANNEL	API called with invalid channel identifier
0x10	J1939DCM_E_PARAM_CONFIG	Configuration does not allow desired operation
0x11	J1939DCM_E_OVERRUN	Task overrun detected;
0x20	J1939DCM_E_UNINIT	API called although J1939DCM is not initialized
0x21	J1939DCM_E_REINIT	J1939Dcm_Init have been called although module is already initialized
0xFF	J1939DCM_E_NO_ERROR	Used for local variables initialization

Table 3-4 Errors reported to DET

4 Integration

This chapter gives necessary information for the integration of the MICROSAR J1939DCM into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the J1939DCM contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Description
J1939Dcm.c	Source file of the J1939DCM implementation
J1939Dcm.h	Main header file which shall be included by modules using the J1939DCM
J1939Dcm_Priv.h	Module internal definitions
J1939Dcm_Types.h	Module specific type definitions
J1939Dcm_Cbk.h	Prototypes of callbacks provided by the J1939DCM

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the Configurator 5 configuration tool.

File Name	Description
J1939Dcm_Cfg.h	Precompile definitions
J1939Dcm_Lcfg.c	Configurations tables which can only be changed at link time
J1939Dcm_Lcfg.h	Data prototypes for link time configuration tables
J1939Dcm_PBcfg.c	Configurations tables which may be changed at post-build time
J1939Dcm_PBcfg.h	Data prototypes for post build configuration tables

Table 4-2 Generated files

4.1.3 Include Structure

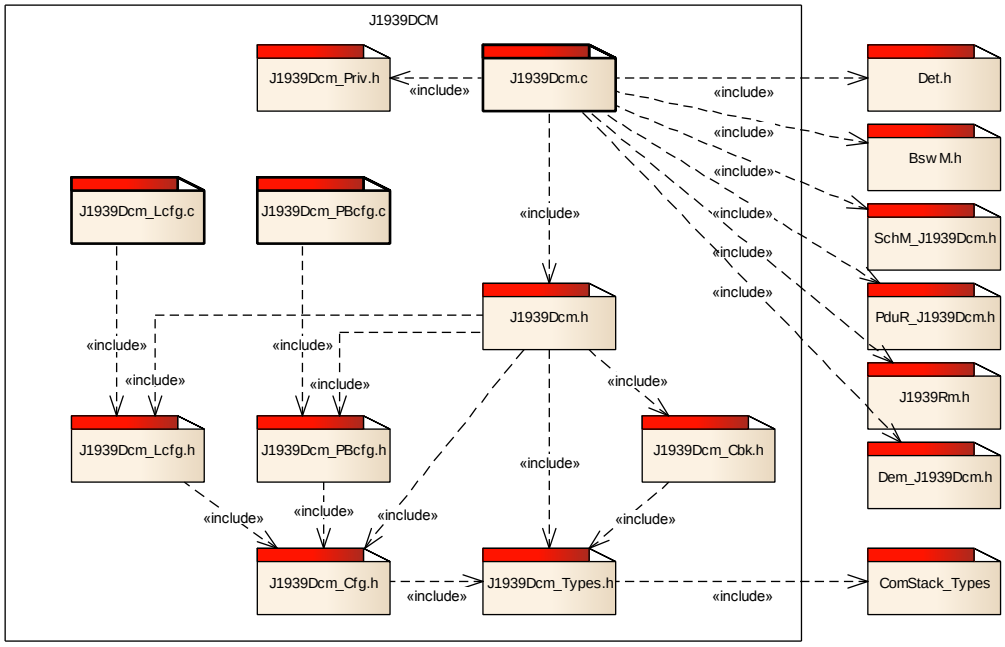

Figure 4-1 J1939DCM Include Structure

4.2 Critical Sections

As the J1939DCM handles all external requests on task level, the requests from APIs must be queued. Whenever the content of a queue is changed, the J1939DCM uses the critical section J1939DCM_EXCLUSIVE_AREA_0 to ensure data consistency.

The following APIs may use this critical section:

- J1939Dcm_MainFunction
- J1939Dcm_SetState
- J1939Dcm_RequestIndication
- J1939Dcm_RxIndication
- J1939Dcm_CopyTxData
- J1939Dcm_TpTxConfirmation
- J1939Dcm_DemTriggerOnDTCStatus
- J1939Dcm_MemResponseTransmit
- J1939Dcm_MemDataTransmit

The exclusive area can be omitted if it is guaranteed that all of these APIs are only called in the same context and cannot interrupt each other (e.g. when polling is used).

4.3 Memory Access

For memory access operations, the SAE J1939 specification defines the following diagnostic messages:

> DM14: Memory Access Request

> DM15: Memory Access Response

> DM16: Binary Data Transfer

> DM17: Boot Load Data

> DM18: Data Security

How these messages have to be used is described in Appendix C of [6] and highly depends on the needs of a specific use case. Therefore the memory access handling itself have to be done by the application, e.g. by a complex device driver.

The J1939DCM only provides interfaces (APIs and call-outs to the application) for message reception and transmission. The following table gives an overview over these interfaces

Interface / Function Pointer Definition	Type	Usage
J1939Dcm_MemRequestRxFuncType	Call-out	Reception of DM14
J1939Dcm_MemResponseTransmit	API	Transmission of DM15
J1939Dcm_MemResponseTxConfFuncType	Call-out	TxConfirmation for DM15
J1939Dcm_MemDataRxStartFuncType	Call-out	Reception of DM16 and DM18
J1939Dcm_MemDataRxCopyFuncType	Call-out	Reception of DM16 and DM18
J1939Dcm_MemDataRxIndFuncType	Call-out	Reception of DM16, DM17 and DM18
J1939Dcm_MemDataTransmit	API	Transmission of DM16 and DM18
J1939Dcm_MemDataTxConfFuncType	Call-out	TxConfirmation for DM16 and DM18

Table 4-3 Memory Access Interfaces provided by J1939DCM

The call-outs must be specified in the configuration tool and implemented in the application according to chapter 5.5.1

Also a header file with the prototypes of the implemented functions has to be provided. It is specified via the configuration parameter

- J1939DcmMemAccessHeaderFile

4.3.1 Memory Access Request (DM14)

DM14 is used by a diagnostic tool to request memory access, so the message only needs to be received.

