

MICROSAR CAN Network Management

Technical Reference

CAN NM

Authors	Markus Drescher
Version	4.18.00
Status	Released

Document Information

History

Author	Date	Version	Remarks
Oliver Hornung	2007-12-10	4.0.0	ESCAN00023633: Update Documentation for AUTOSAR Release 3
Oliver Hornung	2008-02-01	4.1.0	Added Coordination Extension; Updated configuration.
Oliver Hornung	2008-03-11	4.2.0	ESCAN00025282: Configuration changes
Oliver Hornung	2008-04-02	4.2.1	Corrected Coordination Extension
Oliver Hornung	2008-04-21	4.3.0	ESCAN00025499: Renamed Technical Reference
Oliver Hornung	2008-05-02	4.3.1	Corrected Memory Mapping
Oliver Hornung	2008-11-28	4.4.0	ESCAN00031692: Adapted Configuration and API
Oliver Hornung	2009-03-31	4.5.0	ESCAN00034277: Re-structured document; added chapter with service functions that are used by CAN NM.
Oliver Hornung	2009-07-31	4.06.00	ESCAN00036641: Documented changed behavior when requesting and releasing bus communication in state Bus Sleep. ESCAN00036642: Documented Multiple ECU Support
Oliver Hornung	2009-09-17	4.06.01	ESCAN00037225: Added hint for message length configuration
Oliver Hornung	2009-11-27	4.07.00	ESCAN00037848: Added feature description 'NM user data via Com' Clarified restart behavior (ESCAN00037081) Added note for message layout configuration (ESCAN00039378)
Oliver Hornung	2010-03-03	4.07.01	ESCAN00041215: Corrected GENy module name; adapted database attribute description for "NmType"
Oliver Hornung	2010-03-30	4.08.00	ESCAN00041850: Added QNX Support
Oliver Hornung	2010-04-30	4.09.00	ESCAN00042564: Added Extended Bus Synchronization
Oliver Hornung	2010-06-23	4.10.00	ESCAN00043403: Added 'Check Limp Home' API description
Oliver Hornung	2010-07-21	4.11.00	ESCAN00043887: Added 'Gateway Extension Support' description
Daniel Hof	2010-08-11	4.12.00	Added descriptions for: ESCAN00043787: Active Wake-up Bit ESCAN00043772: Car Wake-up ESCAN00043794: Partial Networking ESCAN00043785: Fast Wake-up

Oliver Hornung	2011-02-10	4.13.00	<p>ESCAN00045764: Adapted DET and DEM error description</p> <p>ESCAN00045879 Reworked Partial Networking description</p> <p>ESCAN00046097: Added description for usage of PduInfoType</p> <p>ESCAN00046915 Adapted description for Fast Wake-up (Immediate Nm Transmissions)</p> <p>ESCAN00047362 Added descriptions of critical section codes</p> <p>ESCAN00047497 Added description for usage in multiple configuration setups</p>
Markus Drescher	2011-05-26	4.13.01	ESCAN00051145: Added description for 'Dem2Det Enabled' switch and updated Error Handling
Markus Drescher	2011-07-04	4.13.02	<p>ESCAN00051912: Adapted description for state transitions from Prepare Bus Sleep to Network Mode, adapted CanNm_PassiveStartUp API description</p> <p>ESCAN00051972: Updated Include Structure figure</p>
Markus Drescher	2011-06-30	4.14.00	<p>ESCAN00048297: Added description for Callbacks for Partial Networking</p> <p>ESCAN00051150: Adapted description of 'Usage of PduInfoType'</p>
Markus Drescher	2012-01-12	4.15.00	<p>ESCAN00053603: Removed VStdLib and DrvCan critical section handling</p> <p>ESCAN00054369: Adapted description of 'Msg Timeout Time'</p> <p>ESCAN00054839: Added description for Support for RX PDUs with different lengths</p>
Markus Drescher	2012-06-20	4.16.00	ESCAN00059530: Adapted API descriptions for availability of the functions in chapters 5.5.2.5, 5.5.2.7.1, 5.5.2.8.1, 5.5.2.9.1; adapted chapter 5.6
Markus Drescher	2012-09-06	4.17.00	<p>ESCAN00059456: Updated version of reference document [3]</p> <p>ESCAN00059750: Added descriptions of the values written by CanNm_GetState (chapter 5.5.2.2)</p> <p>ESCAN00059987: Added caution box to chapter 3.23 about the 'Com User Data Enabled' configuration setting</p> <p>ESCAN00060186: Reworked chapter 3.5.5</p> <p>ESCAN00060200: Updated chapters 3.13 and 6.3.2, created chapter 6.3.2.1.</p> <p>ESCAN00061255: Fixed name of CanNm_TxConfirmation (chapter 5.7.1.2)</p>

Markus Drescher	2013-05-08	4.18.00	<p>ESCAN00061017: Adapted chapter 6.3.1</p> <p>ESCAN00061692: Adapted Multiple ECU Use Case descriptions for NM User Data via Com (chapters 3.13, 3.15.1)</p> <p>ESCAN00065299: Improved send behavior descriptions in chapter 3.5.3 and Immediate Nm Transmission feature descriptions in chapter 3.20 and 3.21</p> <p>ESCAN00065569: Adapted chapter 3.23.5 and 6.3.2</p> <p>ESCAN00065570: Extended chapter 3.15</p> <p>ESCAN00065637: Adapted chapter 5.5.1.1</p> <p>ESCAN00067267: Added chapter 'Component History', adapted chapters 2 and 3, merged chapter 7 with chapter 3, swapped chapters 5 and 6, removed 'Compiler Abstraction and Memory Mapping' from chapter 4, various improvements</p> <p>ESCAN00067268: Replaced Nm_PrepareBusSleep by Nm_PrepareBusSleepMode</p>
-----------------	------------	---------	--

Table 1-1 History of the Document

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	SRS AUTOSAR Network Management	2.1.0
[2]	AUTOSAR	SWS NM Interface	1.2.0
[3]	AUTOSAR	SWS CAN NM	3.4.0
[4]	AUTOSAR	SWS Development Error Tracer	2.2.2
[5]	AUTOSAR	SWS Diagnostic Event Manager	3.2.0
[6]	AUTOSAR	SWS CAN Interface	3.3.0
[7]	AUTOSAR	SWS BSW Scheduler	1.2.0
[8]	Vector	AN-AND-1-108 Glossary of CAN Protocol Terminology http://www.vector-informatik.de	1.0.0
[9]	Vector	Technical Reference Identity Manager	≥ 1.1.5
[10]	Vector	Technical Reference MICROSAR PDU Router	≥ 3.12.02
[11]	Vector	AN-ISC-8-1118 MICROSAR BSW Compatibility Check	1.0.0
[12]	AUTOSAR	AUTOSAR_BasicSoftwareModules.pdf	1.5.0

Table 1-2 Reference Documents



Please note

We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one

	specified in your questionnaire, Vector's release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.
--	---

Contents

1	Component History	12
2	Introduction	13
2.1	Naming Conventions	13
2.2	Architecture Overview	13
2.2.1	Architecture of AUTOSAR Software	13
2.2.2	Architecture of AUTOSAR Network Management.....	14
3	Functional Description	16
3.1	Overview of the Functional Scope	16
3.1.1	Deviations Against AUTOSAR 3.2.2	16
3.1.1.1	Initialization	16
3.1.1.2	PDU ID Used as Argument for CanNm_TxConfirmation	17
3.1.1.3	Dependencies between Configuration Parameters	17
3.1.1.4	Coordinator Synchronization Support	17
3.1.1.5	NM User Data via Com.....	17
3.1.1.6	Active Wake-up Bit Support.....	17
3.1.1.7	Redundant BSWMD Parameters	17
3.1.1.8	Unsupported Features.....	18
3.1.2	Additions/ Extensions	18
3.1.2.1	Link-Time Support for Channels	18
3.1.2.2	Single Channel Optimization	18
3.1.2.3	Memory Initialization.....	18
3.1.2.4	Error Reporting to Diagnostic Event Manager	18
3.1.2.5	Disable Transmission Error Reporting	18
3.1.2.6	Limp Home Indication.....	19
3.1.2.7	Extended Remote Sleep Indication.....	19
3.1.2.8	Multiple ECU Support.....	19
3.1.2.9	QNX Support.....	19
3.1.2.10	Extended Bus Synchronization	19
3.1.2.11	Gateway Extension Support	19
3.1.2.12	Dem2Det Enabled	19
3.1.2.13	Message Reception with Different PDU Lengths	19
3.1.2.14	Deactivation of Partial Networking during Run-Time	19
3.1.2.15	Features Provided Beyond the AUTOSAR Standard	20
3.1.3	Limitations.....	20
3.1.3.1	Ranges of Timers	20
3.2	Network Management Mechanism	21
3.3	Passive Mode.....	22

3.4	Operation Modes and States	22
3.4.1	Network Mode	23
3.4.1.1	Repeat Message State	24
3.4.1.2	Normal Operation State	24
3.4.1.3	Ready Sleep State	24
3.4.2	Prepare Bus-Sleep Mode	24
3.4.3	Bus-Sleep Mode	25
3.4.4	Wake-up Registration	25
3.4.5	User Data Handling	26
3.5	Network Management Message Transmission and Reception	26
3.5.1	AUTOSAR CAN Interface	26
3.5.2	PDU Message Layout	26
3.5.3	Message Transmissions	27
3.5.4	Bus Load Reduction	28
3.5.5	Support for RX PDUs with Different Lengths	28
3.6	Node Detection	29
3.7	NM PDU Receive Indication	29
3.8	Communication Control	29
3.9	Gateway Functionality	30
3.9.1	Remote Sleep Indication and Cancellation	30
3.9.2	Extended Remote Sleep Indication and Cancellation	30
3.9.3	Bus Synchronization	30
3.10	Limp Home Indication and Cancellation	30
3.11	Coordinator Synchronization Support	31
3.12	Error Handling	32
3.12.1	Development Error Detection	32
3.12.1.1	Det_ReportError	32
3.12.1.2	Error Codes	32
3.12.2	Diagnostic Event Manager	32
3.12.2.1	Dem_ReportErrorStatus	33
3.12.2.2	Error Codes	33
3.13	Multiple ECU Support	33
3.14	Multiple Configuration Support	34
3.15	NM User Data via Com	34
3.15.1	Configuration Preconditions in an AUTOSAR ECU Configuration	35
3.15.1.1	Caveats for Multiple ECU Support	37
3.15.2	Configuration Preconditions in a DBC Based Configuration	37
3.16	QNX Support	37
3.17	Extended Bus Synchronization	38
3.18	Gateway Extension Support	38
3.19	Active Wake-up Handling	39

3.20	Immediate Nm Transmissions	39
3.21	Immediate Restart Enabled	40
3.22	Car Wake-up	41
3.22.1	Rx-Path	41
3.22.2	Tx-Path	41
3.23	Partial Networking	42
3.23.1	Availability of Partial Network Request Information	42
3.23.2	Transmission of the CRI Bit in the NM User Data	42
3.23.3	Filter Algorithm for Received NM Messages	42
3.23.4	Aggregation of Requested Partial Networks	43
3.23.5	Spontaneous Sending of NM Messages	43
3.23.5.1	Using Com Transmission on Change Mechanism	44
3.23.5.2	Using NM Request and Immediate Nm Transmission	44
3.23.6	Deactivation of Partial Networking during Run-Time	44
3.24	Usage of PduInfoType	44
4	Integration	45
4.1	Files	45
4.2	Include Structure	46
4.3	Version Changes	46
4.4	Initialization	46
4.5	Main Functions	47
4.6	Critical Sections	48
4.7	Critical Section Codes	48
5	API Description	50
5.1	API Categories	50
5.2	Data Types	50
5.3	Global Variables	50
5.4	Global Constants	50
5.4.1	AUTOSAR Specification Version	50
5.4.2	Component Versions	51
5.4.3	Vendor and Module ID	51
5.5	Services Provided by CAN NM	52
5.5.1	Administrative Functions	52
5.5.1.1	CanNm_Init: Initialization of CAN NM	52
5.5.1.2	CanNm_MainFunction: Channel Specific Main Functions of CAN NM	53
5.5.2	Service Functions of CAN NM	54
5.5.2.1	CanNm_GetVersionInfo: Version Information API	54
5.5.2.2	CanNm_GetState: Get the State of the Network Management	55
5.5.2.3	CanNm_PassiveStartUp: Wake up Network Management	56

5.5.2.4	Wake-up Registration	57
5.5.2.4.1	CanNm_NetworkRequest: Request the Network	57
5.5.2.4.2	CanNm_NetworkRelease: Release the Network.....	58
5.5.2.5	Communication Control Service	59
5.5.2.5.1	CanNm_DisableCommunication: Disable NM Message Transmission	59
5.5.2.5.2	CanNm_EnableCommunication: Enable NM Message Transmission	60
5.5.2.6	User Data Handling	61
5.5.2.6.1	CanNm_SetUserData: Set User Data.....	61
5.5.2.6.2	CanNm_GetUserData: Get User Data	62
5.5.2.6.3	CanNm_GetPduData: Get NM Pdu Data	63
5.5.2.7	Node Detection.....	64
5.5.2.7.1	CanNm_RepeatMessageRequest: Set Repeat Message Request Bit.....	64
5.5.2.7.2	CanNm_GetNodeIdentifier: Get Source Node Identifier	65
5.5.2.7.3	CanNm_GetLocalNodeIdentifier: Get Local Source Node Identifier	66
5.5.2.8	Bus Synchronization.....	67
5.5.2.8.1	CanNm_RequestBusSynchronization: Synchronization of Networks	67
5.5.2.9	Remote Sleep Indication	68
5.5.2.9.1	CanNm_CheckRemoteSleepIndication: Check for Remote Sleep Indication	68
5.5.2.10	NM Message Transmission Request	69
5.5.2.10.1	CanNm_Transmit: Spontaneous NM Message Transmission	69
5.5.2.11	Partial Networking	70
5.5.2.11.1	CanNm_ConfirmPnAvailability: Notification for activating the PN filter functionality.....	70
5.5.3	Vector extensions.....	71
5.5.3.1	CanNm_InitMemory: Memory Initialization	71
5.5.3.2	CanNm_SetCoordBits: Set Coordination Bits in the CBV	72
5.5.3.3	CanNm_CheckLimpHomeIndication: Check the Limp Home Status	73
5.5.3.4	CanNm_SetDiagGwReqId: Diagnostic Gateway ID Request.....	74
5.5.3.5	CanNm_DisablePnActivationState: Disable Partial Networking	74
5.6	Services Used by CAN NM	75
5.7	Callbacks Provided by CAN NM	75
5.7.1	Callback Functions from CAN Interface.....	75
5.7.1.1	CanNm_RxIndication: NM Message Indication.....	75
5.7.1.2	CanNm_TxConfirmation: NM Message Confirmation Function.....	77
5.8	Callback Functions used by CAN NM.....	77
5.8.1	Service Callback Functions of CAN NM	77

5.8.2	Additional Service Callback Functions of CAN NM	78
5.8.2.1	Callbacks for Limp Home Indication	78
5.8.2.2	Callbacks for Coordination Extension	78
5.8.2.3	Callbacks for Gateway Extension Support.....	78
5.8.2.4	Callbacks for Partial Networking	78
5.8.2.5	Generator Compatibility Error	79
6	Configuration.....	80
6.1	Configuration Variants	80
6.2	Configuration in Data Base.....	80
6.3	Configuration with GENy	82
6.3.1	Component Configuration	84
6.3.2	Channel Configuration.....	90
6.3.2.1	Identity Specific Channel Configuration	94
6.3.3	Timing Restrictions.....	96
6.3.4	NM Message Structure Configuration Notes	96
7	Glossary and Abbreviations	97
7.1	Glossary.....	97
7.2	Abbreviations	97
8	Contact.....	99

Illustrations

Figure 2-1	Architecture of AUTOSAR Stack.....	14
Figure 2-2	Interface to Adjacent Modules of the CAN NM	15
Figure 3-1	State Diagram of CAN NM from SWS CAN NM [3]	22
Figure 3-2	Usual Behavior of NM Transmissions when Repeat Message is entered..	27
Figure 3-3	Immediate Transmission due to Signal Change inside User Data I-PDU ..	35
Figure 3-4	Immediate Nm Transmissions.....	39
Figure 3-5	Behavior for NM Transmissions if Immediate Restart Enabled is ON.....	41
Figure 4-1	Include Structure of CAN NM.....	46
Figure 6-1	Component Selection in GENy	83
Figure 6-2	Component Configuration CAN NM	84
Figure 6-3	Channel Configuration CAN NM	91
Figure 6-4	Identity-Specific Channel Configuration CAN NM	95

Tables

Table 1-1	History of the Document	4
Table 1-2	Reference Documents	4
Table 1-1	Component history.....	12
Table 2-1	Naming Conventions	13
Table 3-1	Supported AUTOSAR standard conform features	16
Table 3-2	Not supported AUTOSAR standard conform features	18

Table 3-3	Features provided beyond the AUTOSAR standard.....	20
Table 3-4	PDU NM Message Layout	26
Table 3-5	NM Coordination Bits.....	31
Table 3-6	DET Error Codes of CAN NM	32
Table 3-7	DEM Error Codes of CAN NM	33
Table 3-8	Configuration Precondition Overview for AUTOSAR ECU Configurations ..	36
Table 4-1	CAN NM Integration Files	45
Table 4-2	Critical Section Codes	49
Table 5-1	Data Types	50
Table 5-2	Specification Version API Data	51
Table 5-3	Component Version API Data	51
Table 5-4	Vendor/Module ID.....	51
Table 5-5	Services Used by the CAN NM.....	75
Table 6-1	Database Attributes for CAN NM	82
Table 6-2	Component Configuration CAN NM	89
Table 6-3	Channel Configuration CAN NM	94
Table 6-4	Identity-Specific Channel Configuration CAN NM	95
Table 6-5	CAN NM Timing Restrictions	96
Table 7-1	Glossary	97
Table 7-2	Abbreviations.....	98

1 Component History

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

Component Version	New Features
1.00.00	Creation for AUTOSAR 1.0
3.00.00	Adaption for AUTOSAR 2.1
4.00.00	Adaption for AUTOSAR 3.0
4.01.00	Added Coordinator Synchronization Support
4.02.00	Support variant link-time for the number of channels Extended Coordinator Synchronization Support
4.09.00	Multiple ECU Support
4.12.00	Added 'User Data via Com' Feature
4.14.00	Added 'QNX Support' Feature
4.15.00	Added 'Extended Bus Synchronization' Feature
4.16.00	Added 'Limp Home Indication and Cancellation' Feature
4.17.00	Added 'Gateway Extension Support' Feature
4.19.00	Added 'Partial Networking' Feature Added 'Car Wakeup' Feature Added 'Active Wakeup Bit' Feature Added 'Immediate Transmissions' Feature
4.22.00	Adapted 'Msg Timeout' Handling Added Support for RX PDUs with different lengths
4.26.00	Added 'Disable Partial Networking During Initialization' Feature

Table 1-1 Component history

2 Introduction

This document describes the concept, features, API and the configuration of the AUTOSAR CAN Network Management as specified in [3]. Also the integration of the Network Management into the Vector CANbedded stack is covered by this document. The FlexRay Network Management is not covered by this document.

Supported AUTOSAR Release*:	3	
Supported Configuration Variants:	pre-compile, link-time, post-build	
Vendor ID:	CANNM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CANNM_MODULE_ID	31 decimal (according to ref. [12])

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

Please note that in this document the term Application is not used strictly for the user software but also for any higher software layer, like e.g. a Communication Control Layer. Therefore, Application refers to any of the software components using the CAN NM.

For further information please also refer to the AUTOSAR SWS specifications referenced in the section 'Reference Documents'.

2.1 Naming Conventions

The names of the service functions provided by the NM Interface and CAN NM always start with a prefix that denominates the software component where the service is located. E.g. a service that starts with 'CanNm_' is implemented within the CAN NM.

Naming conventions

Nm_	Services of NM Interface.
CanNm_	Services of CAN NM.
Det_	Services of Development Error Tracer.
Dem_	Services of Diagnostic Event Manager.

Table 2-1 Naming Conventions

Nodes which are configured to be passive will be also referred to as passive nodes. Accordingly nodes that are not passive will be termed as active nodes.

2.2 Architecture Overview

2.2.1 Architecture of AUTOSAR Software

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

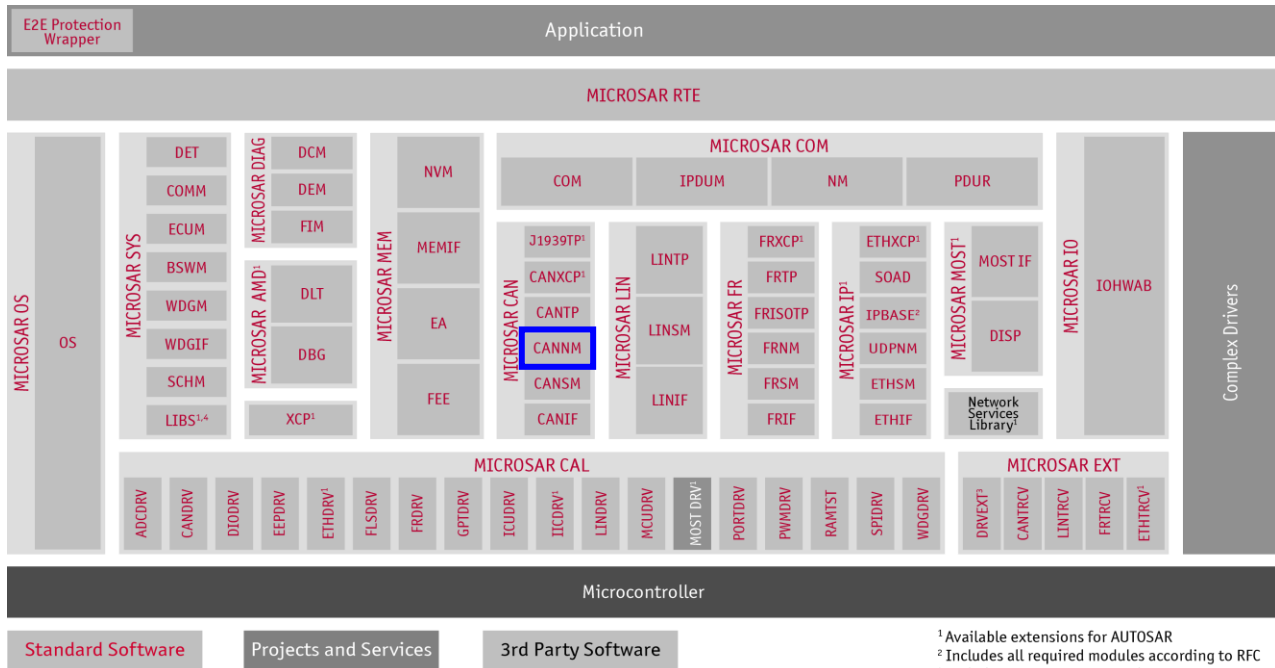


Figure 2-1 Architecture of AUTOSAR Stack

2.2.2 Architecture of AUTOSAR Network Management

The AUTOSAR Network Management consists of three modules:

- > NM Interface¹
- > CAN NM
- > FlexRay NM¹

The NM Interface schedules function calls from the application to the respective module for each channel, e.g. for a CAN channel the corresponding CAN NM function will be called. CAN NM exclusively interacts with the NM Interface.

The communication bus specific functionality is incorporated in the corresponding bus-specific NM. The CAN-specific part implements the network management algorithm and is responsible for the transmission of NM messages on the communication bus by interacting with the AUTOSAR CAN Interface.

The next figure shows the interfaces to adjacent modules of the CAN NM. These interfaces are described in chapter 5, 'API Description'.

¹ Not covered by this document.

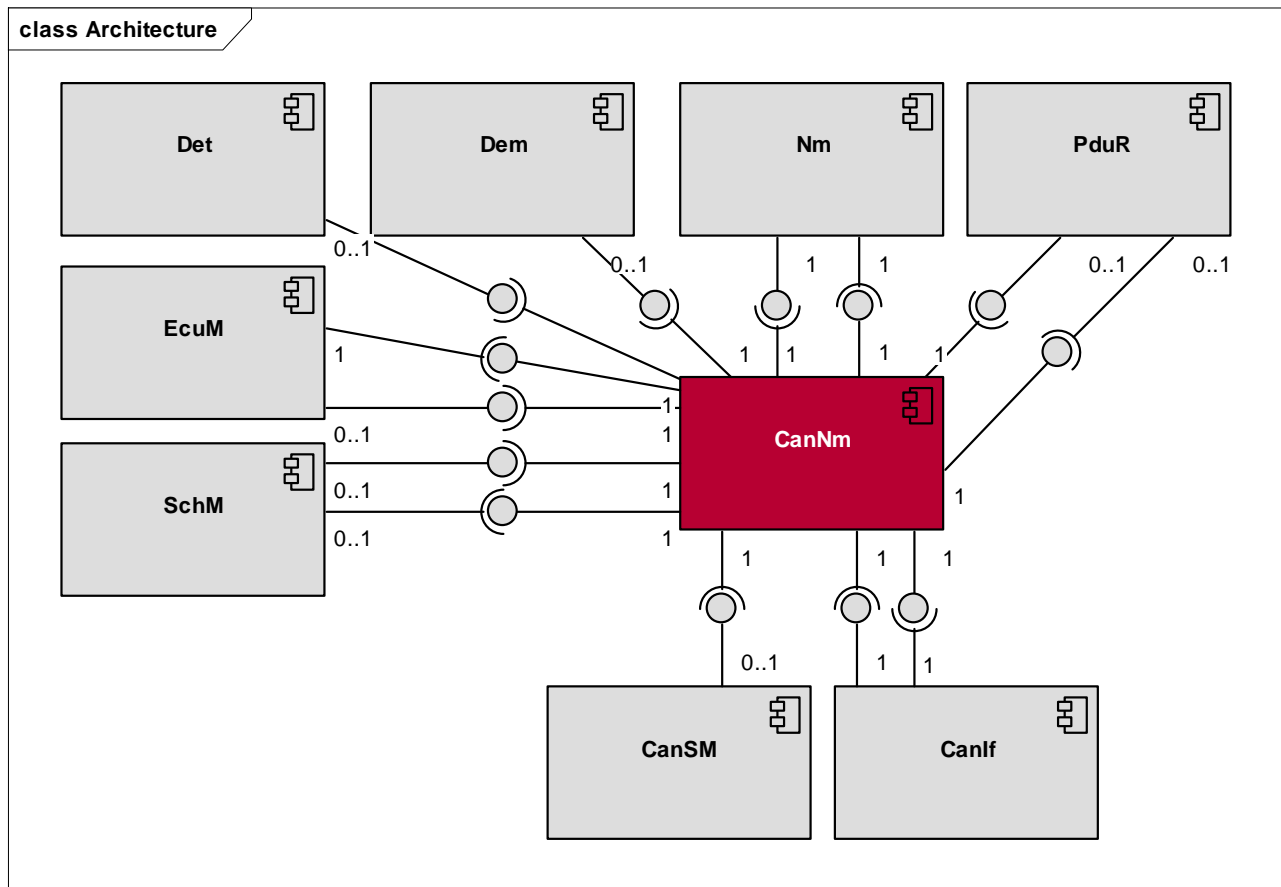


Figure 2-2 Interface to Adjacent Modules of the CAN NM

Applications do not access the services of the BSW modules directly. They use the service ports provided by the BSW modules via the RTE. Since the CAN NM has no service ports, the CAN NM cannot be accessed via RTE by the application.

3 Functional Description

3.1 Overview of the Functional Scope

The Network Management is a network comprehensive protocol that provides services for the organization of the network. It is a decentralized and direct network management. That means that every ECU transmits a special network management message, which is reserved for the network management only.

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

The AUTOSAR standard functionality is specified in [3], 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

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

- > Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [3] are supported:

Supported AUTOSAR Standard Conform Features
Controlled transition of all ECUs to bus-sleep mode and vice versa.
User Data Handling
Node Detection
Remote Sleep Indication
Passive Mode Support
Multiple ECU Support
Multiple Configuration Support
NM User Data via Com
Active Wake-up Bit
Immediate Nm Transmissions
Car Wake-up
Partial Networking
Usage of PduInfoType

Table 3-1 Supported AUTOSAR standard conform features

3.1.1 Deviations Against AUTOSAR 3.2.2

3.1.1.1 Initialization

If RAM is not implicitly initialized by the start-up code of the project, the function `CanNm_InitMemory` has to be called to ensure that all relevant variables are initialized

correctly (e.g. the state machine for checking if the module is initialized) as specified by AUTOSAR.

3.1.1.2 PDU ID Used as Argument for CanNm_TxConfirmation

The CAN Interface specification [6] and the CAN NM specification [3] have different interpretations belonging the value of the API parameter `canNmTxPduId` of the function `CanNm_TxConfirmation`. To avoid inconsistencies this implementation adopts the interpretation of the CAN Interface specification where this parameter refers not to the NM channel handle but to the PDU ID.

3.1.1.3 Dependencies between Configuration Parameters

Following additional dependencies between configuration parameters are added to avoid bad configurations:

- > `CANNM_COM_CONTROL_ENABLED` must be set to `STD_OFF` for passive nodes.
- > `CANNM_NODE_DETECTION_ENABLED` must be set to `STD_OFF` for passive nodes.

3.1.1.4 Coordinator Synchronization Support

There are APIs of the Coordinator Synchronization Support that are not supported. This includes:

- > The usage of `Nm_CoordReadyToSleepIndication` (indication is done through `Nm_ActiveCoordIndication` instead)
- > The provision of the `CanNm_SetSleepReadyBit` function (`CanNm_SetCoordBits` may be used instead to indicate the value of the Sleep Ready bit)
- > The BSWMD parameter `CanNmCoordinatorSyncSupport` does not exist. Instead, the preprocessor define in the code depends on the value of the corresponding parameter in the NM Interface.

3.1.1.5 NM User Data via Com

The BSWMD parameter `CanNmComUserDataSupport` is called `CanNmComUserDataEnabled` instead.

3.1.1.6 Active Wake-up Bit Support

The BSWMD parameter `CanNmActiveWakeupBitEnabled` is not located inside the `CanNmGlobalConfig` container but inside `CanNmChannelConfig` container.

3.1.1.7 Redundant BSWMD Parameters

The following BSWMD parameters are redundant to other parameters, but are nevertheless mandatory since they were mandatory in previous versions of [3]:

- > The multiplicity of the `CanNmNumberOfChannels` parameter is 1..1, but should be 0..1 instead according to [3].
- > The multiplicity of the `CanNmUserDataLength2` parameter is 1..1, but should be 0..1 instead according to [3].

² Typo is intended (parameter name according to AUTOSAR)

3.1.1.8 Unsupported Features

The following features specified in [3] are not supported:

Not Supported AUTOSAR Standard Conform Features
-

Table 3-2 Not supported AUTOSAR standard conform features

3.1.2 Additions/ Extensions

The following extensions of the CAN NM software specifications ([3]) are available within the Network Management embedded software components. If required, the extensions have to be enabled during configuration.

3.1.2.1 Link-Time Support for Channels

In [3] the number of channels is a pre-compile parameter and cannot be changed at link-time. For a better library support it must be possible to change the number of channels at link-time. Therefore the implementation of the CAN NM supports link-time changes of the channels.

3.1.2.2 Single Channel Optimization

For single channel systems it is possible to optimize the source code for saving precious resources (ROM, RAM and CPU load). This optimization is only possible when source code is available. For the configuration please refer to chapter 6.3.1 'Component Configuration'.

Please note that single channel optimization can only be enabled in pre-compile configurations. The value must not be changed at link-time. If single-channel optimization is enabled and the configuration variant is link-time the number of CAN NM channels must not be changed.

3.1.2.3 Memory Initialization

AUTOSAR expects the start-up code to automatically initialize RAM. Due to not every startup code of embedded targets reinitializes all variables correctly it may happen that the state of a variable may be not initialized, when this should be the case. To avoid this problem the Vector AUTOSAR NM provides additional functions to initialize the relevant variables of the CAN NM.

Refer also to chapter 5.5.3.1 'CanNm_InitMemory: Memory Initialization'.

3.1.2.4 Error Reporting to Diagnostic Event Manager

According to the AUTOSAR specifications error reporting to the Diagnostic Event Manager is mandatory. If the AUTOSAR NM is used in environments without a Diagnostic Event Manager it should be possible to deactivate the error reporting. Therefore the switch `CANNM_PROD_ERROR_DETECT` has been added (refer also to chapter 6.3.1 'Component Configuration').

3.1.2.5 Disable Transmission Error Reporting

The error reporting for the following transmission errors can be disabled (refer also to chapter 6.3.1 'Component Configuration'):

> `CANNM_E_DEV_NETWORK_TIMEOUT (DET)`

- > CANNM_E_NETWORK_TIMEOUT (DEM)
- > CANNM_E_CANIF_TRANSMIT_ERROR (DEM)

3.1.2.6 Limp Home Indication

This additional feature allows the detection and cancellation of Limp Home, i.e. if no NM message could be received and transmitted for a certain amount of time (see also chapter 3.10 'Limp Home Indication and Cancellation'). Please note that this feature is OEM-specific and not available by default. If available it can be activated or deactivated in the configuration. Also refer to chapter 6 'Configuration'.

3.1.2.7 Extended Remote Sleep Indication

This additional feature allows checking the remote sleep state also in the Repeat Message state (see also chapter 3.9.2 'Extended Remote Sleep Indication and Cancellation'). Please note that this feature is OEM-specific and not available by default. If available it can be activated or deactivated in the configuration. Also refer to chapter 6 'Configuration'.

3.1.2.8 Multiple ECU Support

The CAN NM supports Multiple ECU configuration when the configuration in GENy is done with an AUTOSAR ECU configuration file. Refer to chapter 3.13 'Multiple ECU Support' for more information.

3.1.2.9 QNX Support

The CAN NM supports context switch. Refer to chapter 3.16 'QNX Support' for more information.

3.1.2.10 Extended Bus Synchronization

The CAN NM supports the possibility to enable the API 'CanNm_RequestBusSynchronization' additionally. More details can be found in chapter 3.17 'Extended Bus Synchronization'.

3.1.2.11 Gateway Extension Support

The CAN NM provides support for the 'Gateway Extension' functionality in the NM Interface. Refer to chapter 3.18 'Gateway Extension Support' for further information.

3.1.2.12 Dem2Det Enabled

The CAN NM provides support for the different Dem/Det error requirements specified in AUTOSAR Release 3.0, 3.1 and 3.2. Refer to chapter 3.12 'Error Handling' for further information.

3.1.2.13 Message Reception with Different PDU Lengths

The CAN NM supports the reception of PDUs with different lengths. Refer to chapter 3.5.5 'Support for RX PDUs with Different Lengths' for further information.

3.1.2.14 Deactivation of Partial Networking during Run-Time

The FlexRay NM supports the deactivation of Partial Networking during run-time. Refer to chapter 3.23.6 and 5.4.3.2 for details.

3.1.2.15 Features Provided Beyond the AUTOSAR Standard

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
Link-Time Support for Channels
Single Channel Optimization
Memory Initialization
Error Reporting to the Diagnostic Event Manager
Disable Transmission Error Reporting
Limp Home Indication
Extended Remote Sleep Indication
Multiple ECU Support
QNX Support
Extended Bus Synchronization
Gateway Extension Support
Dem2Det Enabled
Message Reception with Different PDU Lengths
AUTOSAR Monitoring and Debugging Run-Time Measurement
Deactivation of Partial Networking during Run-Time

Table 3-3 Features provided beyond the AUTOSAR standard



Note

Some of these additional non-AUTOSAR features are only available if they are explicitly ordered by the customer.

3.1.3 Limitations

3.1.3.1 Ranges of Timers

The ranges for the following timers are limited concerning the specified range:

- > NM Timeout Timer: 1..65535 ms
- > Remote Sleep Indication Timer: 1..65535 ms (if remote sleep indication is enabled)
- > NM message Cycle Time: 1..65535 ms
- > NM message Reduced Time: 1..65535 ms
- > NM message Tx Timeout Time: 1..65535 ms

All timers should be multiples of the main functions cycle time.

The following provides a detailed description of the functional scope.

3.2 Network Management Mechanism

As described above the AUTOSAR NM is a decentralized direct network management. This means that every network node has the same functionality and performs state and operation mode changes self-sufficient depending on the internal state and whether network management messages are still received.

The network management mechanism is quite simple:

- > Every network node transmits its NM messages only as long as it needs to communicate with other network nodes. Normally bus-communication is required as long as KL15 is set or during follow-up.
- > If there is no more network node in the whole network that need to communicate with other network nodes, any node transmits no more NM messages.
- > Each network node performs a transition to bus-sleep mode a certain time after the last NM message has been transmitted by any node. Therefore all nodes will go to bus-sleep mode together.
- > If any network node requires bus-communication at any time it can wake up the whole network by transmitting NM messages.

**Caution**

The transmission of application messages, e.g. transmitted by the Com, does not stop immediately after the last NM message has been transmitted.

**FAQ**

The application is in charge of the decision whether the bus communication is required or not.

The following figure shows the state diagram of the CAN NM. The events are calls of CAN NM functions by the application or data link layer or the timeout of internal timers.

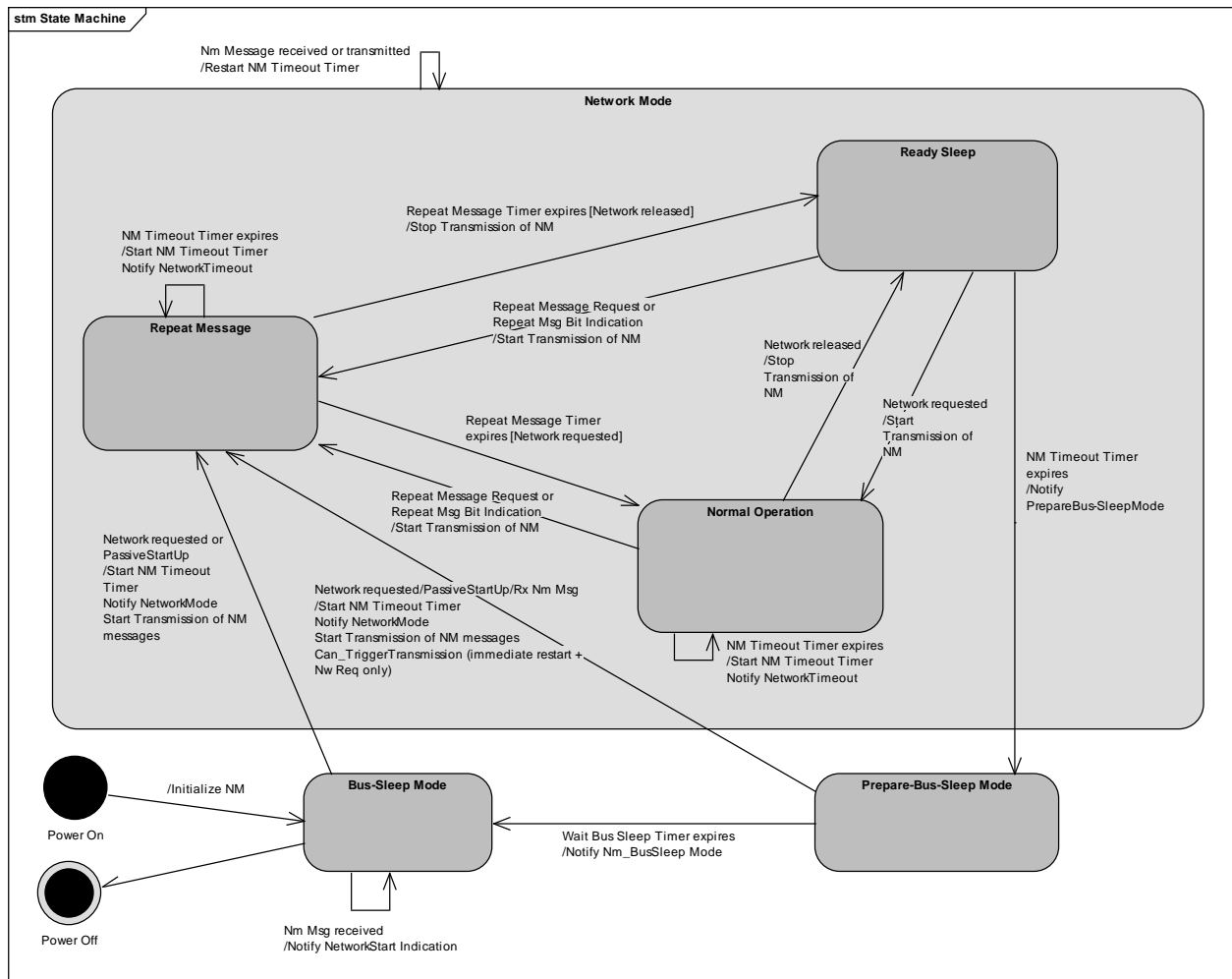


Figure 3-1 State Diagram of CAN NM from SWS CAN NM [3]

3.3 Passive Mode

Nodes in passive mode cannot transmit NM messages and therefore they do not actively participate in the network. Due to that, passive nodes cannot request the network and they change immediately from Repeat Message State to Ready Sleep State, i.e. the repeat message time is configured to zero.

This mode can be used for nodes that do not need to keep the bus awake to save resources.

3.4 Operation Modes and States

The AUTOSAR NM consists of three operation modes:

- > Network Mode
- > Prepare Bus-Sleep Mode
- > Bus-Sleep Mode

The NM Interface is notified about changes of the operation mode by calling the following functions:

Entering Bus-Sleep Mode:

```
void Nm_BusSleepMode ( const NetworkHandleType nmChannelHandle
) (5.8.1)
```

Entering Network Mode:

```
void Nm_NetworkMode ( const NetworkHandleType nmChannelHandle ) (5.8.1)
```

Leaving Network Mode:

```
void Nm_PrepareBusSleepMode (
    const NetworkHandleType nmChannelHandle ) (5.8.1)
```

Information about the current state and the current mode is provided by the service call `CanNm_GetState` (chapter 5.5.2.2).

The CAN NM notifies changes of the current state to the NM Interface by calling the optional function

```
void Nm_StateChangeNotification (
    const NetworkHandleType nmChannelHandle,
    const Nm_StateType nmPreviousState,
    const Nm_StateType nmCurrentState ) (5.8.1)
```

3.4.1 Network Mode

The Network Mode comprises three states:

- > Repeat Message
- > Normal Operation
- > Ready Sleep

This is the mode in that the ECU is 'online' and participates in the network. The participation in the network is active or passive depending on the state:

- > Active participation: a node keeps the network awake (Repeat Message State and Normal Operation State).
- > Passive participation: a node is ready for sleep (Ready Sleep State) and any other node keeps the network alive.