The J1939DCM directly forwards a reception of DM14 to the application. The function which shall be used for notification is set by the configuration parameter:

- J1939DcmMemRequestIndicationFunc

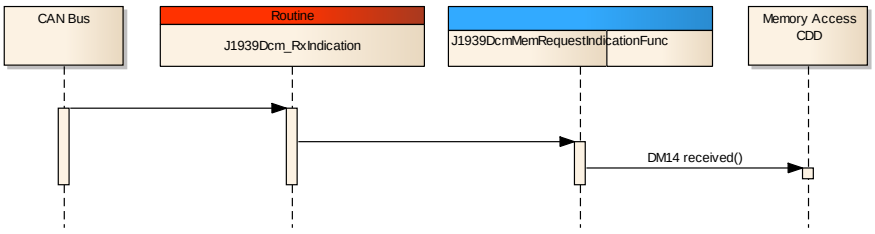

Figure 4-2 Memory Access: DM14 Reception Sequence

4.3.2 Memory Access Response (DM15)

DM15 is used by the ECU to respond to a memory access request, so the message only need to be transmitted.

Transmission of DM15 is triggered through `J1939Dcm_MemResponseTransmit()`.

The subsequent TxConfirmation is directly forwarded to the application. The function which shall be used for notification is set by the configuration parameter:

- J1939DcmMemResponseTxConfFunc

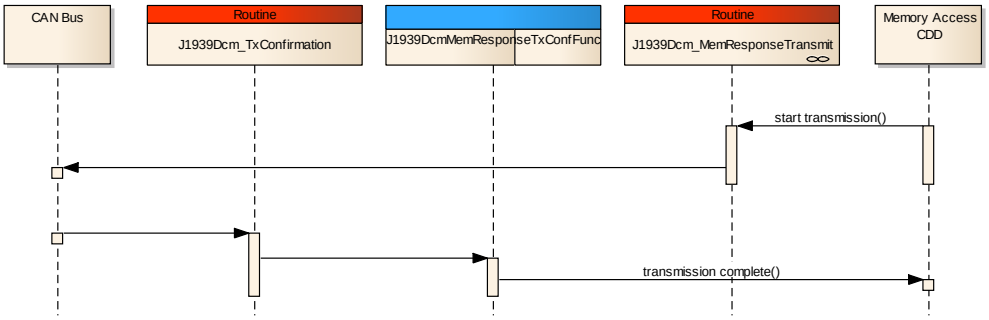

Figure 4-3 Memory Access: DM15 Transmission Sequence

4.3.3 Binary Data Transfer (DM16)

DM16 is used to transfer binary data between the ECU and the diagnostic tool. Therefore it may be received or transmitted.

On reception, the J1939DCM directly forwards all TP API calls to the application. For this, three function have to be provided which are set by the following configuration parameters:

- J1939DcmMemDataStartOfReceptionFunc
- J1939DcmMemDataCopyRxFunc
- J1939DcmMemDataRxIndicationFunc

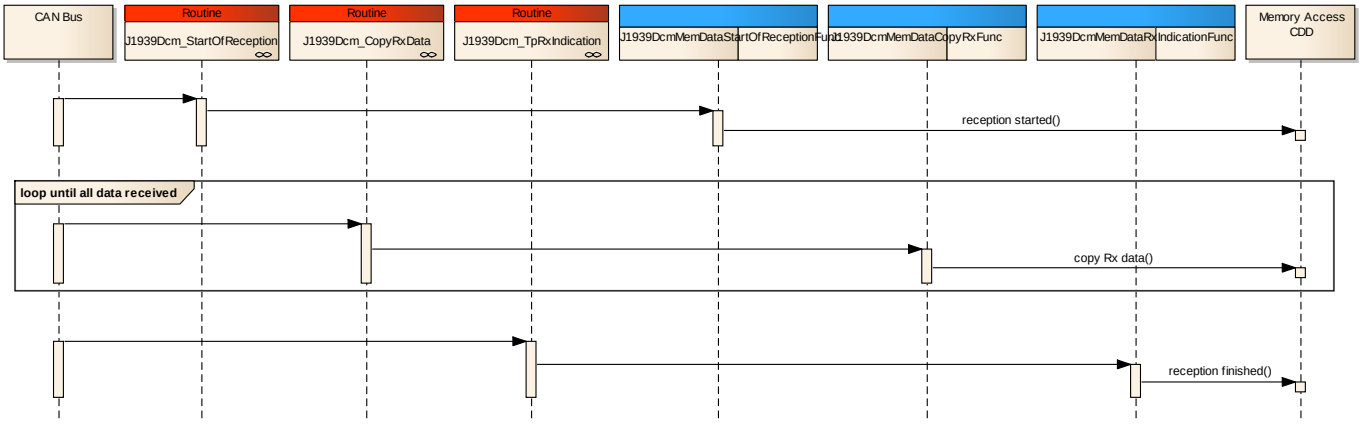

Figure 4-4 Memory Access: DM16 Reception Sequence

Transmission of DM16 is triggered through `J1939Dcm_MemDataTransmit()`. Based on the PduInfo parameter of this API, the J1939DCM will also handle the CopyTxData calls of the TP. Therefore it is important that the application locks the buffer which is passed with the transmit request until the TxConfirmation occurs.