The application is notified about entering the Network Mode by a call of the function:

```
void Nm_NetworkMode ( const NetworkHandleType nmChannelHandle ) (5.8.1)
```

The NM Interface notifies leaving the Network Mode to the application by a call of the function:

```
void Nm_PrepareBusSleepMode (
    const NetworkHandleType nmChannelHandle ) (5.8.1)
```



Info

The Com is active during Network Mode. It is started upon entry and stopped upon exit of Network Mode. I.e. application messages are transmitted and received within Network Mode!

3.4.1.1 Repeat Message State

The Repeat Message State is entered:

- > If a NM message has been received in Prepare Bus-Sleep Mode.
- > If the network has been requested by a call of `CanNm_NetworkRequest` in Bus-Sleep or Prepare Bus-Sleep Mode.
- > If the network is woken up from Bus-Sleep Mode or from Prepare Bus-Sleep Mode by a call of `CanNm_PassiveStartUp`.
- > If any network node (including itself) has requested node detection in Ready Sleep or Normal Operation State.

In Repeat Message State the NM messages are transmitted cyclically regardless whether bus load reduction is enabled or disabled.

The Repeat Message State is left after a certain customizable time.

Depending on the bus-communication need of the application Normal Operation State or Ready Sleep State is entered upon exit of Repeat Message State.

3.4.1.2 Normal Operation State

The network management stays in Normal Operation State until the bus-communication is released. The local bus-communication request of the application is distributed in the network by the transmission of NM messages.

3.4.1.3 Ready Sleep State

The network management stays in Ready Sleep State as long as the application does not request bus-communication and the application of any other node still requests bus-communication (by transmitting NM messages).

A certain customizable time after the last network node has released bus-communication a transition to Prepare Bus-Sleep Mode is performed (i.e. Network Mode is left).

3.4.2 Prepare Bus-Sleep Mode

The transmission of application messages is stopped when entering Prepare Bus-Sleep Mode. The bus activity is calmed down (pending message are still transmitted) in this mode and finally there is no more activity on the bus.

After the 'wait bus sleep time' the drop out of Prepare Bus-Sleep Mode to Bus-Sleep Mode the NM Interface is notified by the service call:

```
void Nm_BusSleepMode ( const NetworkHandleType nmChannelHandle  
    ) (5.8.1)
```



Caution

When entering Bus-Sleep Mode the physical bus interface has to be put in sleep mode. On CAN channels the transceiver and the CAN-Controller have to be put in sleep mode. This is done by the ComM.

The Prepare Bus-Sleep Mode is left to Network Mode upon successful reception of a NM message or if the network has been requested by a call of `CanNm_NetworkRequest` or if the network has been woken up by a call of `CanNm_PassiveStartUp`.

3.4.3 Bus-Sleep Mode

All network nodes perform a transition to bus-sleep mode at almost the same time, if no NM message is lost and there hasn't been a wake-up by any node.

The bus-sleep mode is left (wake-up) by a call of

```
Nm_ReturnType CanNm_PassiveStartUp (
    const NetworkHandleType nmChannelHandle )
```

 (5.5.2.3)

or if the network has been requested by a call of

```
Nm_ReturnType CanNm_NetworkRequest (
    const NetworkHandleType nmChannelHandle )
```

 (5.5.2.4.1)

In both cases Repeat Message State will be entered (see Chapter 3.4.1.1 'Repeat Message State').

If a NM message is received in Bus-Sleep Mode the service

```
void Nm_NetworkStartIndication (
    const NetworkHandleType nmChannelHandle )
```

 (5.8.1)

is called by CAN NM.

3.4.4 Wake-up Registration

The network management needs to know whether the application requires bus communication. Per default the network management does not actively participate in the network. The active participation in the network is requested by the service

```
Nm_ReturnType CanNm_NetworkRequest (
    const NetworkHandleType nmChannelHandle )
```

 (5.5.2.4.1)

Calling this function in Bus-Sleep Mode starts the network and leads to a transition to Repeat Message State (see Chapter 3.4.3 'Bus-Sleep Mode').

If bus communication is not required anymore it can be released with the service

```
Nm_ReturnType CanNm_NetworkRelease (
    const NetworkHandleType nmChannelHandle )
```

 (5.5.2.4.2)


Caution

When the communication control service is used the bus-communication shall not be released as long as the NM message transmission ability is disabled.

Note that a bus-communication request is handled within the next task. Nevertheless it is ensured that a communication request always leads to start-up even if the communication is released before the next task is executed. Within Network Mode a fast toggling (i.e. without task execution in between) of the communication status does not lead to any action.

3.4.5 User Data Handling

The user data for the NM message transmitted next on the bus can be set by the service:

```
Nm_ReturnType CanNm_SetUserData (
    const NetworkHandleType nmChannelHandle,
    const uint8* nmUserDataPtr )
```

 (5.5.2.6.1)

The service

```
Nm_ReturnType CanNm_GetUserData (
    const NetworkHandleType nmChannelHandle,
    uint8* nmUserDataPtr )
```

 (5.5.2.6.2)

allows reading the user data of the last received message on the bus.

As the NM PDU layout is completely configurable, the user data placement depends on the given configuration.

The PDU layout and the content of the user data itself are OEM specific and therefore provided by the OEM.

Note that for setting of NM user data a second possibility can be configured. Refer to chapter 3.15 'NM User Data via Com' for more information. If the feature 'NM User Data via Com' is used the API `CanNm_SetUserData()` is not available.

3.5 Network Management Message Transmission and Reception

3.5.1 AUTOSAR CAN Interface

The network management requests the transmission of NM messages by calling the service `CanIf_Transmit`. The application has to take care of the user data. For details refer to chapter 3.4.5 'User Data Handling'.

The successful transmission of every network management message is confirmed by the CAN Interface with the service

```
void CanNm_TxConfirmation ( PduIdType canNmTxPduId )
```

 (5.7.1.2)

The CAN Interface indicates the reception of NM message by calling the service

```
void CanNm_RxIndication ( PduIdType canNmRxPduId,
    const PduInfoType* PduInfoPtr )
```

 (5.7.1.1)

3.5.2 PDU Message Layout

The default PDU Message Layout is described in the following table:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 7	User data 5							
Byte 6	User data 4							
Byte 5	User data 3							
Byte 4	User data 2							
Byte 3	User data 1							
Byte 2	User data 0							
Byte 1	Source Node Identifier							
Byte 0	Control Bit Vector							

Table 3-4 PDU NM Message Layout

The number of User Data Bytes as well as the positions of the Control Bit Vector and Source Node Identifier can be configured arbitrarily but depend on the availability of the corresponding features (User Data Support / Node Detection of Coordinator Extension Support / Node Identifier).

Refer to chapter 6.3 'Configuration with GENy' for more details about the configuration.

3.5.3 Message Transmissions

A standard sequence for NM message transmissions when Repeat Message is entered is depicted in Figure 3-2.

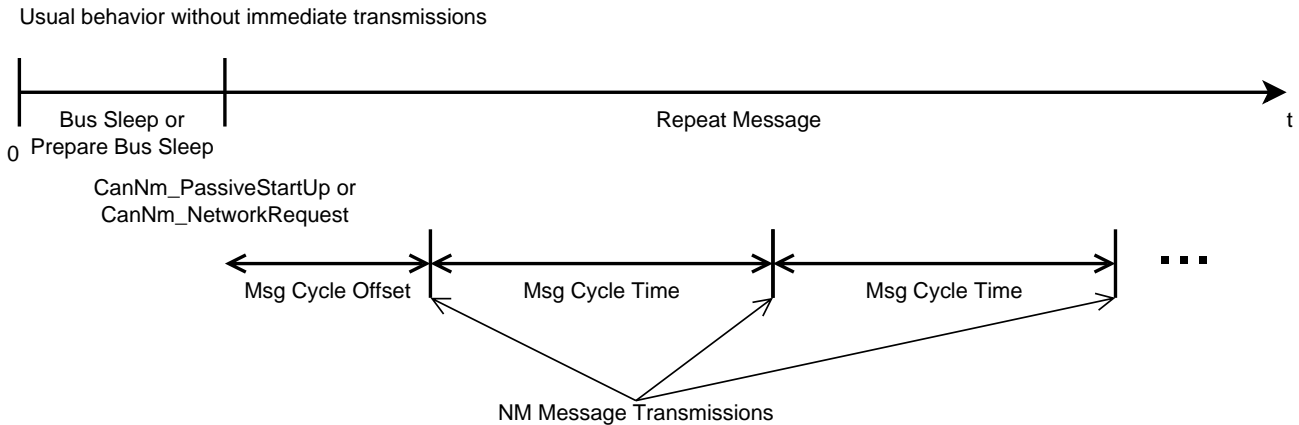


Figure 3-2 Usual Behavior of NM Transmissions when Repeat Message is entered

The first NM message is transmitted when 'Msg Cycle Offset' ms has been elapsed after Repeat Message has been entered. The next NM message will be transmitted after Msg Cycle Time has been elapsed. The configuration settings that influence this behavior are:

- > 'Msg Cycle Offset': time before the transmission of the first NM message
- > 'Msg Cycle Time': time between each message transmission
- > 'Immediate Restart Enabled': if enabled, an additional NM message will be sent immediately upon an active request (CanNm_NetworkRequest was called) from Prepare Bus Sleep to Repeat Message (see also chapter 3.21) in case 'Msg Cycle Offset' is greater than zero
- > 'Immediate Nm Transmissions': if this setting is greater than zero, an immediate NM message will be sent when Repeat Message is entered due to an active request. The interval between NM messages will be different for the next ('Immediate Nm Transmissions' - 1) NM messages to be sent. Refer to chapter 3.20 for more details.
- > 'Repeat Message Time': this setting determines for how long CanNm shall keep the Repeat Message state. If the node has been requested passively, the next state will be Ready Sleep.

A detailed description of these configuration parameters is given in chapter 6.3.1 and 6.3.2.

Note that the NM message is sent as long as the NM state is 'Repeat Message' or 'Normal Operation'. For details about these states, see also chapter 3.4.1.

**Note**

The lower layer (e.g. CAN Interface) may reject the send request if Network Mode has just been entered.

CanNm usually does not retry to issue the send request of the NM message. There are features, which may enable retries in certain conditions:

If 'Immediate Nm Transmissions' are greater than zero, the rejected send request is not considered as 'Immediate Transmission' (see chapter 3.20).

3.5.4 Bus Load Reduction

The bus load reduction is started automatically if enabled and when Normal Operation state is entered. When Normal Operation state is left, bus load reduction algorithm is stopped.

3.5.5 Support for RX PDUs with Different Lengths

The CanNm supports messages with different lengths (DLCs). This support can be enabled by disabling the 'Adjust CanIf RangeConfig DLCCheck' setting in the module configuration.

**Info**

The 'Adjust CanIf RangeConfig DLCCheck' setting is most likely pre-configured and cannot be changed inside GENy. Probably it is not even visible in the CanNm module settings.

The setting is enabled, if a macro definition of `CANNM_CANIF_RANGE_CONFIG_DLC_CHECK` can be found in `CanNm_Cfg.h`.

If the 'Adjust CanIf RangeConfig DLCCheck' setting is disabled, you can enter within the CanIf configuration (see also the description of the 'DLC Check' feature in [6]) the minimum DLC value that shall be accepted for NM messages. Smaller messages will then be discarded. Messages with same DLC or greater DLC will be received and forwarded to the NM.

If the 'Adjust CanIf RangeConfig DLCCheck' setting is enabled, the value in the CanIf configuration will be automatically set by the NM to 'Pdu Length' and cannot be changed by the user. Therefore only NM messages where the DLC is equal to (or greater than) 'Pdu Length' will be accepted by CanNm.

In all cases, CanNm evaluates only bytes within the NM messages up to 'Pdu Length', i.e. in case the received NM message length is greater than the configured 'Pdu Length' the respective bytes will be ignored. For smaller messages than 'Pdu Length', the missing bytes are considered as being zero.

Examples:

The length of a received message is 8, 'Pdu Length' is configured to 6. In this case, the last two user data bytes are not further processed by CanNm (e.g. `CanNm_GetUserData` does not return data for these two bytes). The message is accepted regardless of the 'Adjust CanIf RangeConfig DLCCheck' setting.

The length of a received message is 4, 'Pdu Length' is configured to 6. In this case, bytes 4 and 5 are considered as being zero (e.g. `CanNm_GetUserData` returns 0 for these

bytes). The message is not accepted at all if the 'Adjust CanIf RangeConfig DLCCheck' setting is enabled, because the minimum required DLC for accepting the message is set to 'Pdu Length' = 6.

In summary, the minimum required number of bytes for a received NM message that should be processed by CanNm has to be configured by using the CanIf DLC Check feature [6]. If 'Adjust CanIf RangeConfig DLCCheck' is disabled, the user may configure the CanIf DLC Check by himself/herself, otherwise it is handled automatically by CanNm.

Refer to chapter 6.3 'Configuration with GENy' for more details about the configuration.

3.6 Node Detection

In order to detect which nodes are currently present within the network, this mechanism can be used. If a network node requests node detection, the requesting node performs a transition to Repeat Message State and sets the Repeat Message Bit within the NM PDU. Upon reception of the Repeat Message Bit all network nodes perform a transition to Repeat Message State. This allows the requesting node to collect all source node identifiers from active nodes.

The local source node identifier can be retrieved by the service

```
Nm_ReturnType CanNm_GetLocalNodeIdentifier (
    const NetworkHandleType nmChannelHandle,
    uint8* const nmNodeIdPtr )
```

 (5.5.2.7.3)

The source node identifier from the last received message can be retrieved by the service

```
Nm_ReturnType CanNm_GetNodeIdentifier (
    const NetworkHandleType nmChannelHandle,
    uint8* const nmNodeIdPtr )
```

 (5.5.2.7.2)

3.7 NM PDU Receive Indication

The NM Interface is notified about the reception of an NM message by the optional function

```
void Nm_PduRxIndication ( const NetworkHandleType
                           nmChannelHandle )
```

 (5.8.1)

The CAN NM notifies the callback directly to the NM Interface in context of the function CanNm_RxIndication (see chapter 5.7.1.1).

3.8 Communication Control

In order to support ISO 14229 Communication Control Service \$28 the network management has a message transmission control status, which allows disabling the transmission of NM messages while bus-communication is requested. Therefore the function

```
Nm_ReturnType CanNm_DisableCommunication (
    const NetworkHandleType nmChannelHandle )
```

 (5.5.2.5.1)

can be called. The transmission of NM messages will be stopped within the next CAN NM main function call.

The NM PDU transmission ability is enabled again by the service

```
Nm_ReturnType CanNm_EnableCommunication (
    const NetworkHandleType nmChannelHandle )
```

 (5.5.2.5.2)


Caution

An ECU shall not shut down if the NM PDU transmission ability is disabled.

3.9 Gateway Functionality

3.9.1 Remote Sleep Indication and Cancellation

In order to synchronize networks it might be necessary to get an indication whether no more network nodes require bus-communication. This is the so-called 'Remote Sleep Indication'. The start of the remote sleep indication is indicated by

```
void Nm_RemoteSleepIndication (
    const NetworkHandleType nmChannelHandle )
```

 (5.8.1)

If any NM message is received during Normal Operation State or Ready Sleep State after the remote sleep indication the service 'Remote Sleep Cancellation' is called:

```
void Nm_RemoteSleepCancellation (
    const NetworkHandleType nmChannelHandle )
```

 (5.8.1)

It is also possible to retrieve the current remote sleep state by calling the service:

```
void CanNm_CheckRemoteSleepIndication (
    const NetworkHandleType nmChannelHandle,
    boolean* const nmRemoteSleepIndPtr )
```

 (5.5.2.9.1)

Remote sleep indication can only be checked in Ready Sleep state and Normal Operation state.

3.9.2 Extended Remote Sleep Indication and Cancellation

For a gateway it might be necessary to check the remote sleep indication status also in the Repeat Message State. Therefore this feature can be enabled if the 'Remote Sleep Indication' is enabled.

Please note that this feature is additional and can only be enabled if explicitly available.

3.9.3 Bus Synchronization

In order to synchronize networks for a synchronized shutdown it might be necessary to transmit an asynchronous NM message to reset the network timers. This can be done by calling the service:

```
Nm_ReturnType CanNm_RequestBusSynchronization (
    const NetworkHandleType nmChannelHandle )
```

 (5.5.2.8.1)

However this service shall only be called within Network Mode.

3.10 Limp Home Indication and Cancellation

If in Normal Operation state a NM message could not be transmitted and no NM message is received for a configurable amount of time AUTOSAR just provides the possibility of

error reporting to the DET and DEM if error reporting is enabled. Therefore the CAN NM provides the possibility of an additional error reporting via the callback

```
void Nm_LimpHomeIndication (
    const NetworkHandleType nmChannelHandle )
```

 (5.8.2.1)

to the upper layer.

If a NM message was successfully transmitted or received or the Network Mode is left after 'Limp Home Indication' the NM Interface is notified by calling

```
void Nm_LimpHomeCancelation (
    const NetworkHandleType nmChannelHandle )
```

 (5.8.2.1)

Additionally the Limp Home status can be checked by a call of

```
Nm_ReturnType CanNm_CheckLimpHomeIndication (
    const NetworkHandleType nmChannelHandle,
    boolean* const nmLimpHomeIndPtr )
```

 (5.8.2.1)

Please note that this feature is additional and can only be enabled if explicitly available (see also chapter 3.1 'Overview of the Functional Scope').

3.11 Coordinator Synchronization Support

For supporting the NM Interface Coordination Extension with more than one coordinator connected to the same channel it is necessary to provide three additional bits in the CBV. Therefore the service call

```
Nm_ReturnType CanNm_SetCoordBits (
    const NetworkHandleType nmChannelHandle,
    const uint8 nmCoordBits )
```

 (5.5.3.2)

allows to set and clear those bits. The following table shows all values for setting those bits and the corresponding symbol defined in the `NmStack_Types.h`:

Value	Symbol	Corresponding Bit
0x08	NM_COORD_BIT_SLEEP	Announce Sleep Bit; Bit 3 in the CBV
0x06	NM_COORD_BIT_PRIO_MASK	Two Priority Bits; Bit 1 and 2 in the CBV
0x0E	NM_COORD_BIT_MASK	All three Coordination Bits; Bit 1 – 3 in the CBV

Table 3-5 NM Coordination Bits

For clearing the bits any value that does not contain one of the three coordination bits set can be used, e.g. 0x00.

If a message is received where at least the priority bits are set, the following callback to the NM Interface is called correspondingly:

```
void Nm_ActiveCoordIndication (
    const NetworkHandleType nmChannelHandle,
    const uint8 nmCoordPrio,
    const uint8 nmSleepInd )
```

 (5.8.2.2)

Please note that this feature is additional and can only be enabled if explicitly available.

**Caution**

The 'Coordinator Extension Support' requires the Control Bit Vector.
Therefore this feature has to be enabled if the Coordination Extension Support is used.

3.12 Error Handling

3.12.1 Development Error Detection

Reporting of development errors is configurable (refer to chapter 6.3.1 'Component Configuration').

3.12.1.1 Det_ReportError

Development errors are reported by the service

```
void Det_ReportError (
    uint16 ModuleId, uint8 InstanceId,
    uint8 ApiId, uint8 ErrorId )
```

(5.6)

Please refer to the documentation of the development error tracer [4] for further information and a detailed description of the API. The module Id, API Ids and error Ids can be found within the software components' header file.

3.12.1.2 Error Codes

CAN NM reports the following error codes to the Development Error Tracer:

Error Code	Description
0x01 CANNM_E_NO_INIT	API service used without module initialization.
0x02 CANNM_E_INVALID_CHANNEL	API service used with wrong channel handle.
0x03 CANNM_E_INIT_FAILED ³	NM initialization has failed.
0x05 CANNM_E_CANIF_TRANSMIT_ERROR ³	Call of function CanIf_Transmit has failed.
0x11 CANNM_E_DEV_NETWORK_TIMEOUT	NM-Timeout Timer has abnormally expired outside of the Ready Sleep State.
0x12 CANNM_E_NULL_POINTER ⁴	Null pointer has been passed as an argument.
0x20 CANNM_E_RXINDICATION_DLC_ERROR ⁵	DLC of received NM message does not match with configured PDU Length.
0x21 CANNM_E_PDUR_TRIGGER_TX_ERROR ⁵	Call of function PduR_TriggerTransmit failed.

Table 3-6 DET Error Codes of CAN NM

3.12.2 Diagnostic Event Manager

Production errors are reported to the diagnostic event manager if the CANNM_PROD_ERROR_DETECT switch is set to STD_ON (refer to chapter 6.3.1 'Component Configuration').

³ This error is only reported if the 'Dem2Det Enabled' switch is enabled. The error is specified as development error in AUTOSAR Release 3.2.

⁴ Error does not apply to the function CanNm_Init.

⁵ Vector extension

3.12.2.1 Dem_ReportErrorStatus

Reporting errors to the diagnostic event manager is done by the service

```
void Dem_ReportErrorStatus (
    uint8 ErrorId, uint8 StatusId )
```

 (5.6)

Please refer to the documentation of the diagnostic event manager [5] for further information and a detailed description of the API. The error Ids and the status Ids can be found within the diagnostic event manager components' header file.

3.12.2.2 Error Codes

Error Code	Description
CANNM_E_INIT_FAILED ⁶	NM initialization has failed.
CANNM_E_CANIF_TRANSMIT_ERROR ⁶	Call of Can Interface function CanIf_Transmit has failed
CANNM_E_NETWORK_TIMEOUT	NM-Timeout Timer has abnormally expired outside of the Ready Sleep State.

Table 3-7 DEM Error Codes of CAN NM

3.13 Multiple ECU Support

The CAN NM supports Multiple ECU configuration when the configuration in GENy is done with an AUTOSAR ECU configuration file. Refer to [9] for more information about the Multiple ECU feature.

When this feature is used by applying an appropriate ECU configuration file in GENy, the generated code will contain multiple configuration structures. When initializing the CAN NM the active ECU must be chosen by passing the corresponding pointer to the active configuration within the initialization function

```
void CanNm_Init (
    const CanNm_ConfigType * const nmConfigPtr )
```

 (5.5.1.1)

Refer also to chapter 4.4 'Initialization'.

For CAN NM the 'Msg Cycle Offset', the 'Msg Reduced Time', the CAN NM transmission message handle (i.e. for each ECU instance one CAN NM message must be available in the configuration) and the parameter 'Node Id' are different depending on the active configuration.

Refer to chapter 6.3.2 for more details about these configuration settings.

⁶ This error is only reported if the 'Dem2Det Enabled' switch is disabled. The error is specified as production error in AUTOSAR Release 3.0 and Release 3.1.

**Caution**

Note that if the NM User Data via Com feature (see also chapter 3.15) is used in combination with the Multiple ECU Support, a PDU Overlay has to be created for the NM User Data TX I-PDU. Refer to chapter 3.15.1.1 for details.

The PDU Overlay has to be used, because the NM User Data TX I-PDU is always the same object independent of the currently active ECU (determined by the pointer value passed to CanNm_Init).

3.14 Multiple Configuration Support

The CAN NM can be used in multiple configuration environments when the configuration in GENy is done with an AUTOSAR ECU configuration file. No special support is provided except that the main function can be called even if the initialization was not performed because CAN NM is not active in the chosen identity. In such a case the main function does not perform any actions. Also no DET error is notified (refer to chapter 3.12.1 'Development Error Detection').

**Note**

The CAN NM implementation assumes that no function (except the main function) will be called for channels that are not active in the selected identity.

In contradiction to the feature 'Multiple ECU Support' only one configuration is available which has to be used for all identities.

3.15 NM User Data via Com

The CAN NM supports the possibility to write the NM user data via Com signals. Therefore the signals have to be provided within an extra I-PDU in the configuration. The CAN NM updates its transmission buffer each time before sending a NM message with the current data. Therefore it calls the function:

```
void PduR_CanNmTriggerTransmit (
    PduIdType ComTxPduId, uint8 * CanNmSduPtr ) (5.6)
```

When the NM message has been successfully transmitted the confirmation is forwarded to by calling the function:

```
void PduR_CanNmTxConfirmation (
    PduIdType ComTxPduId ) (5.6)
```

Depending on the signal and I-PDU configuration a signal change can lead to a request for an immediate NM message transmission by calling the function

```
void CanNm_Transmit (
    PduIdType CanNmTxPduId,
    const PduInfoType * PduInfoPtr )
```

(5.5.2.10.1)

The CAN NM then transmits the changed data in the next main function when the transmission of NM messages is allowed. This behavior is given in Figure 3-3. Afterwards the message cycle timer is restarted, i.e. the cyclic message transmission raster changes.

The spontaneous transmission through CanNm_Transmit is allowed in the NM states Repeat Message and Normal Operation if and only if

- > 'Pn Enabled' is ON and 'Pn Handle Multiple Network Requests' is OFF AND/OR
- > 'Car Wake Up Rx Enabled' is ON.

Behavior with CanNm_Transmit usage

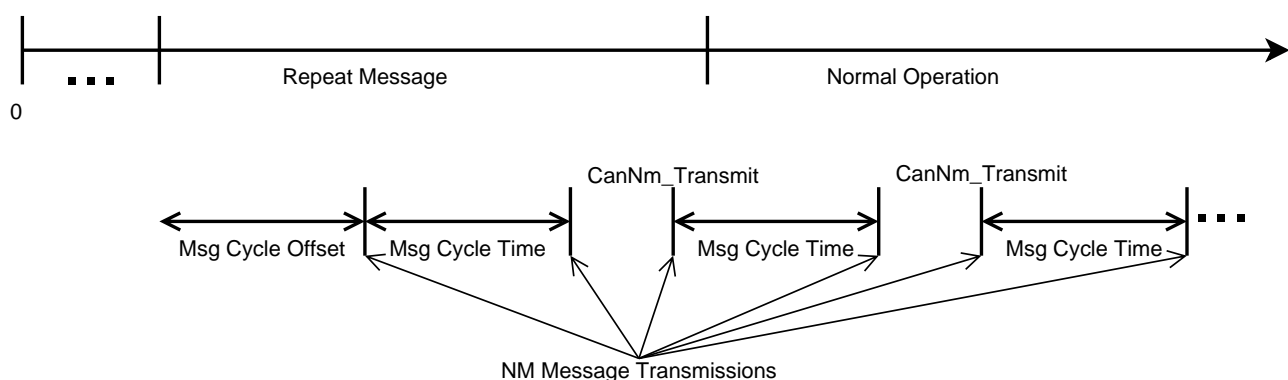


Figure 3-3 Immediate Transmission due to Signal Change inside User Data I-PDU



Info

Note that this feature can be currently only configured when using an AUTOSAR ECU configuration file.

The following chapters describe more detailed the configuration preconditions of this feature. How this feature is configured in GENy can be found in chapter 6.3.2 'Channel Configuration'.

Note that some additional configuration for this feature has to be done in the PDU router. Refer to [10] for details.

3.15.1 Configuration Preconditions in an AUTOSAR ECU Configuration

For using the feature 'NM User Data via Com' some additional configuration content within the AUTOSAR system description extract is necessary. The following table provides an overview of the items that have to be added to the system description.

Configuration Element	Description
Signal I-PDU	For each NM message one signal I-PDU must be configured. An appropriate signal mapping to the I-Signals has to be defined here. I-PDUs are defined in the ECU-specific part.

Configuration Element	Description
I-Signal	Multiple system signals can be defined for each NM message. At least one signal is required. I-Signals are defined in the ECU-specific part and refer to a system signal.
System Signal	For each I-Signal a corresponding system signal is necessary which defines length, data type and initial value.
I-PDU Port	For each I-PDU an I-PDU port with the communication direction 'OUT' is required.
Signal Port	For each signal a signal port with the communication direction 'OUT' is required.
I-PDU Triggering	For each I-PDU an I-PDU triggering is required that references to the corresponding I-PDU port and the signal I-PDU.
Signal Triggering	For each I-Signal a signal triggering is required that references to the corresponding signal port and I-Signal.

Table 3-8 Configuration Precondition Overview for AUTOSAR ECU Configurations

**Caution**

Changing the AUTOSAR System Description or AUTOSAR ECU Extract is usually not the task of the user. This is the task of the OEM.

Note that the AUTOSAR System Description and AUTOSAR ECU Extracts may even be generated by the Vector toolchain from other input files (e.g. dbc) and these files already contain this information.

Additional a reference from the NM PDU to the related I-PDU with the signals must be established. This can be accomplished by

- > adding 'Admin Data' to the NM PDU. This applies to ECU Extracts for AUTOSAR versions less than 3.2. The following an example of how this should be done:

```
<NM-PDU>
  <SHORT-NAME>NM_PDU</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="DV:SIGNAL-I-PDU-REF">
        <SD>/ECU/PDUS/APPL_NM_UD_1_TGW</SD>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
  <LENGTH>64</LENGTH>
</NM-PDU>
```

- > adding 'ISignalToIPduMappings' to the NM PDU. This applies to ECU Extracts for AUTOSAR versions greater than or equal to 3.2. Note that the Vector toolchain automatically creates a Signal I-PDU containing the signals provided in the

'ISignalToI-PDU-Mappings' list in the AUTOSAR ECU Configuration. The short name of this created I-PDU is the concatenation of the short name of the NM-PDU and the suffix '_UD'. The following example demonstrates how this should be done (NM_PDU_UD will be the short name of the Signal I-PDU):

```
<NM-PDU>
  <SHORT-NAME>NM_PDU</SHORT-NAME>
  <LENGTH>8</LENGTH>
  <I-SIGNAL-TO-I-PDU-MAPPINGS>
    <I-SIGNAL-TO-I-PDU-MAPPING>
      <SHORT-NAME>NM_USR_DT</SHORT-NAME>
      <PACKING-BYTE-ORDER>MOST-SIGNIFICANT-BYTE-LAST</PACKING-
BYTE-ORDER>
      <SIGNAL-REF DEST="I-
SIGNAL">/ISignal/NM_USR_DT_SIGNAL</SIGNAL-REF>
      <START-POSITION>32</START-POSITION>
    </I-SIGNAL-TO-I-PDU-MAPPING>
  </I-SIGNAL-TO-I-PDU-MAPPINGS>
</NM-PDU>
```

3.15.1.1 Caveats for Multiple ECU Support

If Multiple ECU Support is used in the ECU Configuration, a PDU Overlay has to be created by the user for the NM TX PDU. Refer to [9] for details about creating such a PDU Overlay for the Physical Multiple ECU Use Case.

For AUTOSAR versions 3.2 the Signal I-PDU which has to be configured for a PDU Overlay cannot be found directly inside the ECU Extract, it is created by the Vector toolchain during the creation of the AUTOSAR ECUC. The short name of the NM-PDU ("NM_PDU" in the example above) has to be used for configuring the PDU Overlay.

3.15.2 Configuration Preconditions in a DBC Based Configuration

The configuration of this feature via DBC is not supported.

3.16 QNX Support

The AUTOSAR CAN NM provides QNX support.

This feature can be used in ECUs where the main operating system has a long booting time and until the system has booted a so called mini-driver containing parts of the communication stack takes care of the basic communication tasks. After the main operating system is running the state of the mini-driver shall be transferred so that the main system can take over the already started communication. Therefore the API for getting the module context has to be available in the mini-driver and the API for the setting of the module context has to be available in the main system.

This feature is not described in detail here as it is part of the Vector QNX wrapper module and shall only be used by this module. Details can be found in the corresponding technical reference of the QNX wrapper module.

3.17 Extended Bus Synchronization

If this feature is enabled the following API is available independently from 'Bus Synchronization Enabled':

```
void CanNm_RequestBusSynchronization (
    const NetworkHandleType nmChannelHandle )           (5.5.2.8.1)
```

This API can be used by the application to perform additional NM message transmissions in the start-up phase to reduce start-up delays of other nodes.



Caution

The API shall only be called by the application during start-up phase. It is strictly recommended to call this API not when the NM already has entered 'Ready Sleep State', especially when the coordination feature is used.

This feature is optional and has to be configured (refer to chapter 6.3.1 'Component Configuration').

3.18 Gateway Extension Support

The gateway extension support contains two optional features that can be enabled separately in the configuration, the 'NM Gateway Extension Support' and the 'Diagnostic Gateway Extension Support'.

When 'NM Gateway Extension Support' or 'Diagnostic Gateway Extension Support' is enabled then CAN NM provides the following additional callback to the application each time a NM message is received:

```
void Nm_GwPduRxIndication (
    const NetworkHandleType nmChannelHandle,
    uint8 nmNodeId, uint8 nmReqId, uint8 nmReqCh )       (5.8.2.3)
```

Within this callback the node identifier, byte 4 (requested identifier) and byte 5 (requested channels) of the NM message are passed to the NM Interface.

When 'Diagnostic Gateway Extension Support' is enabled then additionally the following service function is provided by the CAN NM:

```
Nm_ReturnType CanNm_SetDiagGwReqId ( uint8 nmReqId )   (5.5.3.4)
```

If this function is called by the NM Interface CAN NM sets byte 4 in the NM transmission data buffer with the passed identifier and requests an additional NM message transmission on all CAN NM channels.



Info

Note that an additional NM message transmission can only be performed when the CAN NM is in state 'Repeat Message' or 'Normal Operation'. Otherwise the request is discarded.

More information about the configuration of the features can be found in chapter 6.3.1 'Component Configuration'.

3.19 Active Wake-up Handling

The mode change from Bus-Sleep Mode or Prepare Bus-Sleep Mode to Network Mode triggered by `CanNm_NetworkRequest()` is specified as "Active Wake-up". Upon an Active Wake-up the CAN NM sets the active wake-up bit within the Control Bit Vector at bit position 4.

This feature is optional and has to be configured. For configuration information refer to chapter 6.3.1 'Component Configuration' and chapter 6.3.2 'Channel Configuration'.

3.20 Immediate Nm Transmissions

If an Active Wake-up occurs the CAN NM transmits the first NM message immediately (the NM message offset time is ignored) when entering Repeat Message State. For the next NM messages CAN NM uses a faster NM message cycle time. Afterwards it uses the normal NM message cycle time. This behavior is illustrated in Figure 3-4.

Behavior with $n := \text{Immediate Nm Transmissions} > 0$

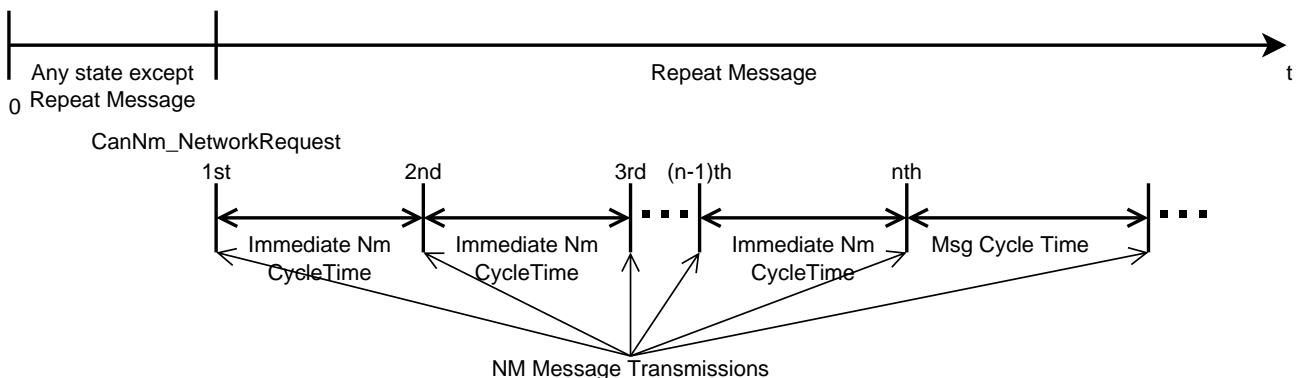


Figure 3-4 Immediate Nm Transmissions

The number of 'Immediate Nm Transmissions' is the number that is configured for this parameter in the channel settings in GENy (see also chapter 6.3.2 'Channel Configuration'). As it can be seen in Figure 3-4, after the first Immediate Nm Transmission the interval between the NM messages is 'Immediate Nm CycleTime' for $(n-1)$ times. Then, the usual interval 'Msg Cycle Time' is used again.

Note that "Any state except Repeat Message" in Figure 3-4 refers to 'Bus Sleep' and 'Prepare Bus Sleep'. If the setting 'Pn Handle Multiple Network Requests' is ON, it also refers to 'Ready Sleep' and 'Normal Operation'.

This feature is optional and has to be enabled in the configuration. The amount of messages that are transmitted faster ('Immediate Nm Transmissions') and the fast message cycle time ('Immediate Nm CycleTime') can also be configured. Refer to chapter 6.3.1 'Component Configuration' and chapter 6.3.2 'Channel Configuration' for configuration details.



Note

This feature should not be confused with the possibility for an immediate transmission if the 'Com User Data Enabled' feature is on (chapter 3.15) and should also not be confused with the 'Immediate Restart Enabled' feature described in the following chapter.



Note

If the send request of an 'immediate transmission' is rejected by the lower layer (e.g. CanIf), the rejected send request is not considered as 'immediate transmission'. That means that the counter that counts the number of 'immediate transmissions' *ImmediateNmMsgCount* is not decremented.

Example: Let 'Immediate Nm Transmissions' := 2. The initial counter value of *ImmediateNmMsgCount* is 1.

1. When Repeat Message has just been entered, the first transmission request $TReq_A$ is rejected. *ImmediateNmMsgCount* is not decremented.
2. CanNm waits 'Immediate Msg CycleTime' (first interval t_{int1st}).
3. CanNm sends the NM message successfully. *ImmediateNmMsgCount* is decremented to 0.
4. CanNm waits 'Immediate Msg CycleTime' again (second interval t_{int2nd}).
5. CanNm sends the next NM message successfully.
6. Then 'Msg Cycle Time' is waited until the next NM message is sent because *ImmediateNmMsgCount* is already 0.

If the first NM transmission request $TReq_A$ was successful (step 1), the second interval time t_{int2nd} would be 'Msg Cycle Time' instead of 'Immediate Msg CycleTime'.

3.21 Immediate Restart Enabled

This feature enables the possibility to send an additional NM message upon the transition from Prepare Bus Sleep to Repeat Message if and only if the network is requested actively (e.g. there is a request for Full Communication in ComM).

Behavior with Immediate Restart Enabled

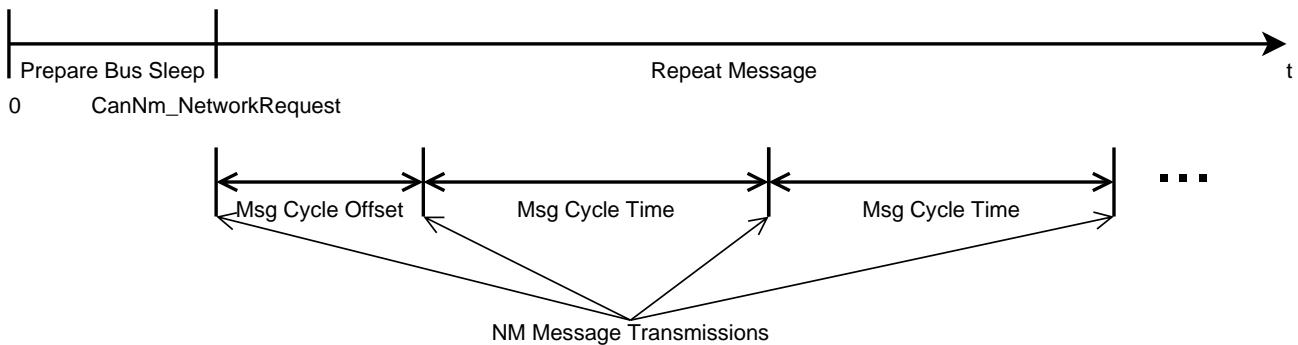


Figure 3-5 Behavior for NM Transmissions if Immediate Restart Enabled is ON

This behavior is depicted in Figure 3-5. Note that the only difference to the standard transmission behavior (i.e. 'Immediate Restart Enabled' would be OFF, for the behavior see also chapter 3.5.3) is the additional NM message right after Repeat Message has been entered.



Note

The additional NM message is only sent if the 'Msg Cycle Offset' setting is greater than 0 and if 'Immediate Nm Transmissions' = 0 (refer to chapter 3.20 for details).

3.22 Car Wake-up

Every ECU shall be able to wake up all other ECUs of the car. This wake-up request information is contained in the NM message user data of an ECU. The central gateway ECU evaluates the Car Wake-up request information and wakes up all connected communication channels.

This feature is optional and has to be configured. Further information about the configuration can be found in chapter 6.3.1 'Component Configuration' and chapter 6.3.2 'Channel Configuration'.

3.22.1 Rx-Path

If the CAN NM receives a NM message it evaluates the user data content. If the Car Wake-up Bit is set and the Node ID passes a filter (if Node ID filter is enabled) the CAN NM notifies the NM via the following callback function:

```
void Nm_CarWakeUpIndication (
    const NetworkHandleType nmChannelHandle )
```

(5.8.1)

3.22.2 Tx-Path

For the transmission of the Car Wake-up Bit it has to be set at the corresponding location within the NM user data. If the feature 'NM User Data via Com' is used and the corresponding signal and I-PDU are configured for directly transmitting a changed signal the information is sent immediately. Refer also to chapter 3.15 'NM User Data via Com'.

**Info**

It is recommended to use the feature 'NM User Data via Com' for the transmission path.

3.23 Partial Networking

To reduce the power consumption of ECUs it shall be possible to switch off the communication stack during active bus communication. To control the shutdown and wake-up of such ECUs the CAN NM provides an additional algorithm. The NM message user data contains the information which partial networks (PN) are requested. This information is evaluated by the CAN NM and provided to the upper layer in an aggregated form by updating the content of additional I-PDUs in the Com.

**Caution**

The configuration setting 'Com User Data Enabled' has to be enabled to use the Partial Networking feature. Refer to chapter 3.15 for more details about the 'NM User Data via Com' feature.

Algorithm details are described in the following sub-chapters.

This feature and all of its sub-features are optional and have to be configured. For configuration information refer to chapter 6.3.1 'Component Configuration' and chapter 6.3.2 'Channel Configuration'.

3.23.1 Availability of Partial Network Request Information

To distinguish between NM messages containing PN cluster request information (CRI) and NM messages without CRI a special bit in the control bit vector (bit 6) is used. Only if this bit is set the NM message contains PN information and will be processed by the algorithm.

3.23.2 Transmission of the CRI Bit in the NM User Data

The CAN NM sets the CRI Bit at bit position 6 in the Control Bit Vector to 1 for each channel if the Partial Networking feature is enabled on the corresponding channel.

3.23.3 Filter Algorithm for Received NM Messages

NM messages that are not relevant for an ECU with PN must be dropped. Therefore the content of received NM messages is evaluated after the filter algorithm described in this section has been activated. Otherwise the usual way of receiving messages is being used. The filter is disabled after the initialization of the CAN NM module. The message reception filter is being activated after a call of `CanNm_ConfirmPnAvailability` (refer to chapter 5.5.2.11.1 'CanNm_ConfirmPnAvailability: Notification for activating the PN filter functionality' for further details). If Bus Sleep Mode is being entered, the filter is being disabled. The filter algorithm works as follows:

If the CRI bit is cleared the NM message is not relevant for the ECU.