The TxConfirmation is directly forwarded to the application. The function which shall be used for notification is set by the configuration parameter:

- J1939DcmMemDataTxConfFunc

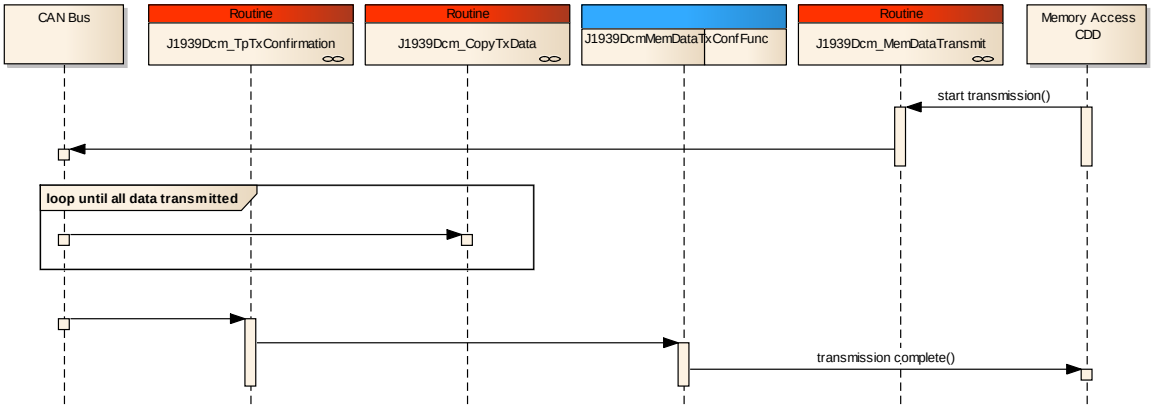

Figure 4-5 Memory Access: DM16 Transmission Sequence

Because all functions (for reception and transmission) are shared by DM16, DM17 and DM18, each API has a DataKind parameter which depends on the according message contents.

For DM16, this parameter is set to `J1939DCM_MEM_DATA_BINARY`.

4.3.4 Boot Load Data (DM17)

DM17 is used to transfer boot load data to an ECU, so the message only needs to be received.

The J1939DCM directly forwards a reception of DM17 to the application. The function which shall be used for notification is set by the configuration parameter:

- J1939DcmMemDataRxIndicationFunc

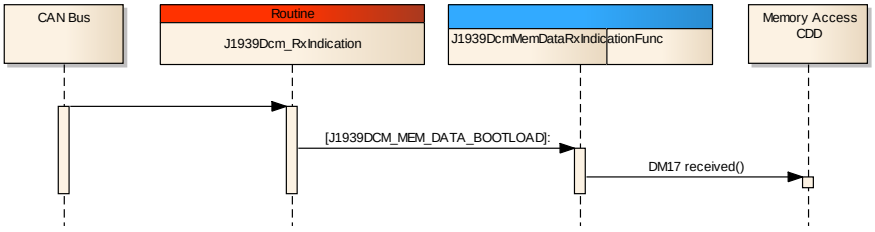

Figure 4-6 Memory Access: DM17 Reception Sequence

Because the function is shared by DM16, DM17 and DM18, it has a DataKind parameter which depends on the according message contents.

For DM17, this parameter is set to `J1939DCM_MEM_DATA_BOOTLOAD`.

4.3.5 Data Security (DM18)

DM18 is used to transfer security keys and seeds between the ECU and the diagnostic tool, so it may be received or transmitted.

Reception and transmission works the same way as for DM16 (see 4.3.3, Figure 4-4 and Figure 4-5).

For DM18, the DataKind parameter provided by each function, is set to `J1939DCM_MEM_DATA_SECURITY`.

4.4 Post-build Modes

4.4.1 Post-build Selectable

Via the post-build selectable, different variants can be flashed in an ECU at the build time where only one of which shall be activated at a specific moment. None of the flashed variants can be altered in the post-build time. For more information, please refer to [9].

4.4.2 Post-build Loadable

Post-build loadable helps to modify an existing variant in the post-build time (post-compile time). For more information, please refer to [8].

4.4.3 Post-build Loadable Selectable

The features of both modes selectable and loadable are combined. Therefore, in the run time, a variant can be selected using post-build selectable and then updated using post-build loadable. For more information please refer to 4.4.1 and 4.4.2.

4.4.4 Post-build Deletable

It helps to delete parameters that were created at the link time. This feature is considered also as a part of the post-build Loadable. For more information, please refer to [8].

4.4.5 Configuration

When post-build loadable is activated, all supported DMs will be activated as well even if they are not configured. Therefore, to avoid compiler errors, make sure to configure both containers /J1939Dcm/J1939DcmGeneral/J1939DcmMemAccessFunctions, (e.g. Figure 4-7) and /Dem/DemGeneral/DemGeneralJ1939, (e.g. Figure 4-8) in the precompile phase as if all the supported DMs are configured. Those two containers are not post-build capable. Therefore, they must be configured in the precompile phase for example as given below:

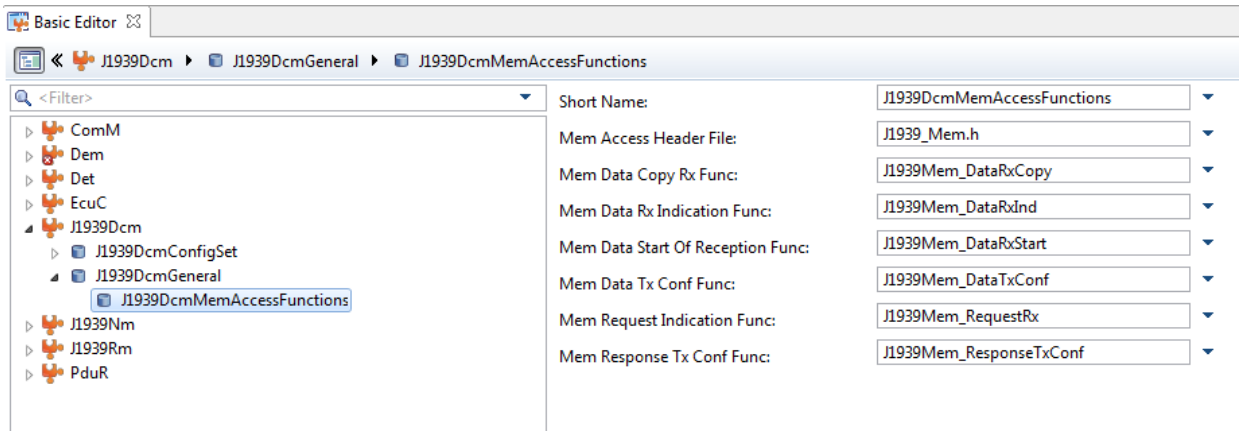

Figure 4-7 Configuration of J1939DcmMemAccessFunctions for post-build loadable

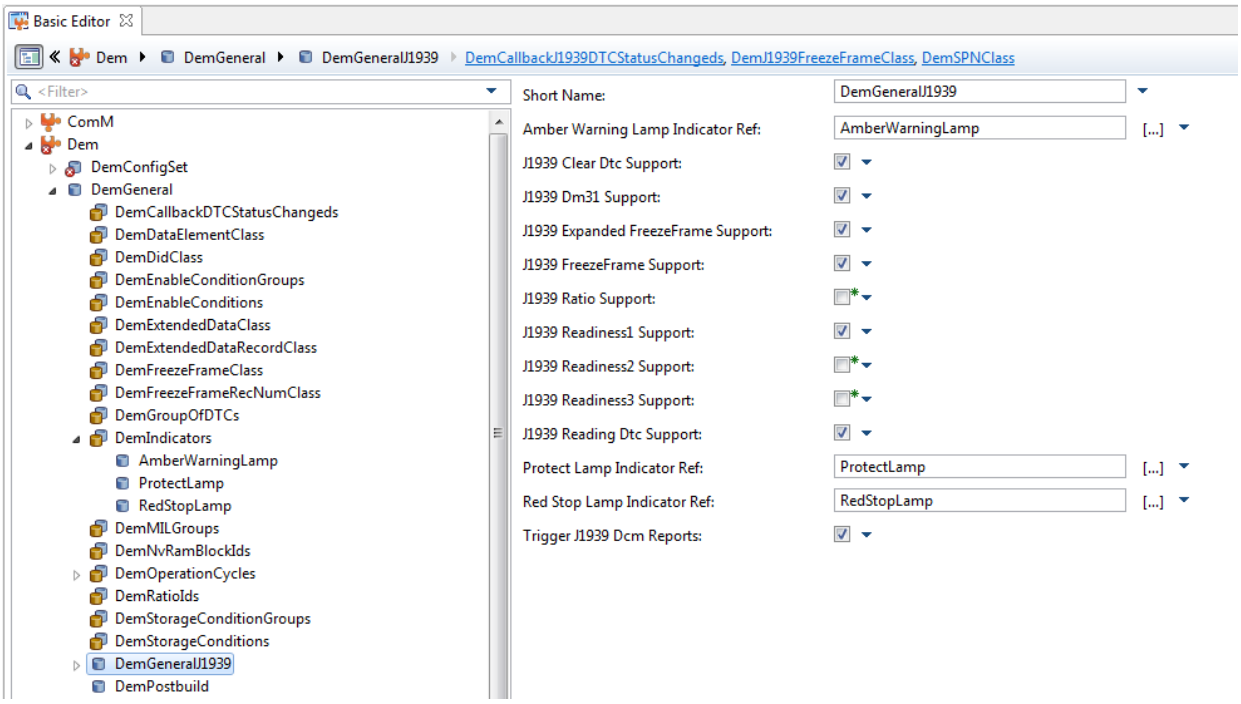

Figure 4-8 Configuration of DemGeneralJ1939 for post-build loadable

As a precaution, a validation rule is implemented in Configurator 5 to assure that the container J1939DcmMemAccessFunctions is configured when the configuration mode is set to post-build loadable.

4.4.6 Post-build Capable Containers

4.4.6.1 Diagnostic Messages

The diagnostic message container is post-build capable. Therefore, it can be added or deleted in the post-build phase. A diagnostic message container is a single instance of the diagnostic message. Each of the diagnostic message containers (diagnostic message instances) is defined in a specific node and references a specific channel. Deleting a single container of the diagnostic message does not mean the whole message is deleted. In order to delete a diagnostic message, all instances of this particular message have to be deleted from all configured nodes.

4.4.6.2 Channels

Channels are post-build capable but with some restrictions. On one hand, Channels cannot be added or deleted in the post-build phase. Therefore, it must be known from the precompile time how many channels shall be needed. Configurator 5 optimizes the generated code. If a channel is not being referenced by any diagnostic message, the channel shall not be generated.

On the other hand, the ComM channel that is referenced by J1939DCM channel (1939DcmComMChannelRef), the bus type in which a J1939DCM channel shall be activated (1939DcmBusType), and the affiliation of the channel to ISOBUS (J1939DcmIsobusChannel) are post-build capable.

4.4.6.3 Nodes

Nodes are not post-build capable. However, SPNs are post-build capable. Therefore, SPNs can be added or deleted in the post-build phase. SPNs are solely node dependent.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

The types defined by the J1939DCM are described in this chapter.

Type Name	Description	Value Range
J1939Dcm_StateType	This type represents the communication state of the J1939DCM.	0 = J1939DCM_STATE_ONLINE
		1 = J1939DCM_STATE_OFFLINE
J1939DcmBusType	Possible bus types for a J1939DCM channel. The definition is derived from the bit order in the DM13 message	0 = J1939DCM_CURRENT_NETWORK
		1 = J1939DCM_J1587
		2 = J1939DCM_J1922
		3 = J1939DCM_J1939_NETWORK_1
		4 = J1939DCM_J1939_NETWORK_2
		5 = J1939DCM_ISO9141
		6 = J1939DCM_J1850
		7 = J1939DCM_OTHER
		8 = J1939DCM_J1939_NETWORK_3
J1939Dcm_MemDataType	Data content for memory access messages to be received or transmitted	16 = J1939DCM_MEM_DATA_BINARY
		17 = J1939DCM_MEM_DATA_BOOTLOAD
		18 = J1939DCM_MEM_DATA_SECURITY

Table 5-1 Enumeration type definitions

Type Name	Description	Struct Members
J1939Dcm_MemReqParamType	Parameters of a received DM14 message. For detailed definition, see [6].	Length
		Command
		Pointer
		PointerType
		Pointer Extension
		Key
J1939Dcm_MemRespParamType	Parameters of a DM15 message to be transmitted. For detailed definition, see [6].	Length
		Status
		EDCPExtension
		ErrorIndicator
		Seed
J1939Dcm_AddressInfoType	Addressing information for reception and transmission. It contains the ComM Channel Id and the J1939Nm Addresses.	ChannelId
		SourceAddress
		DestinationAddress