If the CRI bit is set the CAN NM evaluates the CRI content of the NM message. The location and the length of the CRI in the NM user data can be configured. Each bit within the CRI content represents one cluster. The corresponding cluster is being requested if and only if the bit that belongs to the cluster is set. Because not every cluster is relevant for the ECU a configurable PN filter mask is applied to the CRI content. Irrelevant cluster requests can be ignored by setting the corresponding bit in the filter to 0. If at least one bit within the received PN information matches with a bit in the PN filter mask the NM message is relevant for the ECU, otherwise the NM message is not relevant for the ECU.

If a NM message is not relevant and the configuration parameter 'All Nm Messages Keep Awake' is true the standard NM message reception handling is done, otherwise the NM message is ignored.

If a NM message is relevant the CAN NM performs the standard NM message reception and additionally the filtered PN content of this message is used for the further PN algorithm.

3.23.4 Aggregation of Requested Partial Networks

The CAN NM aggregates requested PN information by two slightly different algorithms. First the external (received) and internal (sent) PN requests are aggregated over all networks (channels) to a combined state called External Internal Requests Aggregated (EIRA). Second only the external (received) PN requests are aggregated for each network to the so called External Requests Aggregated (ERA) state. Both algorithms can be activated independently in the configuration.

For the EIRA algorithm every received or sent NM message on any network is evaluated and the relevant PN information (according to the PN filter mask and the CRI bit) is combined to one aggregated state. Therefore this state contains the information which partial networks are active on the whole ECU.

The ERA algorithm performs the evaluation of the received NM messages and storage of the relevant PN information (according to the PN filter mask and the CRI bit) per network. Therefore the ERA state contains for each network the information which partial networks are requested by other ECUs and have to be active due to external needs.

Whenever a cluster is requested the first time (i.e. a bit is set the first time within this PN information) the new request is stored and a timer is started. When the request is repeated before the timer elapses the timer is restarted. When the timer elapses the request is deleted.

Any change (storing or deleting a request) within the EIRA or ERA leads to an update of the content of the EIRA or ERA I-PDU in the Com. Therefore the following function is called with the corresponding EIRA or ERA PDU handle:

```
void PduR_CanNmRxIndication (
    PduIdType ComTxPduId, uint8 * CanNmSduPtr ) (5.6)
```

Note that one ERA I-PDU exists for each network.

3.23.5 Spontaneous Sending of NM Messages

When a new PN is internally requested the corresponding bit in the NM message user data will be set. This request must be immediately visible on the bus by sending the updated user data content as fast as possible. Therefore two mechanisms can be used.

3.23.5.1 Using Com Transmission on Change Mechanism

When the NM user data is set via Com the signals can be configured for immediate transmission on change. This would lead to one additional NM message transmission whenever the content of the signal changes. Refer also to chapter 3.15 'NM User Data via Com'.

To enable this behavior, the setting 'Pn Handle Multiple Network Requests' has to be turned OFF (see also chapter 6.3.2).

3.23.5.2 Using NM Request and Immediate Nm Transmission

When CAN NM is in Network Mode and the upper layer requests network again by calling the function 'CanNm_NetworkRequest' (see chapter 5.5.2.4.1 'CanNm_NetworkRequest: Request the Network' for details) the CAN NM performs a state transition to Repeat Message. This leads to an immediate transmission of the NM message followed by several transmissions with a faster cycle time.



Caution

Note that the feature 'Immediate Nm Transmission' (refer to chapter 3.20 'Immediate Nm Transmissions') must be enabled when using this mechanism for spontaneous sending of NM messages.

Note that this mechanism will only be active if PN feature is enabled.

To enable this behavior, the setting 'Pn Handle Multiple Network Requests' has to be turned ON (see also chapter 6.3.2).

3.23.6 Deactivation of Partial Networking during Run-Time

The whole Partial Network feature can be deactivated on all CAN NM channels during run-time. To accomplish this, the function CanNm_DisablePNActivationState has to be called after CanNm_Init and before the first call of CanNm_MainFunction_X.

This allows the deactivation on all CAN NM channels.

3.24 Usage of PduInfoType

Since AUTOSAR Release 3.2 APIs to the PduR or from the CanIf dealing with user data shall use only the PduInfoType for passing data instead of passing only the data. PduInfoType does not only contain the pointer to the data buffer but also the length of the passed data. Additionally the transmission triggering functions shall return a value to indicate whether the user data handling could be performed or failed.

This API type with PduInfoType parameters is supported by the CAN NM. The other API type used in AUTOSAR Releases 3.0 and 3.1 is no longer supported by the CAN NM.

The CanNm calls or implements the following APIs that are affected by this feature:

- > CanNm_RxIndication (refer to chapter 5.7.1.1 for details)
- > PduR_CanNmTriggerTransmit (refer to chapter 5.6 for details)
- > PduR_CanNmRxIndication (refer to chapter 5.6 for details)

4 Integration

4.1 Files

The CAN NM consists of the following files:

Files of CAN NM










CanNm.c	Source code of CAN NM. The user must not change this file!	
CanNm.h	API of CAN NM. The user must not change this file!	
CanNm_Cbk.h	API of CAN NM callback functions. The user must not change this file!	
CanNm_Cfg.c	Pre-compile variant configuration source file. The user must not change this file!	
CanNm_Cfg.h	Configuration header file for CAN NM. The user must not change this file!	
CanNm_Lcfg.c	Link-time variant Configuration source file. The user must not change this file!	
CanNm_Lcfg.h	Link-time variant Configuration header file. The user must not change this file!	
CanNm_Pbcfg.c	Post-build variant Configuration source file. The user must not change this file!	
CanNm_Pbcfg.h	Post-build variant Configuration header file. The user must not change this file!	

Table 4-1 CAN NM Integration Files

4.2 Include Structure

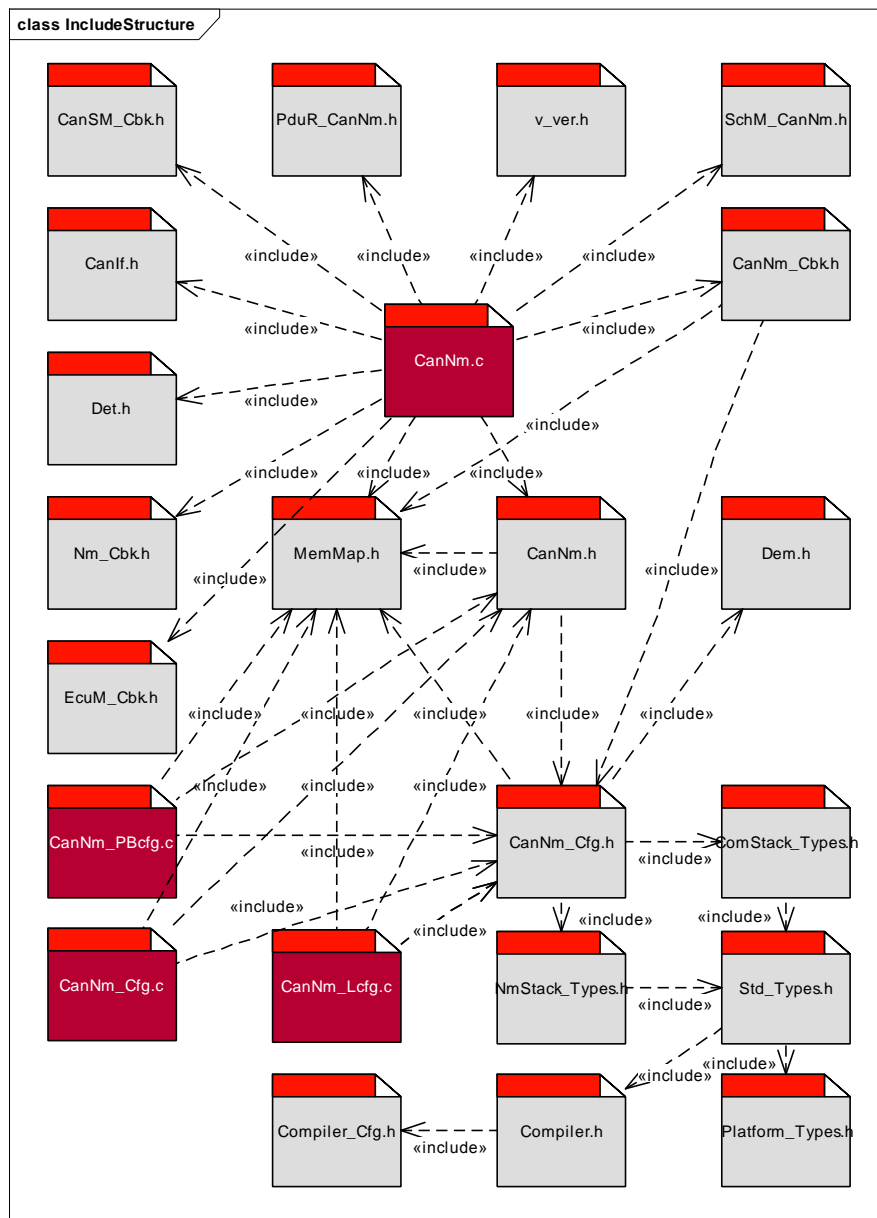


Figure 4-1 Include Structure of CAN NM

4.3 Version Changes

Changes and the release versions of the CAN NM are listed at the beginning of the header and source code.

4.4 Initialization

Before the CAN NM can be used it has to be initialized by the application. The initialization has to be carried out before any other functionality of the CAN NM is executed. It shall take place after initialization of the CAN Interface and prior to initialization of the NM Interface.

Also refer to chapter 5.5.1.1 'CanNm_Init: Initialization of CAN NM'.

**Caution**

The CAN NM assumes that some variables are initialized with zero at start-up. If the embedded target does not initialize RAM within the start-up code the function 'CanNm_InitMemory' has to be called during start-up and before the initialization is performed. Refer also to chapter 3.1.2.3 'Memory Initialization'.

**Note**

In an AUTOSAR environment where the ECU Manager is used, the initialization is performed within the ECU Manager. When using the feature 'Multiple ECU Support' multiple configurations exist and depending on which identity shall be active the correct one has to be chosen.

**Note**

In a CANbedded environment where the CCL is used the initialization is performed by the CCL. 'Multiple ECU Support' is not possible in a CANbedded environment.

4.5 Main Functions

CAN NM contains one or more main functions that have to be called cyclic on task level. Default timing value therefore is 10 milliseconds. The main functions are generated according to the number of CAN NM channels and have to be called individually for each channel. The call cycle time value for the main functions has to be set in the configuration settings (refer to chapter 6.3.2 'Channel Configuration'). It is not relevant in which order the main functions are called.

For further information refer also to chapter 5.5.1.2 'CanNm_MainFunction: Channel Specific Main Functions of CAN NM'.

**Note**

In an AUTOSAR environment where the BSW Scheduler (SchM) is used the main functions are called by the SchM and must not be called by the application. If the SchM is available in GENy, the configuration of the cyclic task timing is automatically updated in the SchM when setting / changing the cycle time in the CAN NM configuration.

**Note**

In a CANbedded environment where the CCL is used the main functions are called by the CCL and must not be called by the application. The CCL is in charge to call the CAN NM with the correct cycle time.

4.6 Critical Sections

The AUTOSAR standard provides with the BSW Scheduler (SchM) a BSW module, which handles entering and leaving critical sections.

Critical sections are supported by the BSW Scheduler.

It is recommended to use the critical section support of the BSW Scheduler. In this case the CAN NM calls the following function when entering a critical section:

```
void SchM_Enter_CanNm ( uint8 ExclusiveArea ) (5.6)
```

When the critical section is left the following function is called by the CAN NM:

```
void SchM_Exit_CanNm ( uint8 ExclusiveArea ) (5.6)
```

The critical section (exclusive area) codes passed to these functions have to be defined and mapped to corresponding interrupt locks by the BSW Scheduler. This can be done in GENy at the BSW Scheduler configuration page. The corresponding sections are already entered there and have to be adapted to the correct interrupt locking mechanism. Details which section needs what kind of interrupt lock are provided in chapter 4.7 'Critical Section Codes'.

For more information about the BSW Scheduler please refer to [7].

4.7 Critical Section Codes

Since the BSW Scheduler is used for critical sections (see also chapter 4.6 'Critical Sections') the CAN NM uses several critical section codes which must lead to corresponding interrupt locks. The mapping is described in the following table:

Critical Section Define	Interrupt Lock
CANNM_EXCLUSIVE_AREA_0	No interruption by any interrupt is allowed. Therefore this section must always lock global interrupts.
CANNM_EXCLUSIVE_AREA_1	<p>No interruption of CanNm_MainFunction by CanNm_SetUserData, CanNm_SetCoordBits or CanNm_SetDiagGwReqId allowed. This means that global interrupts have to be used for this section only if CanNm_MainFunction can be interrupted by one of the following functions:</p> <ul style="list-style-type: none"> > CanNm_SetUserData > CanNm_SetCoordBits > CanNm_SetDiagGwReqId <p>Otherwise no interrupt locks are necessary.</p>

CANNM_EXCLUSIVE_AREA_2	<p>No interruption of CanNm_SetUserData by CanNm_MainFunction allowed.</p> <p>This means that global interrupts must be locked if CanNm_SetUserData can be interrupted by the following functions:</p> <ul style="list-style-type: none"> > CanNm_MainFunction <p>Otherwise no interrupt locks are necessary.</p>
CANNM_EXCLUSIVE_AREA_3	<p>No interruption of CanNm_SetCoordBits by CanNm_MainFunction allowed</p> <p>This means that global interrupts must be locked if CanNm_SetCoordBits can be interrupted by the following functions:</p> <ul style="list-style-type: none"> > CanNm_MainFunction <p>Otherwise no interrupt locks are necessary.</p>
CANNM_EXCLUSIVE_AREA_4	<p>No interruption of CanNm_RxIndication by CanNm_GetUserData or CanNm_GetPduData allowed</p> <p>This means that global interrupts must be locked if CanNm_RxIndication can be interrupted by the following functions:</p> <ul style="list-style-type: none"> > CanNm_GetUserData > CanNm_GetPduData <p>Otherwise no interrupt locks are necessary.</p>
CANNM_EXCLUSIVE_AREA_5	<p>No interruption of CanNm_GetUserData or CanNm_GetPduData by CanNm_RxIndication allowed</p> <p>This means that global interrupts must be locked if CanNm_GetUserData or CanNm_GetPduData can be interrupted by the following functions:</p> <ul style="list-style-type: none"> > CanNm_RxIndication <p>Otherwise no interrupt locks are necessary.</p>

Table 4-2 Critical Section Codes

For API details refer to chapter 5 'API Description'.

5 API Description

5.1 API Categories

The AUTOSAR Network Management supports a multi-channel indexed API. However in single channel configurations this API can be optimized in order to save runtime (refer to chapter 3.1.2.2 'Single Channel Optimization').

5.2 Data Types

The software module CAN NM uses the standard AUTOSAR data types that are defined within `Std_Types.h` and the platform specific data types that are defined within `Platform_Types.h`. Furthermore the standard AUTOSAR NM Stack Types defined within `NmStack_Types.h` are used.

CAN NM also uses the Communication Stack Types defined within `ComStack_Types.h`.

Additionally the following software module specific data types are used for configuration purposes:

Name	Type	Description
<code>CanNm_LChannelConfigType</code>	struct	Structure for all channel specific configuration parameters that are at most link-time able.
<code>CanNm_LConfigType</code>	struct	Structure for configuration parameters that are at most link-time able. Contains the channel specific parameters.
<code>CanNm_PbChannelConfigType</code>	struct	Structure for channel specific configuration parameters that are post-build able.
<code>CanNm_ConfigType</code>	struct	Structure for configuration parameters which are post-build able. Contains the channel specific parameters.

Table 5-1 Data Types

These data types are defined within the software components configuration header file.

5.3 Global Variables

There are no global variables within CAN NM.

5.4 Global Constants

5.4.1 AUTOSAR Specification Version

The version of AUTOSAR specification on which the appropriate implementation is based on is provided by three BCD coded defines:

Name	Type	Description
CANNM_AR_MAJOR_VERSION	BCD	Contains the major specification version number.
CANNM_AR_MINOR_VERSION	BCD	Contains the minor specification version number.
CANNM_AR_PATCH_VERSION	BCD	Contains the patch level specification version number.

Table 5-2 Specification Version API Data

5.4.2 Component Versions

The source code versions of CAN NM are provided by three BCD coded macros (and additionally as constants):

Name	Type	Description
CANNM_SW_MAJOR_VERSION (CanNm_MainVersion)	BCD	Contains the major component version number.
CANNM_SW_MINOR_VERSION (CanNm_SubVersion)	BCD	Contains the minor component version number.
CANNM_SW_PATCH_VERSION (CanNm_ReleaseVersion)	BCD	Contains the patch level component version number.

Table 5-3 Component Version API Data

These constants are declared as external and can be read by the application at any time.

5.4.3 Vendor and Module ID

CAN NM provides the vendor identifier according to AUTOSAR as defines:

Name	Type	Description	Value
CANNM_VENDOR_ID	-	Vendor ID according to AUTOSAR.	30
CANNM_MODULE_ID	-	Module ID according to AUTOSAR.	31

Table 5-4 Vendor/Module ID

5.5 Services Provided by CAN NM

5.5.1 Administrative Functions

5.5.1.1 CanNm_Init: Initialization of CAN NM

CanNm_Init

Prototype

```
void CanNm_Init ( const CanNm_ConfigType * const nmConfigPtr )
```

Parameter

nmConfigPtr	Configuration Pointer
-------------	-----------------------

Return code

-	-
---	---

Service ID

Service ID	0x00
------------	------

Functional Description

Initialization of CAN NM.

Particularities and Limitations

- > During the execution of the function `CanNm_Init()`, it has to be ensured that the execution is not interrupted by any other function of the CanNm module. This can for instance be accomplished by
 - > global interrupt locks OR
 - > CAN interrupt locks.
- > Call context: task level.
- > Not re-entrant.
- > This service has to be called after initialization of CAN Interface and before calling any NM service.

5.5.1.2 CanNm_MainFunction: Channel Specific Main Functions of CAN NM

CanNm_MainFunction

Prototype

```
void CanNm_MainFunction_X ( void )  
(with X=0... Number of CAN NM channels)
```

Parameter

-	-
---	---

Return code

-	-
---	---

Service ID

Service ID	0x13
------------	------

Functional Description

Main functions of CAN NM.

For each CAN NM channel one main function is generated that has to be called with the corresponding channel handle as postfix.

Particularities and Limitations

- > These functions have to be called cyclically on task level.
- > Call context: task level.
- > Not re-entrant.
- > If these services are already called by the OS, SchM or the CCL they must not be called by the application.

5.5.2 Service Functions of CAN NM

5.5.2.1 CanNm_GetVersionInfo: Version Information API

CanNm_GetVersionInfo

Prototype	
void CanNm_GetVersionInfo (Std_VersionInfoType* NmVerInfoPtr)	
Parameter	
NmVerInfoPtr	Pointer where the Version Information shall be copied to
Return code	
-	-
Service ID	
Service ID	0xF1
Functional Description	
This service returns the version information of this module. The version information includes the Module Id, the Vendor Id, the Instance Id and the vendor specific version numbers (BSW00407).	
Particularities and Limitations	
<ul style="list-style-type: none"> > Call context: task level > Re-entrant > This function is enabled if <code>CANNM_VERSION_INFO_API</code> is <code>STD_ON</code> 	

**Caution**

Since there is no interface from CAN NM to any higher software layer except DET, DEM or NM Interface the following services must not be called by ComM, CCL or the application!

5.5.2.2 CanNm_GetState: Get the State of the Network Management**CanNm_GetState****Prototype**

```
Nm_ReturnType CanNm_GetState ( const NetworkHandleType nmChannelHandle,
                               Nm_StateType * const      nmStatePtr,
                               Nm_ModeType * const      nmModePtr )
```

Parameter

nmChannelHandle	Identification of the physical channel
nmStatePtr	Pointer where state of the network management shall be copied to
nmModePtr	Pointer where the mode of the network management shall be copied to

Return code

NM_E_OK	No error
NM_E_NOT_OK	Getting of NM state has failed

Service ID

Service ID	0x0B
------------	------

Functional Description

Get the state and the mode of the network management.

One of the following state values is written to the variable where nmStatePtr points to:

- > NM_STATE_BUS_SLEEP (Bus Sleep)
- > NM_STATE_PREPARE_BUS_SLEEP (Prepare Bus Sleep)
- > NM_STATE_READY_SLEEP (Ready Sleep)
- > NM_STATE_NORMAL_OPERATION (Normal Operation)
- > NM_STATE_REPEAT_MESSAGE (Repeat Message)

One of the following mode values is written to the variable where nmModePtr points to:

- > NM_MODE_BUS_SLEEP (Bus Sleep Mode)
- > NM_MODE_PREPARE_BUS_SLEEP (Prepare Bus Sleep Mode)
- > NM_MODE_NETWORK (Network Mode)

For details about the states and modes refer to chapter 3.4.

Particularities and Limitations

- > Call context: task and interrupt level
- > Re-entrant
- > Called by NM Interface

5.5.2.3 CanNm_PassiveStartUp: Wake up Network Management

CanNm_PassiveStartUp

Prototype

```
Nm_ReturnType CanNm_PassiveStartUp ( const NetworkHandleType nmChannelHandle )
```

Parameter

nmChannelHandle	Identification of the physical channel
-----------------	--

Return code

NM_E_OK	No error
NM_E_NOT_OK	Start of Network Management has failed
NM_E_NOT_EXECUTED	Start of Network Management is currently not executed

Service ID

Service ID	0x01
------------	------

Functional Description

Passive startup of the CAN NM state machine. It triggers the transition from the Bus Sleep Mode / Prepare Bus Sleep Mode to the Network Mode (Repeat Message State). This service has no effect if the current state is not Bus Sleep Mode / Prepare Bus Sleep Mode. In that case NM_E_NOT_EXECUTED is returned.

Particularities and Limitations

- > Call context: task and interrupt level
- > Re-entrant
- > Called by NM Interface

5.5.2.4 Wake-up Registration

5.5.2.4.1 CanNm_NetworkRequest: Request the Network

CanNm_NetworkRequest

Prototype	
Nm_ReturnType CanNm_NetworkRequest (const NetworkHandleType nmChannelHandle)	
Parameter	
nmChannelHandle	Identification of the physical channel
Return code	
NM_E_OK	No error
NM_E_NOT_OK	Requesting the network has failed
Service ID	
Service ID	0x02
Functional Description	
<p>Request the network if the ECU needs to communicate on the bus.</p> <p>If 'Pn Feature' is enabled a network request within the network mode leads to a state transition to repeat message. Refer to chapter 3.23.5.2 'Using NM Request and Immediate Nm Transmission' for details.</p>	
Particularities and Limitations	
<ul style="list-style-type: none"> > Call context: task and interrupt level > Re-entrant > Called by NM Interface > Not available for passive nodes (CANNM_PASSIVE_MODE_ENABLED is STD_ON) 	

5.5.2.4.2 CanNm_NetworkRelease: Release the Network

CanNm_NetworkRelease

Prototype

```
Nm_ReturnType CanNm_NetworkRelease ( const NetworkHandleType nmChannelHandle )
```

Parameter

nmChannelHandle	Identification of the physical channel
-----------------	--

Return code

NM_E_OK	No error
NM_E_NOT_OK	Releasing the network has failed
NM_E_NOT_EXECUTED	Releasing the network is currently not executed.

Service ID

Service ID	0x03
------------	------

Functional Description

Release the network if the ECU doesn't have to communicate on the bus. This service has no effect if the NM Message Transmission is disabled. In that case NM_E_NOT_EXECUTED is returned.

Particularities and Limitations

- > Call context: task and interrupt level
- > Re-entrant
- > Called by NM Interface
- > Not available for passive nodes (CANNM_PASSIVE_MODE_ENABLED is STD_ON)

5.5.2.5 Communication Control Service

5.5.2.5.1 CanNm_DisableCommunication: Disable NM Message Transmission

CanNm_DisableCommunication

Prototype

```
Nm_ReturnType CanNm_DisableCommunication (
    const NetworkHandleType nmChannelHandle )
```

Parameter

nmChannelHandle	Identification of the physical channel
-----------------	--

Return code

NM_E_OK	No error
NM_E_NOT_OK	Disabling NM message transmission has failed
NM_E_NOT_EXECUTED	Disabling NM message transmission is currently not executed

Service ID

Service ID	0x0C
------------	------

Functional Description

Disable transmission of NM messages. This service has no effect if the current state is not Normal Operation State and the network is not requested. In that case NM_E_NOT_EXECUTED is returned.

Particularities and Limitations

- > Call context: task and interrupt level
- > Only re-entrant with different channel handles
- > Called by NM Interface
- > This function is enabled if CANNM_COM_CONTROL_ENABLED is STD_ON and CANNM_PASSIVE_MODE_ENABLED is STD_OFF.

5.5.2.5.2 CanNm_EnableCommunication: Enable NM Message Transmission

CanNm_EnableCommunication

Prototype

```
Nm_ReturnType CanNm_EnableCommunication (
    const NetworkHandleType nmChannelHandle )
```

Parameter

nmChannelHandle	Identification of the physical channel
-----------------	--

Return code

NM_E_OK	No error
NM_E_NOT_OK	Enabling NM message transmission has failed
NM_E_NOT_EXECUTED	Enabling NM message transmission is currently not executed

Service ID

Service ID	0x0D
------------	------

Functional Description

Enable transmission of NM messages. This service has no effect if the NM Message Transmission is not disabled. In that case NM_E_NOT_EXECUTED is returned.

Particularities and Limitations

- > Call context: task and interrupt level
- > Only re-entrant with different channel handles
- > Called by NM Interface
- > This function is enabled if CANNM_COM_CONTROL_ENABLED is STD_ON and CANNM_PASSIVE_MODE_ENABLED is STD_OFF.

5.5.2.6 User Data Handling

5.5.2.6.1 CanNm_SetUserData: Set User Data

CanNm_SetUserData

Prototype	
Nm_ReturnType CanNm_SetUserData (const NetworkHandleType nmChannelHandle, const uint8* const nmUserDataPtr)	
Parameter	
nmChannelHandle	Identification of the physical channel
nmUserDataPtr	Pointer to the user data for the next NM message
Return code	
NM_E_OK	No error
NM_E_NOT_OK	Setting of user data has failed
Service ID	
Service ID	0x04
Functional Description	
Set user data for NM messages transmitted next on the bus.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Call context: task and interrupt level > Only re-entrant with different channel handles > Called by NM Interface > This function is enabled if <code>CANNM_USER_DATA_ENABLED</code> is <code>STD_ON</code> > Not available for passive nodes (<code>CANNM_PASSIVE_MODE_ENABLED</code> is <code>STD_ON</code>) or if <code>CANNM_COM_USER_DATA_ENABLED</code> is <code>STD_ON</code> 	

5.5.2.6.2 CanNm_GetUserData: Get User Data

CanNm_GetUserData

Prototype

```
Nm_ReturnType CanNm_GetUserData ( const NetworkHandleType nmChannelHandle,
                                   uint8* const nmUserDataPtr )
```

Parameter

nmChannelHandle	Identification of the physical channel
nmUserDataPtr	Pointer where user data out of the last received NM message shall be copied to

Return code

NM_E_OK	No error
NM_E_NOT_OK	Getting of user data has failed

Service ID

Service ID	0x05
------------	------

Functional Description

Get user data from the last received NM message on the bus.

Particularities and Limitations

- > Call context: task and interrupt level
- > Re-entrant
- > Called by NM Interface
- > This function is enabled if `CANNM_USER_DATA_ENABLED` is `STD_ON`

5.5.2.6.3 CanNm_GetPduData: Get NM Pdu Data

CanNm_GetPduData

Prototype

```
Nm_ReturnType CanNm_GetPduData ( const NetworkHandleType nmChannelHandle,  
                                uint8* const nmPduDataPtr )
```

Parameter

nmChannelHandle	Identification of the physical channel
nmPduDataPtr	Pointer where the NM Pdu Data shall be copied to

Return code

NM_E_OK	No error
NM_E_NOT_OK	Getting of NM Pdu Data has failed

Service ID

Service ID	0x0A
------------	------

Functional Description

Get the whole PDU data of the last received NM message.

Particularities and Limitations

- > Call context: task and interrupt level
- > Re-entrant
- > Called by NM Interface
- > This function is enabled if CANNM_NODE_ID_ENABLED is STD_ON or CANNM_USER_DATA_ENABLED is STD_ON

5.5.2.7 Node Detection

5.5.2.7.1 CanNm_RepeatMessageRequest: Set Repeat Message Request Bit

CanNm_RepeatMessageRequest

Prototype

```
Nm_ReturnType CanNm_RepeatMessageRequest (
    const NetworkHandleType nmChannelHandle )
```

Parameter

nmChannelHandle	Identification of the physical channel
-----------------	--

Return code

NM_E_OK	No error
NM_E_NOT_OK	Requesting of node detection has failed
NM_E_NOT_EXECUTED	Requesting of node detection is currently not executed

Service ID

Service ID	0x08
------------	------

Functional Description

Set Repeat Message Request Bit for NM messages transmitted next on the bus. This service has no effect if the current state is not Ready Sleep State or Normal Operation State. In that case NM_E_NOT_EXECUTED is returned.

Particularities and Limitations

- > Call context: task and interrupt level
- > Only reentrant with different channel handles
- > Called by NM Interface
- > This function is enabled if CANNM_NODE_DETECTION_ENABLED is STD_ON and CANNM_PASSIVE_MODE_ENABLED is STD_OFF.

5.5.2.7.2 CanNm_GetNodeIdentifier: Get Source Node Identifier

CanNm_GetNodeIdentifier

Prototype

```
Nm_ReturnType CanNm_GetNodeIdentifier (
                                const NetworkHandleType nmChannelHandle,
                                uint8* const nmNodeIdPtr )
```

Parameter

nmChannelHandle	Identification of the physical channel
nmNodeIdPtr	Pointer where node identifier out of the last received NM message is copied to.

Return code

NM_E_OK	No error
NM_E_NOT_OK	Getting of the node identifier from the last NM message has failed

Service ID

Service ID	0x06
------------	------

Functional Description

Get Node Identifier of the last received NM message.

Particularities and Limitations

- > Call context: task and interrupt level
- > Re-entrant
- > Called by NM Interface
- > This function is enabled if `CANNM_NODE_ID_ENABLED` is `STD_ON`

5.5.2.7.3 CanNm_GetLocalNodeIdentifier: Get Local Source Node Identifier

CanNm_GetLocalNodeIdentifier

Prototype

```
Nm_ReturnType CanNm_GetLocalNodeIdentifier (
    const NetworkHandleType nmChannelHandle,
    uint8* const nmNodeIdPtr )
```

Parameter

nmChannelHandle	Identification of the physical channel
nmNodeIdPtr	Pointer where the node identifier of the local node shall be copied.

Return code

NM_E_OK	No error
NM_E_NOT_OK	Getting of node identifier of the local node has failed

Service ID

Service ID	0x07
------------	------

Functional Description

Get Node Identifier of the local node.

Particularities and Limitations

- > Call context: task and interrupt level
- > Re-entrant
- > Called by NM Interface
- > This function is enabled if CANNM_NODE_ID_ENABLED is STD_ON

5.5.2.8 Bus Synchronization

5.5.2.8.1 CanNm_RequestBusSynchronization: Synchronization of Networks

CanNm_RequestBusSynchronization

Prototype

```
Nm_ReturnType CanNm_RequestBusSynchronization (
                                                    const NetworkHandleType nmChannelHandle )
```

Parameter

nmChannelHandle	Identification of the physical channel
-----------------	--

Return code

NM_E_OK	No error
NM_E_NOT_OK	Requesting of bus synchronization has failed
NM_E_NOT_EXECUTED	Requesting of bus synchronization is currently not executed

Service ID

Service ID	0xC0
------------	------

Functional Description

Request bus synchronization. This service has no effect if the current mode is not Network Mode. In that case NM_E_NOT_EXECUTED is returned.

Particularities and Limitations

- > Call context: task level only
- > Re-entrant
- > Called by NM Interface
- > This function is enabled if (CANNM_BUS_SYNCHRONIZATION_ENABLED is STD_ON or if CANNM_EXT_BUS_SYNCHRONIZATION is defined) and if CANNM_PASSIVE_MODE_ENABLED is STD_OFF.

5.5.2.9 Remote Sleep Indication

5.5.2.9.1 CanNm_CheckRemoteSleepIndication: Check for Remote Sleep Indication

CanNm_CheckRemoteSleepIndication

Prototype

```
Nm_ReturnType CanNm_CheckRemoteSleepIndication (
    const NetworkHandleType nmChannelHandle,
    boolean* const nmRemoteSleepIndPtr )
```

Parameter

nmChannelHandle	Identification of the physical channel
nmRemoteSleepIndPtr	Pointer where check result of remote sleep indication shall be copied to

Return code

NM_E_OK	No error
NM_E_NOT_OK	Checking of remote sleep indication has failed

Service ID

Service ID	0xD0
------------	------

Functional Description

Check if remote sleep indication takes place or not.

Particularities and Limitations

- > Call context: task and interrupt level
- > Re-entrant
- > Called by NM Interface
- > This function is enabled if `CANNM_REMOTE_SLEEP_IND_ENABLED` is `STD_ON` and `CANNM_PASSIVE_MODE_ENABLED` is `STD_OFF`.

5.5.2.10 NM Message Transmission Request

5.5.2.10.1 CanNm_Transmit: Spontaneous NM Message Transmission

CanNm_Transmit

Prototype	
<pre>Nm_ReturnType CanNm_Transmit (PduIdType CanNmTxPduId, const PduInfoType* PduInfoPtr)</pre>	
Parameter	
CanNmTxPduId	Identification of the NM user data PDU
PduInfoPtr	Pointer to the Pdu data which shall be transmitted
Return code	
NM_E_OK	No error
NM_E_NOT_OK	Requesting NM message transmission has failed
Service ID	
Service ID	0x14
Functional Description	
The CanNm transmits a NM message within the next main function. The NM message timer will be restarted.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Call context: task and interrupt level > Re-entrant > Called by PDU Router > This function is enabled if CANNM_COM_USER_DATA_ENABLED is STD_ON 	

5.5.2.11 Partial Networking

5.5.2.11.1 CanNm_ConfirmPnAvailability: Notification for activating the PN filter functionality

CanNm_ConfirmPnAvailability

Prototype	
<code>void CanNm_ConfirmPnAvailability (const NetworkHandleType nmChannelHandle)</code>	
Parameter	
nmChannelHandle	Identification of the physical channel
Return code	
-	-
Service ID	
Service ID	0x16
Functional Description	
This function is called by CanSM to indicate that the PN filter functionality on the indicated NM channel shall be enabled.	
Particularities and Limitations	
<ul style="list-style-type: none">> Call context: task and interrupt level> Re-entrant> Called by CANSM> This function is enabled if <code>CANNM_PN_FEATURE_ENABLED</code> is <code>STD_ON</code>	

5.5.3 Vector extensions

5.5.3.1 CanNm_InitMemory: Memory Initialization

CanNm_InitMemory

Prototype

```
void CanNm_InitMemory ( void )
```

Parameter

-	-
---	---

Return code

-	-
---	---

Service ID

Service ID	-
------------	---

Functional Description

If RAM is not automatically initialized at start-up, this function must be called from start-up code to ensure that variables which must be initialized with a certain value (e.g. initialization status with UNINIT value) are set to those values. Refer also to chapter 3.1.2.3 'Memory Initialization'.

Particularities and Limitations

- > The function `CanNm_InitMemory()` has to be called with disabled global interrupts.
- > Call context: task level.
- > Not re-entrant.

5.5.3.2 CanNm_SetCoordBits: Set Coordination Bits in the CBV

CanNm_SetCoordBits

Prototype

```
void CanNm_SetCoordBits ( const NetworkHandleType nmChannelHandle,  
                          const uint8 nmCoordBits )
```

Parameter

nmChannelHandle	Identification of the physical channel
nmCoordBits	Additional coordination bits to set

Return code

-	-
---	---

Service ID

Service ID	0x15
------------	------

Functional Description

Set the additional coordination bits within the control bit vector.

Particularities and Limitations

- > Call context: task level only
- > Only re-entrant with different channel handles
- > Called by NM Interface
- > This function is only available if `CANNM_ENABLE_COORD_SYNC` is defined.

5.5.3.3 CanNm_CheckLimpHomeIndication: Check the Limp Home Status

CanNm_CheckLimpHomeIndication

Prototype

```
Nm_ReturnType CanNm_CheckLimpHomeIndication (
                                const NetworkHandleType nmChannelHandle,
                                boolean* const           nmLimpHomeIndPtr )
```

Parameter

nmChannelHandle	Identification of the physical channel
nmLimpHomeIndPtr	Pointer where check result of limp home status shall be copied to

Return code

NM_E_OK	No error
NM_E_NOT_OK	Checking of limp home status has failed

Service ID

Service ID	0xFB
------------	------

Functional Description

Check if CAN NM entered limp home state or not. For details refer to chapter 3.10 'Limp Home Indication and Cancellation'.

Particularities and Limitations

- > Call context: task level only
- > Re-entrant
- > Called by NM Interface
- > This function is enabled if `CANNM_LIMP_HOME_INDICATION` is defined.

5.5.3.4 CanNm_SetDiagGwReqId: Diagnostic Gateway ID Request

CanNm_SetDiagGwReqId

Prototype	
Nm_ReturnType CanNm_SetDiagGwReqId (uint8 nmReqId)	
Parameter	
nmReqId	Identifier requested to be transmitted on all channels
Return code	
NM_E_OK	No error
Service ID	
Service ID	0xFC
Functional Description	
Set the identifier in byte 4 of the NM transmission message and request an additional message transmission.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Call context: task and interrupt level > Not re-entrant > Called by NM Interface > This function is only available if 'Diagnostic Gateway Extension Support' is enabled. 	

5.5.3.5 CanNm_DisablePNActivationState: Disable Partial Networking

CanNm_DisablePNActivationState

Prototype	
void CanNm_DisablePNActivationState (void)	
Parameter	
-	-
Return code	
-	-
Service ID	
Service ID	0xFD
Functional Description	
This function may be called directly after CanNm_Init to completely disable any PN functionality	
Particularities and Limitations	
<ul style="list-style-type: none"> > The function has to be called after CanNm_Init() (see also chapter 5.5.1.1) and before the first call of CanNm_MainFunction_X() (see also chapter 5.5.1.2). > Call context: task and interrupt level > Not re-entrant > Called by Application > This function is only available if 'Pn Feature Enabled' is enabled. 	

5.6 Services Used by CAN NM

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

Component	API
DET	Det_ReportError ⁷
DEM	Dem_ReportErrorStatus ⁸
SchM	SchM_Enter_CanNm ⁹ SchM_Exit_CanNm ⁹
CanIf	CanIf_Transmit ¹⁰
PduR	PduR_CanNmTriggerTransmit ^{10 11} PduR_CanNmTxConfirmation ^{10 11 12} PduR_CanNmRxIndication ¹³

Table 5-5 Services Used by the CAN NM

5.7 Callbacks Provided by CAN NM

5.7.1 Callback Functions from CAN Interface

5.7.1.1 CanNm_RxIndication: NM Message Indication

CanNm_RxIndication

Prototype	
<pre>void CanNm_RxIndication (PduIdType canNmRxPduId, const PduInfoType * PduInfoPtr)</pre>	
Parameter	
canNmRxPduId	Identification of the network through PDU-ID
PduInfoPtr	Pointer to a PduInfoType struct containing data and length of the received CAN NM SDU
Return code	
-	-
Service ID	
Service ID	0x10

⁷ Service only used if the feature 'Dev Error Detect' is enabled

⁸ Service only used if the feature 'Prod Error Detect' is enabled

⁹ Service only used if the critical section handling is done by the BSW Scheduler

¹⁰ Service only used if the feature 'Passive Mode' is disabled

¹¹ Service only used if the feature 'NM User Data via Com' is enabled.

¹² Service only used if the feature 'Immediate Txconf Enabled' is disabled.

¹³ Service only used if the features 'Pn Eira Calc Enabled' or 'Pn Era Calc Enabled' is enabled.

Functional Description

Indication that a CAN NM L-PDU (NM messages) has been successfully received. The API parameter `canNmRxPduId` refers to the NM channel handle, i.e. a mapping from PduId to NM channel handle is not necessary.

Particularities and Limitations

- > Call context: task and interrupt level
- > Only re-entrant with different channel handles
- > Called by CanIf

5.7.1.2 CanNm_TxConfirmation: NM Message Confirmation Function

CanNm_TxConfirmation

Prototype	
<code>void CanNm_TxConfirmation (PduIdType canNmTxPduId)</code>	
Parameter	
<code>canNmTxPduId</code>	ID of CAN NM L-PDU that has been transmitted.
Return code	
-	-
Service ID	
Service ID	0x0F
Functional Description	
Notification of the successful transmission of a NM message (L-PDU). The API parameter <code>canNmTxPduId</code> refers not to the NM channel handle and a mapping from PduId to NM channel handle is done by the CAN NM (refer also to chapter 3.1.1.2 'PDU ID Used as Argument for CanNm_TxConfirmation').	
Particularities and Limitations	
<ul style="list-style-type: none"> > Call context: task and interrupt level > Only re-entrant with different channel handles > Called by CanIf > Not available for passive nodes (<code>CANNM_PASSIVE_MODE_ENABLED</code> is <code>STD_ON</code>) or if <code>CANNM_IMMEDIATE_TXCONF_ENABLED</code> is <code>STD_ON</code> 	

5.8 Callback Functions used by CAN NM

5.8.1 Service Callback Functions of CAN NM

The following, partly optional, service callback functions of CAN NM are implemented within the NM Interface. Please refer to the document of the NM Interface component for a detailed description of these services:

```

void Nm_NetworkStartIndication( const NetworkHandleType
                                nmChannelHandle )
void Nm_NetworkMode(           const NetworkHandleType
                                nmChannelHandle )
void Nm_PrepareBusSleepMode(   const NetworkHandleType
                                nmChannelHandle )
void Nm_BusSleepMode(         const NetworkHandleType
                                nmChannelHandle )
void Nm_RepeatMessageIndication( const NetworkHandleType
                                  nmChannelHandle )
void Nm_RemoteSleepIndication(  const NetworkHandleType
                                  nmChannelHandle )
void Nm_RemoteSleepCancellation( const NetworkHandleType
                                   nmChannelHandle )
void Nm_StateChangeNotification( const NetworkHandleType
                                   nmChannelHandle,

```

```

const Nm_StateType
nmPreviousState,
const Nm_StateType
nmCurrentState )
void Nm_PduRxIndication(
const NetworkHandleType
nmChannelHandle )
void Nm_TxTimeoutException(
const NetworkHandleType
nmChannelHandle )
void Nm_CarWakeUpIndication(
const NetworkHandleType
nmChannelHandle )

```

These functions are called from task level except the `Nm_PduRxIndication` and `Nm_CarWakeUpIndication`, which can be called from interrupt or from task level.