Table 5-2 Structure type definitions

5.2 Services provided by J1939DCM

5.2.1 J1939Dcm_RequestIndication()

Prototype	
Parameter	
`void J1939Dcm_RequestIndication ( uint8 node, NetworkHandleType channel, uint32 requestedPgn, J1939Rm_ExtIdInfoType* extIdInfo, uint8 sourceAddress, uint8 destAddress, uint8 priority)`	
`node`	Node which received the request.
`channel`	Channel on which the request was received.
`requestedPgn`	PGN of the requested PG.
`extIdInfo`	Extended identifier bytes.
`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`	N/A
Functional Description	
Indicates reception of a Request PG.	
Particularities and Limitations	
> Service ID = 0x43 > This function is reentrant > This function is synchronous	
Call context	
> This function can be called from any context (is usually called by J1939Rm)	

Table 5-3 J1939Dcm_RequestIndication()

5.2.2 J1939Dcm_DeInit()

Prototype	
`void J1939Dcm_DeInit ( void )`	
Parameter	
`N/A`	N/A
Return code	
`void`	N/A

Functional Description
This function resets the J1939 Diagnostic Communication Manager to the uninitialized state.
Particularities and Limitations
> Service ID = 0x02 > This function is not reentrant > This function is synchronous
Call context
> This function can be called from any context

Table 5-4 J1939Dcm_DeInit()

5.2.3 J1939Dcm_GetVersionInfo()

Prototype	
`void J1939Dcm_GetVersionInfo ( Std_VersionInfoType * versioninfo )`	
Parameter	
`versioninfo`	Pointer to address where the version information of the module will be stored.
Return code	
`void`	N/A
Functional Description	
This function returns the version information of the J1939Dcm module. The version information includes: Module Id, Vendor Id and Vendor specific version numbers. The version numbers are BCD-coded.	
Particularities and Limitations	
> Service ID = 0x03 > This function is not reentrant > This function is synchronous	
Call context	
> This function can be called from any context	

Table 5-5 J1939Dcm_GetVersionInfo()

5.2.4 J1939Dcm_Init()

Prototype	
`void J1939Dcm_Init ( const J1939Dcm_ConfigType* configPtr )`	
Parameter	
`configPtr`	Pointer to selected configuration structure.

Return code	
void	N/A
Functional Description	
This function initializes the J1939 Diagnostic Communication Manager.	
Particularities and Limitations	
> Service ID = 0x01 > This function is not reentrant > This function is synchronous	
Call context	
> This function can be called from any context	

Table 5-6 J1939Dcm_Init()

5.2.5 J1939Dcm_InitMemory()

Prototype	
void **J1939Dcm_Init** (void)	
Parameter	
N/A	N/A
Return code	
void	N/A
Functional Description	
This function can be used to initialize RAM variables which are part of a VAR_INIT memory section, if this is not done by the startup code.	
Particularities and Limitations	
> Service ID = 0x80 > This function is reentrant > This function is synchronous	
Call context	
> If this function is needed, it shall be called at startup before J1939Dcm_Init	

Table 5-7 J1939Dcm_InitMemory()

5.2.6 J1939Dcm_MainFunction()

Prototype	
void **J1939Dcm_MainFunction** (void)	
Parameter	
N/A	N/A

Return code	
`void`	N/A
Functional Description	
Main function of the J1939 Diagnostic Communication Manager. Used for scheduling purposes and timeout supervision.	
Particularities and Limitations	
> Service ID = 0x04 > This function is not reentrant > This function is synchronous	
Call context	
> This function can be called from any context	

Table 5-8 J1939Dcm_MainFunction()

5.2.7 J1939Dcm_SetState()

Prototype	
`Std_ReturnType J1939Dcm_SetState ( NetworkHandleType channel, uint8 node, J1939Dcm_StateType newState )`	
Parameter	
`channel`	Channel for which the state shall be changed.
`node`	Node for which the state shall be changed.
`newState`	New state the J1939DCM shall enter; see definition of J1939Dcm_StateType for available states.
Return code	
`Std_ReturnType`	E_OK: New communication state was set. E_NOT_OK: Communication state was not changed due to wrong value in NewState or wrong initialization state of the module.
Functional Description	
Changes the communication state of J1939DCM to offline or online.	
Particularities and Limitations	
> Service ID = 0x0B > This function is reentrant > This function is synchronous	
Call context	
> This function can be called from any context	

Table 5-9 J1939Dcm_SetState()

5.2.8 J1939Dcm_MemResponseTransmit()

Prototype	
`Std_ReturnType J1939Dcm_MemResponseTransmit ( J1939Dcm_AddressInfoType* AddressInfo, J1939Dcm_MemRespParamType* MemResponseParameter) )`	
Parameter	
`AddressInfo`	Source / Destination Address and Channel of message to be transmitted.
`MemRequestInfo`	Parameters of the DM15 message to be transmitted.
Return code	
`Std_ReturnType`	> `E_OK`: transmission of DM15 has been started. > `E_NOT_OK`: transmission of DM15 failed.
Functional Description	
Triggers transmission of a Memory Access Response message (DM15). When transmission is finished, the function configured as `J1939DcmMemResponseTxConfFunc` is called.	
Particularities and Limitations	
> Service ID = 0x50 > This function is not reentrant > This function is asynchronous > This function is only available if at least one DM15 message is configured	
Call context	
> This function can be called from any context	

Table 5-10 J1939Dcm_MemResponseTransmit ()

5.2.9 J1939Dcm_MemDataTransmit()

Prototype	
`Std_ReturnType J1939Dcm_MemDataTransmit ( J1939Dcm_MemDataType DataKind, J1939Dcm_AddressInfoType* AddressInfo, PduInfoType* PduInfo)`	
Parameter	
`DataKind`	Kind of data to be transmitted; allowed values are: > `J1939DCM_MEM_DATA_BINARY` > `J1939DCM_MEM_DATA_SECURITY`
`AddressInfo`	Source / Destination Address and Channel of message to be transmitted.
`PduInfo`	Data to be transmitted, specified by pointer and length.

Return code	
Std_ReturnType	> E_OK: transmission has been started. > E_NOT_OK: transmission failed.
Functional Description	
Triggers transmission of a memory access data message (DM16 or DM18). When transmission is finished, the function configured as J1939DcmMemDataTxConfFunc is called.	
Particularities and Limitations	
> Service ID = 0x51 > This function is not reentrant > This function is asynchronous > This function is only available if at least one DM16 or DM18 message is configured	
Call context	
> This function can be called from any context	

Table 5-11 J1939Dcm_MemDataTransmit ()

5.3 Services used by J1939DCM

The following table lists services provided by other components, which are used by the J1939DCM. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
PduR	PduR_J1939Transmit
J1939Rm	J1939Rm_SendAck
Det	Det_ReportError
BswM	BswM_J1939DcmBroadcastStatus
Dem	Dem_J1939DcmClearDTC Dem_J1939DcmFirstDTCwithLampStatus Dem_J1939DcmGetNextDTCwithLampStatus Dem_J1939DcmGetNextFilteredDTC Dem_J1939DcmGetNextFreezeFrame Dem_J1939DcmSetDTCFilter Dem_J1939DcmSetFreezeFrameFilter Dem_J1939DcmReadDiagnosticReadiness1

Table 5-12 Services used by the J1939DCM

5.4 Callback Functions

This chapter describes the callback functions that are implemented by the J1939DCM and can be invoked by other modules. The prototypes of the callback functions are provided in the header file `J1939Dcm_Cbk.h` by the J1939DCM.

5.4.1 J1939Dcm_CopyRxData()

Prototype	
`BufReq_ReturnType J1939Dcm_CopyRxData ( PduIdType id, const PduInfoType* info, PduLengthType* bufferSizePtr )`	
Parameter	
`id`	Identification of the received I-PDU.
`info`	Pointer to the buffer (SduDataPtr) and its length (SduLength) containing the data to be copied by the upper layer module.
`bufferSizePtr`	Available receive buffer after data has been copied.
Return code	
`BufReq_ReturnType`	BUFREQ_OK: Data copied successfully. BUFREQ_E_NOT_OK: Data was not copied because an error occurred.
Functional Description	
This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining data is written to the position indicated by bufferSizePtr.	
Particularities and Limitations	
> Service ID = 0x05 > This function is reentrant > This function is synchronous	
Call context	
> This function can be called from any context	

Table 5-13 J1939Dcm_CopyRxData()

5.4.2 J1939Dcm_CopyTxData()

Prototype	
`BufReq_ReturnType J1939Dcm_CopyTxData ( PduIdType id, PduInfoType* info, RetryInfoType* retry, PduLengthType* availableDataPtr )`	
Parameter	
`id`	Identification of the transmitted I-PDU.
`info`	Provides the destination buffer and the number of bytes to be copied. If not enough transmit data is available, no data is copied. The transport protocol module may retry. A copy size of 0 can be used to indicate state changes in the retry parameter.
`retry`	This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems. If the retry parameter is a NULL_PTR, it indicates that the transmit data can be removed from the buffer immediately after it has been copied. Otherwise, the retry parameter must point to a valid RetryInfoType element. If TpDataState indicates TP_CONFPENDING, the previously copied data must remain in the TP buffer to be available for error recovery. TP_DATACONF indicates that all data that has been copied before this call is confirmed and can be removed from the TP buffer. Data copied by this API call is excluded and will be confirmed later. TP_DATARETRY indicates that this API call shall copy previously copied data in order to recover from an error. In this case TxTpDataCnt specifies the offset in bytes from the current data copy position.
`availableDataPtr`	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
Return code	
`BufReq_ReturnType`	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_BUSY: Request could not be fulfilled, because the required amount of Tx data is not available. The LoTp module can either retry the request with the same PduInfoPtr or treat the return value like BUFREQ_E_NOT_OK. BUFREQ_E_NOT_OK: Data has not been copied. Request failed.
Functional Description	
This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.	

Particularities and Limitations
> Service ID = 0x06
> This function is not reentrant
> This function is synchronous
Call context
> This function can be called from any context

Table 5-14 J1939Dcm_CopyTxData()

5.4.3 J1939Dcm_StartOfReception()

Prototype	
`BufReq_ReturnType J1939Dcm_StartOfReception ( PduIdType id, PduInfoType* info, PduLengthType TpSduLength, PduLengthType* bufferSizePtr )`	
Parameter	
`id`	Identification of the I-PDU.
`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. CAN ID) are appended after the payload data.
`TpSduLength`	Total length of the N-SDU to be received.
`bufferSizePtr`	Available receive buffer in the receiving module. This parameter will be used to compute the Block Size (BS) in the transport protocol module.
Return code	
`BufReq_ReturnType`	BUFREQ_OK: Connection has been accepted, bufferSizePtr indicates the available receive buffer; reception is continued. BUFREQ_E_NOT_OK: Connection has been rejected; reception is aborted, bufferSizePtr remains unchanged. BUFREQ_E_OVFL: No buffer of the required length can be provided, reception is aborted. bufferSizePtr remains unchanged.
Functional Description	
This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF).	
Particularities and Limitations	
> Service ID = 0x07	
> This function is reentrant	
> This function is synchronous	
Call context	
> This function can be called from any context	

Table 5-15 J1939Dcm_StartOfReception()

5.4.4 J1939Dcm_RxIndication()

Prototype	
`void J1939Dcm_RxIndication ( PduIdType id, const PduInfoType* PduInfoPtr )`	
Parameter	
`id`	Identification of the received I-PDU.
`PduInfoPtr`	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Return code	
`void`	N/A
Functional Description	
Called after an I-PDU has been received via the IF API, the result indicates whether the reception was successful or not.	
Particularities and Limitations	
> Service ID = 0x42 > This function is reentrant > This function is synchronous	
Call context	
> This function can be called from any context	

Table 5-16 J1939Dcm_RxIndication()

5.4.5 J1939Dcm_TpRxIndication()

Prototype	
`void J1939Dcm_TpRxIndication ( PduIdType id, Std_ReturnType result )`	
Parameter	
`id`	Identification of the received I-PDU.
`result`	Result of the reception.
Return code	
`void`	N/A
Functional Description	
Called after an I-PDU has been received via the TP API, the result indicates whether the reception was successful or not.	
Particularities and Limitations	
> Service ID = 0x08 > This function is reentrant > This function is synchronous	