5.8.2 Additional Service Callback Functions of CAN NM

The service callbacks described in this chapter are additionally called by the CAN NM when the corresponding feature is available.

5.8.2.1 Callbacks for Limp Home Indication

The following two callbacks are provided for Limp Home detection / cancellation:

```

void Nm_LimpHomeIndication(      const NetworkHandleType
                                nmChannelHandle )
void Nm_LimpHomeCancelation(     const NetworkHandleType
                                nmChannelHandle )

```

These functions are called from task level.

5.8.2.2 Callbacks for Coordination Extension

The following callback is provided to the NM Interface if the Coordination Extension is used:

```

void Nm_ActiveCoordIndication(   const NetworkHandleType
                                nmChannelHandle,
                                const uint8 nmCoordPrio,
                                const uint8 nmSleepInd)

```

This function is called from interrupt or task level.

5.8.2.3 Callbacks for Gateway Extension Support

The following callback is provided by CAN NM if the Gateway Extension Support is enabled:

```

void Nm_GwPduRxIndication(      const NetworkHandleType
                                nmChannelHandle,
                                uint8 nmNodeId,
                                uint8 nmReqId,
                                uint8 nmReqCh)

```

This function is called from interrupt or task level.

5.8.2.4 Callbacks for Partial Networking

The following callback is provided by CAN NM if the Partial Networking Feature is enabled:

```

void CanSM_TxTimeoutException(  NetworkHandleType Channel )

```

This function is called from task level.

5.8.2.5 Generator Compatibility Error

The following callback has to be provided by the ECU Manager if the Version Check feature is not explicitly suppressed or if the CRC Check feature is enabled:

```
void EcuM_GeneratorCompatibilityError(  
    uint16 ModuleId,  
    uint8 InstanceId )
```

This function is called during the initialization, i.e. from task level, of the CAN NM if the Generator Version Check or the CRC Check fails.

Refer to [11] for more information.

6 Configuration

AUTOSAR CAN NM supports three different configuration variants: pre-compile configuration, link-time configuration and post-build configuration. Pre-compile configuration can only be carried out if source code is available. Link-time configuration and post-build configuration are only reasonable if source code is not available, i.e. the components are available as object code or library.

It depends on the OEM and license whether pre-compile configuration with source code or link-time / post-build configuration with object code and library has to be used.

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

- > Configuration in Database, for a detailed description see 6.2
- > Configuration in GENy, for a detailed description see 6.3

6.1 Configuration Variants

The CanNm supports the configuration variants

- > VARIANT-PRE-COMPILE
- > VARIANT-LINK-TIME
- > VARIANT-POST-BUILD

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

6.2 Configuration in Data Base

The following table provides a list of database attributes that apply only to CAN NM.

Attribute	Object	Type	Value	Default	Description
NmType ¹⁴	Network	String	"NmAsr", "<OEM>- OSEK"	None	This attribute defines the type of the NM. To allow the configuration of OSEK NM and CAN NM within the same ECU the CAN NM also accepts a valid OSEK NM value. Note that for OSEK NM this type is OEM-specific.
NmAsrNode ¹⁴	Node	Enumeration	No, yes	No	This attribute defines whether the corresponding node uses the AUTOSAR NM ("Yes") or not ("No").
NmAsrMessage ¹⁴	Message	Enumeration	No, yes	No	This attribute defines whether the corresponding

¹⁴ Database attribute is required if GENy is used for the configuration.

					message is an AUTOSAR NM message ("Yes") or not ("No").
NmAsrMessageCount ¹⁴	Network	Integer	1..256	128	This attribute defines the maximum number of NM message received by the NM (Message Range). There is the requirement that the value of attribute NmAsrMessageCount is 2 ⁿ (where n is a natural number).
NmAsrBaseAddress ¹⁴	Network	Hex	0..7FF	0x500	This attribute defines the base address of the NM messages (e.g. 0x500). Only a certain number of nodes can participate in the NM. The CAN identifiers of NM messages are kept in a certain range. This range starts with the ID given by attribute NmAsrBaseAddress. The size of the range is given by attribute NmAsrMessageCount. There is a general requirement that the value of attribute NmAsrBaseAddress must be chosen in a way that masking is possible. This can be achieved if NmAsrBaseAddress is an integer multiple of the attribute NmAsrMessageCount.
NmAsrNodeIdentifier ¹⁴	Node	Hex	0..FF	0	This attribute defines the Source Node Identifier', which is transmitted in the corresponding NM message byte if activated.
NmAsrTimeoutTime	Network	Integer	1..65535	1000	This attribute defines the NM Message Timeout Time (see also Chapter 6.3.2 'Channel Configuration')
NmAsrWaitBusSleepTime	Network	Integer	0..65535	750	This attribute defines the Wait Bus-Sleep Time (see also Chapter 6.3.2 'Channel Configuration')

NmAsrRepeatMessageTime	Network	Integer	0..65535	400	This attribute defines the Repeat Message Time (see also Chapter 6.3.2 'Channel Configuration')
NmAsrCanMsgCycleTime	Network	Integer	1..65535	100	This attribute defines the NM Message Cycle Time (see also Chapter 6.3.2 'Channel Configuration')
NmAsrCanMsgCycleOffset	Node	Integer	0..65535	0	This attribute defines the NM Message Tx Offset Time (see also Chapter 6.3.2 'Channel Configuration'). This attribute has to be set less than $NmAsrCanMsgCycleTime$.
NmAsrCanMsgReducedTime	Node	Integer	1..65535	50	This attribute defines the NM Message Node Specific Reduced Cycle Time (see also Chapter 6.3.2 'Channel Configuration'). This attribute is node-specific and has to be set greater than $\frac{1}{2} * NmAsrCanMsgCycleTime$ but less than $NmAsrCanMsgCycleTime$.

Table 6-1 Database Attributes for CAN NM

6.3 Configuration with GENy

The configuration of the AUTOSAR Network Management is performed with the generation tool GENy.

It is strongly recommend using the database attributes listed in chapter 6.2 'Configuration in Data Base'. Some attributes are mandatory for the usage of CAN NM. Please note that the configuration items gathered from the database are disabled and grayed and cannot be overwritten by the user.

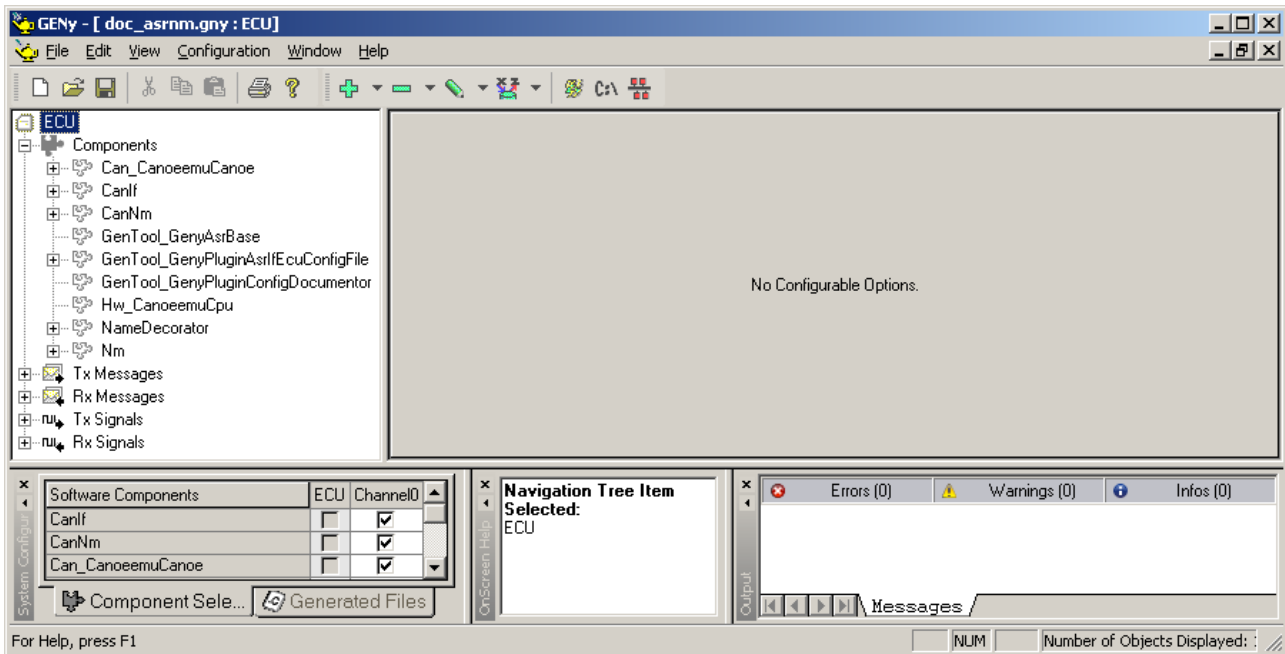


Figure 6-1 Component Selection in GENy

In order to configure the AUTOSAR CAN Network Management component it has to be activated on the designated channels: CanNm.

The activation of CAN NM requires the activation of the CAN Interface ([6]) or of a CANbedded CAN Driver on the same channel.

The component has ECU specific and channel specific settings that have to be customized separately. The detailed configuration is described in the following chapters.



Info

Note that not all parameters described in the following chapters may be visible or available in your configuration. This depends mainly on the available features (refer also to chapter 3.1 'Overview of the Functional Scope') and the OEM pre-configuration.

6.3.1 Component Configuration

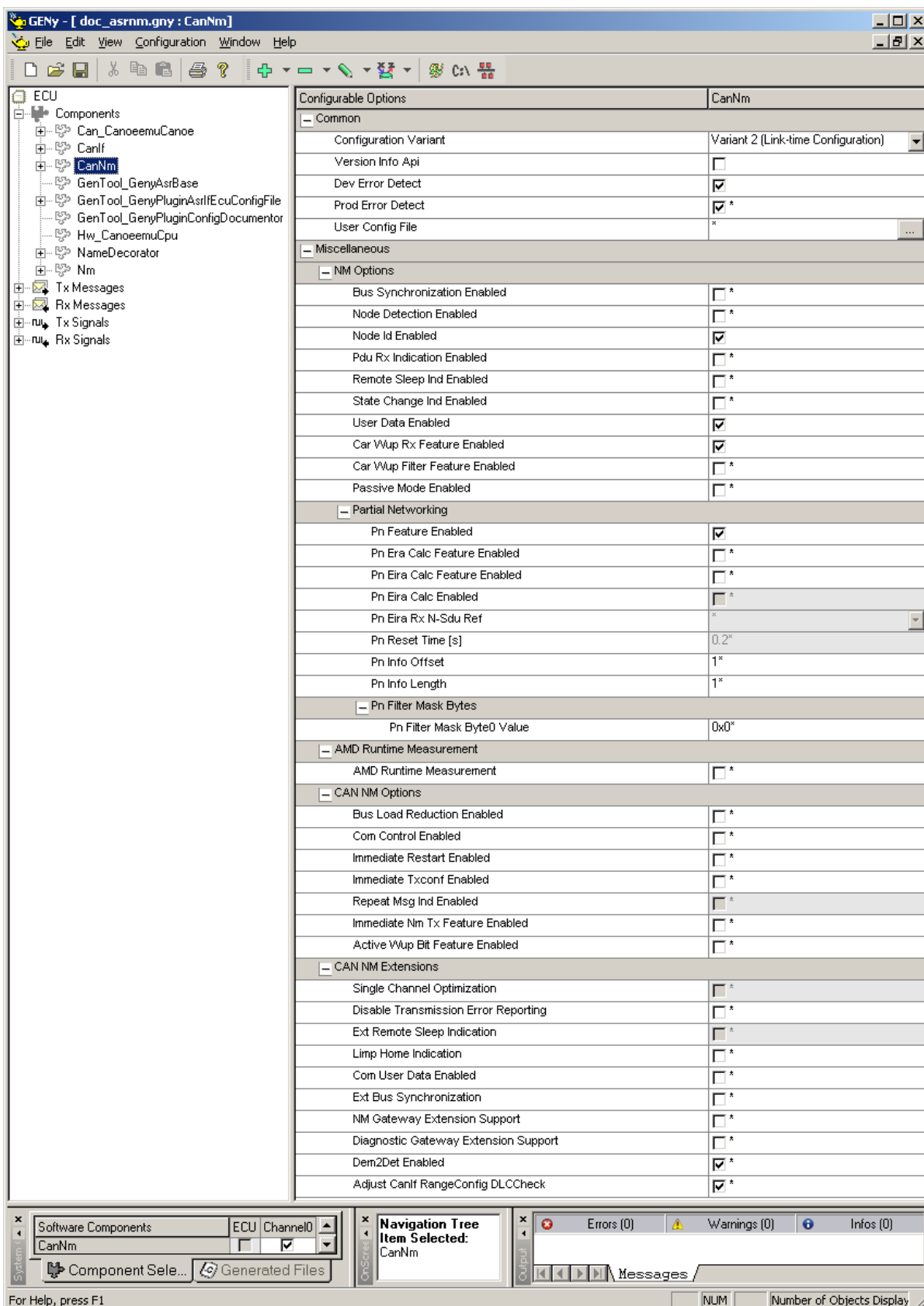


Figure 6-2 Component Configuration CAN NM

Configuration options	Value	Description
Common		
Configuration Variant	<ul style="list-style-type: none"> > Variant 1 > Variant 2 > Variant 3 	Set the AUTOSAR Configuration Variant (Variant 1: Pre-compile configuration; Variant 2: Link-time configuration; Variant 3: Post-build configuration).
Version Info API ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate the version information functionality.
Dev Error Detection ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	<p>Activate/Deactivate error reporting to the development error detection.</p> <p>The development error detection comprises amongst others channel parameter checks and check of initialization upon every service call.</p> <p>It is strongly recommended to disable this option in production code.</p>
Prod Error Detect ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate error reporting to the diagnostic event manager
User Config File	String	<p>Enter the path of a user config file.</p> <p>The user config file is pasted at the end of the configuration file <code>CanNm_Cfg.h</code> during the generation process.</p> <p>It can be used to manually customize the CAN NM.</p>
Post-build configuration¹⁶		
Module Start Address	Memory Address	Set the Start Address of the module's post-build configuration (location in the memory).
NM Options		
Bus Synchronization Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate bus synchronization (gateway functionality). It can only be enabled if Passive Mode is disabled.
Node Detection Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate node detection support. It can only be enabled if Passive Mode is disabled and Node Identifier is enabled.
Node Id Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate node identifier support.
PDU Rx Indication Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate PDU receive indication functionality.
Remote Sleep Ind Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate remote sleep indication (gateway functionality). It can only be enabled if Passive Mode is disabled.
State Change Ind Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate state change notification functionality.

¹⁵ Pre-compile parameter, which should be pre-configured in link-time configurations

¹⁶ Only available when post-build configuration variant is selected.

User Data Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate the user data support.
Car Wup Rx Feature Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	<p>Activate/Deactivate the Car Wake-up support.</p> <p>Note that this parameter just enables the availability of this feature in the code but the algorithm has to be additionally activated on each channel (parameter 'Car Wake Up Rx Enabled', see chapter 6.3.2 'Channel Configuration').</p> <p>Refer to chapter 3.22 'Car Wake-up' for feature details.</p>
Car Wup Filter Feature Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	<p>Activate/Deactivate the Car Wake-up Node ID Filter</p> <p>Note that this parameter just enables the availability of this feature in the code but the algorithm has to be additionally activated on each channel (parameter 'Car Wake Up Filter Enabled', see chapter 6.3.2 'Channel Configuration').</p>
Passive Mode Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate passive mode.
Partial Networking		
Pn Feature Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	<p>Activate/Deactivate the Partial Networking feature availability.</p> <p>Note that this parameter just enables the availability of this feature in the code but the algorithm has to be additionally activated on each channel (parameter 'Pn Enabled', see chapter 6.3.2 'Channel Configuration').</p> <p>Refer to chapter 3.23 'Partial Networking' for feature details.</p>
Pn Era Calc Feature Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	<p>Activate/Deactivate the Partial Networking ERA algorithm availability.</p> <p>Note that this parameter just enables the availability of this feature in the code but the algorithm has to be additionally activated on each channel (parameter 'Pn Era Calc Enabled', see chapter 6.3.2 'Channel Configuration').</p>
Pn Eira Calc Feature Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	<p>Activate/Deactivate the Partial Networking EIRA algorithm availability.</p> <p>Note that this parameter just enables the availability of this feature in the code but the algorithm has to be additionally activated (parameter 'Pn Eira Calc Enabled', see below).</p>
Pn Eira Calc Enabled	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate the Partial Networking EIRA algorithm.
Pn Eira Rx N-Sdu Ref	Combo Box	Select the Name of the Com PDU where the PN information of the EIRA algorithm is provided.
Pn Reset Time [s]	0..65.535	Configure the timeout value for not repeated PN clusters. Value is the same for EIRA and ERA

		algorithm.
Pn Info Offset	1..7	Set the offset where the PN information within the NM PDU starts.
Pn Info Length	0..7	Set the length of the PN information within the NM PDU.
Pn Filter Mask Bytes		
Pn Filter Mask Byte 0 Value	0x0..0xFF	Configure the filter mask value for the first PN information byte. Note that this parameter is only available if 'Pn Info Length' > 0
Pn Filter Mask Byte 1 Value	0x0..0xFF	Configure the filter mask value for the second PN information byte. Note that this parameter is only available if 'Pn Info Length' > 1
Pn Filter Mask Byte 2 Value	0x0..0xFF	Configure the filter mask value for the third PN information byte. Note that this parameter is only available if 'Pn Info Length' > 2
Pn Filter Mask Byte 3 Value	0x0..0xFF	Configure the filter mask value for the fourth PN information byte. Note that this parameter is only available if 'Pn Info Length' > 3
Pn Filter Mask Byte 4 Value	0x0..0xFF	Configure the filter mask value for the fifth PN information byte. Note that this parameter is only available if 'Pn Info Length' > 4
Pn Filter Mask Byte 5 Value	0x0..0xFF	Configure the filter mask value for the sixth PN information byte. Note that this parameter is only available if 'Pn Info Length' > 5
Pn Filter Mask Byte 6 Value	0x0..0xFF	Configure the filter mask value for the seventh PN information byte. Note that this parameter is only available if 'Pn Info Length' > 6
AMD Runtime Measurement		
AMD Runtime Measurement ¹⁵	> Enable > Disable	This parameter defines if AMD Runtime Measurement (RTM) is enabled for this module.
CAN NM Options		
Bus Load Reduction Enabled ¹⁵	> Enable > Disable	Activate/Deactivate the bus load reduction algorithm. It can only be enabled if Passive Mode is disabled.
Com Control Enabled ¹⁵	> Enable > Disable	Activate/Deactivate communication control service. It can only be enabled if Passive Mode is disabled.
Immediate Restart Enabled ¹⁵	> Enable > Disable	Activate/Deactivate immediate restart. It can only be enabled if Passive Mode is disabled.