Call context
> This function can be called from any context

Table 5-17 J1939Dcm_TpRxIndication()

5.4.6 J1939Dcm_TpTxConfirmation()

Prototype	
`void J1939Dcm_TpTxConfirmation ( PduIdType id, Std_ReturnType result )`	
Parameter	
`id`	Identification of the transmitted I-PDU.
`result`	Result of the transmission.
Return code	
`void`	N/A
Functional Description	
This function is called after the I-PDU has been transmitted via the TP API, the result indicates whether the transmission was successful or not.	
Particularities and Limitations	
> Service ID = 0x09 > This function is reentrant > This function is synchronous	
Call context	
> This function can be called from any context	

Table 5-18 J1939Dcm_TpTxConfirmation()

5.4.7 J1939Dcm_TxConfirmation()

Prototype	
`void J1939Dcm_TxConfirmation ( PduIdType id )`	
Parameter	
`id`	Identification of the transmitted I-PDU.
`result`	Result of the transmission.
Return code	
`void`	N/A
Functional Description	
This function is called after the I-PDU has been transmitted via the IF API to indicate that the transmission has been completed.	
Particularities and Limitations	
> Service ID = 0x09 > This function is reentrant > This function is synchronous	

Call context
> This function can be called from any context

Table 5-19 J1939Dcm_TxConfirmation()

5.4.8 J1939Dcm_DemTriggerOnDTCStatus ()

Prototype	
`Std_ReturnType J1939Dcm_DemTriggerOnDTCStatus ( uint32 DTC, uint8 DTCStatusOld, uint8 DTCStatusNew )`	
Parameter	
`DTC`	Diagnostic Trouble Code in UDS format.
`DTCStatusOld`	DTC status before change.
`DTCStatusNew`	DTC status after change.
Return code	
`Std_ReturnType`	Return value unused - only for compatibility with according RTE operation.
Functional Description	
Indicates changes of an UDS DTC status byte. This function is called by the DEM and triggers transmission of a non periodic DM1 message.	
Particularities and Limitations	
> Service ID = 0x0A > This function is not reentrant > This function is synchronous	
Call context	
> This function can be called from any context	

Table 5-20 J1939Dcm_DemTriggerOnDTCStatus ()

5.5 Configurable Interfaces

5.5.1 Memory Access Callout Functions

At its configurable interfaces the J1939DCM defines callout functions to handle the J1939 memory access message (see 4.3).

The mapping is not statically defined by the J1939DCM 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.

A header file containing the function prototypes have to be specified through the configuration parameter **J1939DcmMemAccessHeaderFile**.

To get the type definitions needed to implement the APIs, `J1939Dcm_Types.h` has to be included.

5.5.1.1 J1939DcmMemRequestIndicationFunc

Prototype	
`void` **`[J1939DcmMemRequestIndicationFunc]`** `( J1939Dcm_AddressInfoType* AddressInfo, J1939Dcm_MemReqParamType* MemRequestInfo)`	
Parameter	
`AddressInfo`	Source / Destination Address and Channel of the received message.
`MemRequestInfo`	Parameters of the received DM14 message.
Return code	
`void`	N/A
Functional Description	
Informs the application about the reception of a Memory Access Request message (DM14).	
Particularities and Limitations	
> This function have to be reentrant > At least one DM14 message must be configured	
Call context	
> This function can be called from any context	

Table 5-21 J1939DcmMemRequestIndicationFunc

5.5.1.2 J1939DcmMemResponseTxConfFunc

Prototype
`void` **`[J1939DcmMemResponseTxConfFunc]`** `(Std_ReturnType Result)`

Parameter	
Result	> E_OK: DM15 has been transmitted. > E_NOT_OK: DM15 transmission failed.
Return code	
void	N/A
Functional Description	
Informs the application that a transmission started with J1939Dcm_MemResponseTransmit() is complete.	
Particularities and Limitations	
> This function have to be reentrant > At least one DM15 message must be configured	
Call context	
> This function can be called from any context	

Table 5-22 J1939DcmMemResponseTxConfFunc

5.5.1.3 J1939DcmMemDataStartOfReceptionFunc

Prototype		
BufReq_ReturnType **[J1939DcmMemDataStartOfReceptionFunc]** (		
	J1939Dcm_MemDataType	DataKind,
	J1939Dcm_AddressInfoType*	AddressInfo,
	PduLengthType	TpSduLength,
	PduLengthType*	BufferSizePtr)

Parameter	
DataKind	Kind of data which is received; it can be: > J1939DCM_MEM_DATA_BINARY > J1939DCM_MEM_DATA_SECURITY
AddressInfo	Source / Destination Address and Channel of the received message.
TpSduLength	Overall length of message which is received.
BufferSizePtr	Size of available receive buffer in the receiving module.
Return code	
BufReq_ReturnType	> BUFREQ_OK: Connection has been accepted, bufferSizePtr indicates the available receive buffer; reception is continued. > BUFREQ_E_NOT_OK: Connection has been rejected, reception is aborted. > BUFREQ_E_OVFL: No buffer of the required length can be provided, reception is aborted.
Functional Description	
Informs the application that reception of a memory access data message has started.	

Particularities and Limitations
> This function have to be reentrant > At least one DM16 or DM18 message must be configured
Call context
> This function can be called from any context

Table 5-23 J1939DcmMemDataStartOfReceptionFunc

5.5.1.4 J1939DcmMemDataCopyRxFunc

Prototype

`BufReq_ReturnType [J1939DcmMemDataCopyRxFunc] (`
`  J1939Dcm_MemDataType DataKind,`
`  J1939Dcm_AddressInfoType* AddressInfo,`
`  const PduInfoType* PduInfo,`
`  PduLengthType* BufferSizePtr)`

Parameter	
`DataKind`	Kind of data which is received; it can be: > `J1939DCM_MEM_DATA_BINARY` > `J1939DCM_MEM_DATA_SECURITY`
`AddressInfo`	Source / Destination Address and Channel of the received message.
`PduInfo`	Pointer to the buffer (SduDataPtr) and its length (SduLength) containing the data to be copied by the upper layer module.
`BufferSizePtr`	Available receive buffer after data has been copied.
Return code	
`BufReq_ReturnType`	> `BUFREQ_OK`: Data copied successfully. > `BUFREQ_E_NOT_OK`: Data was not copied because an error occurred.
Functional Description	
Request the application to copy data of a received memory access data message.	
Particularities and Limitations	
> This function have to be reentrant > At least one DM16 or DM18 message must be configured	
Call context	
> This function can be called from any context	

Table 5-24 J1939DcmMemDataCopyRxFunc

5.5.1.5 J1939DcmMemDataRxIndicationFunc

Prototype	
`void [J1939DcmMemDataRxIndicationFunc] ( J1939Dcm_MemDataType DataKind, J1939Dcm_AddressInfoType* AddressInfo, const PduInfoType* PduInfo, Std_ReturnType Result)`	
Parameter	
`DataKind`	Kind of data which is received; it can be: > `J1939DCM_MEM_DATA_BINARY` > `J1939DCM_MEM_DATA_BOOTLOAD` > `J1939DCM_MEM_DATA_SECURITY`
`AddressInfo`	Source / Destination Address and Channel of the received message.
`PduInfo`	Pointer to the buffer (SduDataPtr) and its length (SduLength) containing the data received data. Note: this parameter is only valid when DataKind is `J1939DCM_MEM_DATA_BOOTLOAD`
`Result`	> `E_OK`: data message has been received successfully. > `E_NOT_OK`: receive error after StartOfReception has been indicated.
Return code	
`void`	N/A
Functional Description	
Notify the application that the reception of a memory access data message is complete.	
Particularities and Limitations	
> This function have to be reentrant > At least one DM16, DM17 or DM18 message must be configured	
Call context	
> This function can be called from any context	

Table 5-25 J1939DcmMemDataRxIndicationFunc

5.5.1.6 J1939DcmMemDataTxConfFunc

Prototype	
`void [J1939DcmMemDataTxConfFunc] (J1939Dcm_MemDataType DataKind, Std_ReturnType Result)`	
Parameter	
`DataKind`	Kind of data which have been transmitted; it can be: > `J1939DCM_MEM_DATA_BINARY` > `J1939DCM_MEM_DATA_SECURITY`

Result	> E_OK: data message has been transmitted. > E_NOT_OK: data message transmission failed.
Return code	
void	N/A
Functional Description	
Informs the application that a transmission started with J1939Dcm_MemDataTransmit() is complete.	
Particularities and Limitations	
> This function have to be reentrant > At least one DM16 or DM18 message must be configured	
Call context	
> This function can be called from any context	

Table 5-26 J1939DcmMemDataTxConfFunc

6 Configuration

In the J1939DCM the attributes can be configured according to/with the following methods/tools:

- Configurator 5

6.1 Configuration Variants

The J1939DCM currently supports the configuration variants

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

The configuration classes of the J1939DCM parameters depend on the supported configuration variants. For their definitions please see the J1939Dcm_bswmd.arxml file.

6.2 Configuration in Data Base

In the DBC file it can be configured which diagnostic message shall be supported by the J1939DCM. For each DM, a CAN message must be defined where the CAN-ID Type is set to **J1939 PG (ext. ID)**.

The DM to be supported is then specified through its PGN, which is part of the CAN-ID. For a complete list of PGNs, please refer to the SAE specification [6].

The DBC editor provides a dialog to define a J1939 CAN-ID, which also contains the source address and the priority.

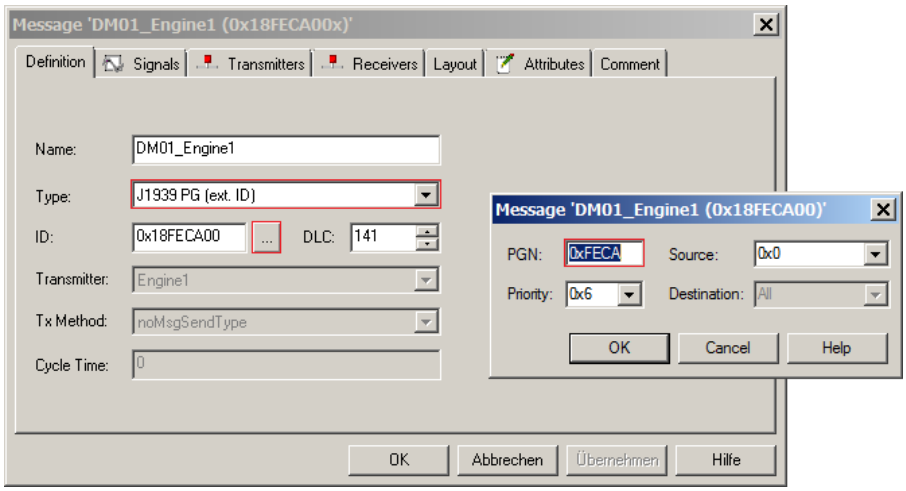

Figure 6-1 Definition of J1939 CAN identifiers in the DBC editor

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
Configurator 5	Configuration tool for MICROSAR4 components.
J1939	SAE Standard for vehicle communication, especially used in commercial vehicles like trucks and tractors.

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
BSWMD	Basis Software Module Description
CAN	Controller Area Network
CDD	Complex Device Driver
CF	Consecutive Frame
DBC	Data Base CAN
DCM	Diagnostic Communication Manager
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DM	Diagnostic Message
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
EDCP	Error Detection and/or Correction Parameter
FF	First Frame (in the context of transport protocol communication) Freeze Frame (in the context of failure storage)
ID	Identifier
IF	Interface
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
OBD	On-board diagnostics
PDU	Protocol Data Unit
PduR	PDU-Router
PG	Parameter Group
PGN	Parameter Group Number

RTE	Runtime Environment
SAE	Society of Automotive Engineers
SDU	Service Data Unit
SF	Single Frame
SPN	Suspect Parameter Number
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification
TP	Transport Protocol
UDS	Unified Diagnostic Services

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

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

www.vector.com