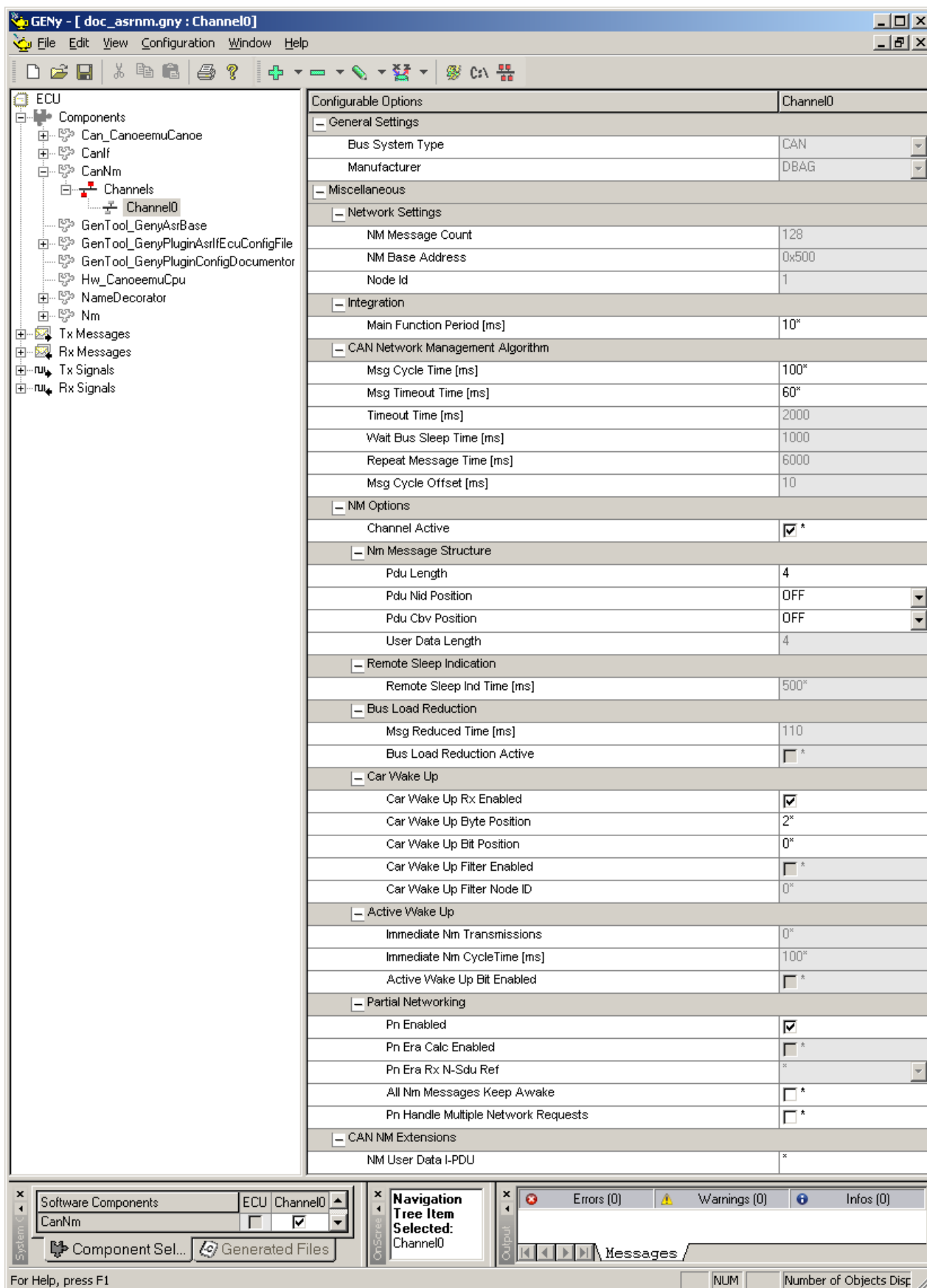
Immediate Txconf Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate immediate transmission confirmation. It can only be enabled if Passive Mode is disabled.
Repeat Msg Ind Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate callback to NM Interface if a state change to Repeat Message due to a repeat message bit or a repeat message request occurred.
Immediate Nm Tx Feature Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	<p>Activate/Deactivate Immediate Nm Transmission feature.</p> <p>Note that this parameter just enables the availability of this feature in the code but the algorithm has to be additionally configured on each channel (parameter 'Immediate NM Transmissions' and 'Immediate Nm Cycle Time', see chapter 6.3.2 'Channel Configuration').</p> <p>Refer to chapter 3.20 'Immediate Nm Transmissions' for feature details.</p>
Active Wup Bit Feature Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	<p>Activate/Deactivate the transmission of the active wake-up bit within the NM user data.</p> <p>Note that this parameter just enables the availability of this feature in the code but the algorithm has to be additionally activated on each channel (parameter 'Active Wake-up Bit Enabled', see chapter 6.3.2 'Channel Configuration').</p> <p>Refer to chapter 3.19 'Active Wake-up Handling' for algorithm details.</p>
CAN NM Extensions		
Single Channel Optimization ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate optimization of the code in single channel pre-compile configurations. Refer to chapter 3.1.2.2 'Single Channel Optimization'.
Disable Transmission Error Reporting ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	<p>Activate/Deactivate the deactivation of the transmission error reporting.</p> <p>If activated the following errors are not reported:</p> <ul style="list-style-type: none"> > CANNM_E_DEV_NETWORK_TIMEOUT > CANNM_E_NETWORK_TIMEOUT > CANNM_E_CANIF_TRANSMIT_ERROR
Ext Remote Sleep Indication ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate the extended remote sleep indication. It can only be enabled if the 'Remote Sleep Indication' is enabled.
Limp Home Indication ^{15,17}	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate the limp home indication. It can only be enabled if Passive Mode is disabled.
Com User Data Enabled	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate the feature 'NM User Data via Com'. It can only be enabled if Passive Mode is disabled. Refer to chapter 3.15 'NM User Data via Com' for more information.

¹⁷ This configuration item is only relevant if the Limp Home Indication feature is available.

Ext Bus Synchronization ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate the feature 'Extended Bus Synchronization'. Refer to chapter 3.17 'Extended Bus Synchronization' for more information.
NM Gateway Extension Support ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate the feature 'NM Gateway Extension Support'. Refer to chapter 3.18 'Gateway Extension Support' for further information.
Diagnostic Gateway Extension Support ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate the feature 'Diagnostic Gateway Extension Support'. Refer to chapter 3.18 'Gateway Extension Support' for further information.
Dem2Det Enabled ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate the feature 'Dem2Det Enabled'. Refer to chapter 3.12 'Error Handling' for further information.
Adjust CanIf RangeConfig DLCCheck ¹⁵	<ul style="list-style-type: none"> > Enable > Disable 	Activate/Deactivate the feature 'Adjust CanIf RangeConfig DLCCheck'. Refer to chapter 3.5.5 'Support for RX PDUs with Different Lengths' for further information.

Table 6-2 Component Configuration CAN NM

6.3.2 Channel Configuration





The screenshot shows the GENy software interface with the 'Channel0' configuration window open. The window is titled 'GENy - [doc_asrnm.gny : Channel0]' and has a menu bar (File, Edit, View, Configuration, Window, Help) and a toolbar. On the left, a tree view shows the project structure under 'ECU' and 'Components', with 'Channel0' selected. The main area displays the 'Configurable Options' for 'Channel0'.


Configurable Options		Channel0
General Settings		
Bus System Type		CAN
Manufacturer		DBAG
Miscellaneous		
Network Settings		
NM Message Count		128
NM Base Address		0x500
Node Id		1
Integration		
Main Function Period [ms]		10*
CAN Network Management Algorithm		
Msg Cycle Time [ms]		100*
Msg Timeout Time [ms]		60*
Timeout Time [ms]		2000
Wait Bus Sleep Time [ms]		1000
Repeat Message Time [ms]		6000
Msg Cycle Offset [ms]		10
NM Options		
Channel Active		<input checked="" type="checkbox"/> *
Nm Message Structure		
Pdu Length		4
Pdu Nid Position		OFF
Pdu Cbv Position		OFF
User Data Length		4
Remote Sleep Indication		
Remote Sleep Ind Time [ms]		500*
Bus Load Reduction		
Msg Reduced Time [ms]		110
Bus Load Reduction Active		<input type="checkbox"/> *
Car Wake Up		
Car Wake Up Rx Enabled		<input checked="" type="checkbox"/>
Car Wake Up Byte Position		2*
Car Wake Up Bit Position		0*
Car Wake Up Filter Enabled		<input type="checkbox"/> *
Car Wake Up Filter Node ID		0*
Active Wake Up		
Immediate Nm Transmissions		0*
Immediate Nm CycleTime [ms]		100*
Active Wake Up Bit Enabled		<input type="checkbox"/> *
Partial Networking		
Pn Enabled		<input checked="" type="checkbox"/>
Pn Era Calc Enabled		<input type="checkbox"/> *
Pn Era Rx N-Sdu Ref		*
All Nm Messages Keep Awake		<input type="checkbox"/> *
Pn Handle Multiple Network Requests		<input type="checkbox"/> *
CAN NM Extensions		
NM User Data I-PDU		*

The bottom of the window shows a status bar with 'For Help, press F1' and a 'NUM' button. The bottom right corner displays 'Number of Objects Disp'.

Figure 6-3 Channel Configuration CAN NM

Configuration options	Value	Description
General Settings		
Bus System Type	Read Only	Bus system type for the specific channel.
Manufacturer	Read Only	Manufacturer according to the database.
Miscellaneous		
Network Settings		
NM Message Count	1..256	Set the number of CAN Ids that are reserved for the network management only. This value determines the filter mask value of the NM range configuration. For a proper configuration this value must be a power of 2. Value is set automatically and is read-only if provided by the database.
NM Base Address	0x0..0x1FFF FFFF	Set the base address of network management ID range. This value determines the range base address (code) value of the NM range configurations. For a proper configuration this value must be a multiple of the parameter 'Nm Message Count'. Value is set automatically and is read-only if provided by the database.
Node Id	Read Only	Configure the source node identifier. This value is transmitted within the NM message. Value is set automatically and is read-only if provided by the database. <div data-bbox="790 1310 877 1400" data-label="Image"></div> <div data-bbox="906 1308 983 1339" data-label="Section-Header">Note</div> <div data-bbox="904 1339 1439 1543" data-label="Text"> <p>Note that this setting may be invisible in the channel settings if you have a Multiple ECU configuration. The setting is visible instead in the identity specific channel settings (see also chapter 6.3.2.1).</p> </div>
Integration		
Main Function Period [ms]	1..255	Set the call cycle time value of the main function for the respective channel. The default value is 10.
CAN Network Management Algorithm		
Msg Cycle Time [ms]	1..65535	Set the cycle time of the NM message in the periodic transmission mode. Value is set automatically and is read-only if provided by the database.
Msg Timeout Time [ms]	1..65535	Set the transmission timeout time of NM message in ms.

		If there is no transmission confirmation by the CAN Interface within this timeout, the CAN NM module shall give an error notification.
Timeout Time [ms]	1..65535	<p>Configure the timeout for the reception and transmission of Network Management Messages. If in Normal Operation State the network node does not require the bus anymore and no more NM messages are received a transition to 'Prepare Bus-Sleep' is performed.</p> <p>Value is set automatically and is read-only if provided by the database.</p>
Wait Bus Sleep Time [ms]	0..65535	<p>Configure the timeout for bus calm down phase (Prepare Bus-Sleep Mode) in ms.</p> <p>Value is set automatically and is read-only if provided by the database.</p>
Repeat Message Time [ms]	0..65535	<p>Set the timeout for Repeat Message State in ms. The Repeat Message State is entered when entering Network Mode and also used for the 'Node Detection' (refer to chapter chapter 3.6 'Node Detection' for further information).</p> <p>Value is set automatically and is read-only if provided by the database.</p>
Msg Cycle Offset [ms]	0..65535	<p>Configure the message transmission offset time in the periodic transmission mode (start delay).</p> <p>Value is set automatically and is read-only if provided by the database.</p> <div data-bbox="785 1254 874 1344">  </div> <div data-bbox="903 1254 1444 1494"> <p>Note</p> <p>Note that this setting may be invisible in the channel settings if you have a Multiple ECU configuration. The setting is visible instead in the identity specific channel settings (see also chapter 6.3.2.1).</p> </div>
NM Options		
Channel Active	<p>> Enable</p> <p>> Disable</p>	Activate/Deactivate a NM channel. If deactivated the channel will be set to Uninitialized.
Nm Message Structure		
Pdu Length	0..8	<p>Configure the length of the NM message.</p> <div data-bbox="778 1832 868 1921">  </div> <div data-bbox="893 1827 1414 1995"> <p>Caution</p> <p>The PDU Length should be configured equally in all nodes within the same network (channel). Background: The CAN NM expects this length for all</p> </div>

		received messages unless the support for RX PDUs with different lengths is active (see chapter 3.5.5 'Support for RX PDUs with Different Lengths' for further details). If the support for RX PDUs with different lengths is not used, the range filter in the CAN Interface is configured adequately and NM messages with a smaller size will not be received.
Pdu Nid Position	<ul style="list-style-type: none"> > Byte 0 > Byte 1 > Off 	Set the position of the node identifier in the NM message. This parameter depends on 'PDU Length' and 'PDU Control Bit Vector Position'.
Pdu Cbv Position	<ul style="list-style-type: none"> > Byte 0 > Byte 1 > Off 	Set the position of the control bit vector in the NM message. This parameter depends on 'PDU Length' and 'PDU Node Identifier Position'.
User Data Length	Read Only	This parameter provides information about the current user data length. It is determined by the settings of 'PDU Length', 'PDU Node Identifier Position' and 'PDU Control Bit Vector Position'.
Remote Sleep Indication		
Remote Sleep Ind Time [ms]	1..65535	Configure timeout for Remote Sleep Indication.
Bus Load Reduction		
Msg Reduced Time [ms]	1..65535	<p>Node specific reduced bus cycle time [ms] in the periodic transmission mode with bus load reduction. Value is set automatically and is read-only if provided by the database.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  <p>Note Note that this setting may be invisible in the channel settings if you have a Multiple ECU configuration. The setting is visible instead in the identity specific channel settings (see also chapter 6.3.2.1).</p> </div>
Bus Load Reduction Active	<ul style="list-style-type: none"> > Enable > Disable 	<p>Activate/Deactivate bus load reduction usage for this channel.</p> <p>This option is not configurable if 'Bus Load Reduction' is disabled.</p>
Car Wake Up		
Car Wake Up Rx Enabled	<ul style="list-style-type: none"> > Enable > Disable 	Activate/deactivate the Car Wake-up handling for this channel.
Car Wake Up Byte Position	0..7	Set the Car Wake-up byte position within the NM user data.

Car Wake Up Bit Position	0..7	Set the Car Wake-up bit position within the 'Car Wake Up Byte'.
Car Wake Up Filter Enabled	> Enable > Disable	Activate/deactivate the Car Wake-up Node ID filtering for this channel.
Car Wake Up Filter Node ID	0..255	Configure the Car Wake-up Node ID for this channel.
Active Wake Up		
Immediate Nm Transmissions	0..255	Configure the number of NM messages which shall be transmitted with the Immediate Nm Cycle Time upon Active Wake-up. If zero is configured the normal transmission handling is performed.
Immediate Nm Cycle Time [ms]	1..65535	Configure the NM message cycle time which shall be used upon Active Wake-up.
Active Wake Up Bit Enabled	> Enable > Disable	Activate/deactivate the setting of the active wake-up bit within the NM user data for this channel.
Partial Networking		
Pn Enabled	> Enable > Disable	Activate/deactivate the Partial Networking algorithm for this channel.
Pn Era Calc Enabled	> Enable > Disable	Activate/deactivate the Partial Networking ERA algorithm for this channel.
Pn Era Rx N-Sdu Ref	Combo Box	Select the name of the Com PDU where the PN information of the ERA algorithm is provided.
All Nm Messages Keep Awake	> Enable > Disable	Activate/deactivate that all NM messages keep the channel awake.
Pn Handle Multiple Network Requests	> Enable > Disable	This parameter enables/disables the behavior whether CanNm_NetworkRequest calls have an effect in Network Mode. If disabled CanNm_NetworkRequest is ignored in Network Mode. If enabled CanNm_NetworkRequest triggers a change from Network Mode to Repeat Message.
CAN NM Extensions		
NM User Data I-PDU	Read Only	Name of the Com PDU where the NM user data can be set if feature 'Com User Data Enabled' is enabled. This parameter must be part of the database. It cannot be configured manually.

Table 6-3 Channel Configuration CAN NM

6.3.2.1 Identity Specific Channel Configuration

If a Multiple ECU Configuration (see also chapter 3.13) is opened in GENy, the settings provided in the following figure and Table 6-4 may be configured for each identity.

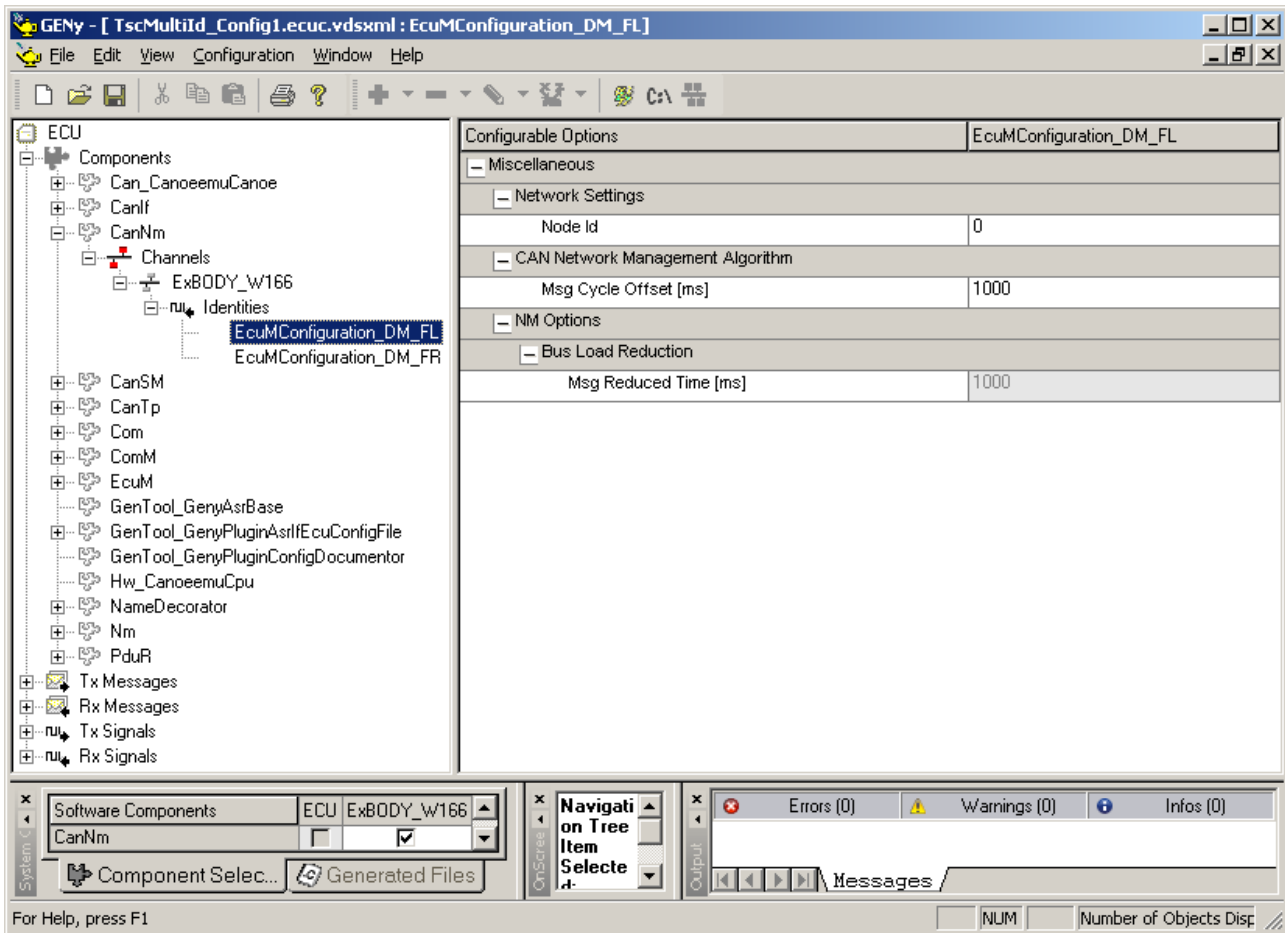


Figure 6-4 Identity-Specific Channel Configuration CAN NM

Configuration options	Value	Description
Miscellaneous		
Network Settings		
Node Id	Read Only	Configure the source node identifier. This value is transmitted within the NM message. Value is set automatically and is read-only if provided by the database.
CAN Network Management Algorithm		
Msg Cycle Offset [ms]	0..65535	Configure the message transmission offset time in the periodic transmission mode (start delay). Value is set automatically and is read-only if provided by the database.
Bus Load Reduction		
Msg Reduced Time [ms]	1..65535	Node specific reduced bus cycle time [ms] in the periodic transmission mode with bus load reduction. Value is set automatically and is read-only if provided by the database.

Table 6-4 Identity-Specific Channel Configuration CAN NM

6.3.3 Timing Restrictions

The following table contains timing constraints that are recommended to ensure that the corresponding algorithms will work correctly:

Timing Parameter	Restrictions
Msg Cycle Offset	< Msg Cycle Time
Msg Timeout Time	< Msg Cycle Time < Msg Reduced Time if Bus Load Reduction Active is enabled
Timeout Time	>> Msg Cycle Time
Repeat Message Time	= 0 if Passive Mode Enabled is enabled
Msg Reduced Time	$\geq (\text{Msg Cycle Time} / 2)$ < Msg Cycle Time
Remote Sleep Indication Time	> NM Message Cycle Time
Pn Reset Time	> Msg Cycle Time < Timeout Time

Table 6-5 CAN NM Timing Restrictions

Note that all timers should be integer multiples of the 'Main Function Period'.

6.3.4 NM Message Structure Configuration Notes

It is possible to enable user data, source node identifier and control bit vector within the message layout (channel configuration) without having the corresponding feature enabled in the component configuration. In such case no API is available to write or read the NM message bytes and the default values will be transmitted (0x00 for source node identifier and control bit vector, 0xFF for user data bytes).

7 Glossary and Abbreviations

Also refer to [8] for a list of common abbreviations and terms.

7.1 Glossary

Term	Description
Confirmation	Notification by the data link layer on asynchronous successful transmission of a CAN message
Identifier	Identifies a CAN message
Indication	Notification by the data link layer on asynchronous reception of a CAN message
Message	One or more signals are assigned to each message.
Signal	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 database, 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 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	A pplication P rogramming I nterface
AUTOSAR	A utomotive O pen S ystem A rchitecture
CAN	C ontroller A rea N etwork
CCL	C ommunication C ontrol L ayer
ComM	C ommunication M anager
CRI	P artial N etwork C luster R equest I nformation
DET	D evelopment E rror T racer
DEM	D iagnostic E vent M anager
DLC	D ata L ength C ode (Number of data bytes of a CAN message)
DLL	D ata link layer
ECU	E lectronic C ontrol U nit
EIRA	E xternal I nternal R equests A ggregated
ERA	E xternal R equests A ggregated
ID	I dentifier (of a CAN message)
IL	I nteraction L ayer
ISR	I nterrupt S ervice R outine
KL15	K lemme 15 (G erman) – C lamp 15 : I gnition O n/ O ff

KL30	Klemme 30 (German) – Clamp 30: Battery voltage
KL31	Klemme 31 (German) – Clamp 31: Ground (GND)
MISRA	Motor Industry Software Reliability Association
NM	Network Management
PDU	Protocol Data Unit
PN	Partial Network / Partial Networking
RAM	Random Access Memory
RI	Reference Implementation (Reference Implementation of the CAN-Driver High Level part)
ROM	Read Only Memory
SchM	Schedule Manager (BSW Scheduler)

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

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

www.vector.com