

# MICROSAR Network Management Interface

## Technical Reference

NM Interface

Version 2.22.00

<b>Authors</b>	Markus Drescher
<b>Status:</b>	Released

## Document Information

### History

Author	Date	Version	Remarks
Oliver Hornung	2006-10-24	1.00.00	ESCAN00018802: Creation of document for AUTOSAR Release 2.1.
Oliver Hornung	2006-12-21	1.01.00	Reworked after User Review.
Oliver Hornung	2007-01-07	1.02.00	ESCAN00019105: Update to actual implementation. Figures updated.
Oliver Hornung	2007-03-09	1.03.00	ESCAN00019899: Database attribute description added.
Oliver Hornung	2007-04-11	1.04.00	ESCAN00020054: Limp Home Indication feature added.
Oliver Hornung	2007-09-24	2.00.00	ESCAN00022463: Update to current Software Specification for AUTOSAR Release 3
Oliver Hornung	2008-02-01	2.01.00	Added coordination extension
Oliver Hornung	2008-03-12	2.02.00	ESCAN00025306: Channel link-time support.
Oliver Hornung	2008-04-02	2.02.01	Adapted coordination extension
Oliver Hornung	2008-04-21	2.03.00	ESCAN00025501: Renaming of Technical Reference
Oliver Hornung	2008-05-02	2.03.01	Corrected Memory Mapping
Markus Schwarz	2008-05-30	2.04.00	Added description of coordination of multiple NMs on one channel
Oliver Hornung	2008-08-20	2.04.01	ESCAN00028990: Added description for synchronous restart / wake-up Adapted configuration description.
Oliver Hornung	2008-08-22	2.05.00	ESCAN00028598: Added support for Communication Manager AUTOSAR Release 2.1
Oliver Hornung	2008-11-28	2.06.00	ESCAN00031694: Adapted Configuration; Reworked Coordination Feature Description
Oliver Hornung	2009-03-31	2.07.00	ESCAN00034278: Chapters partly restructured; Added chapter with service functions that are used by NM Interface.
Oliver Hornung	2009-05-08	2.07.01	ESCAN34966: Reworked Coordinator Extension Description
Oliver Hornung	2009-07-30	2.08.00	ESCAN00036624: Corrected OSEK NM configuration dependency ESCAN00036645: Added description for Communication Control Handling in OSEK NM ESCAN00036652: Added description for Exclusive Areas
Oliver Hornung	2009-11-30	2.09.00	ESCAN00039161: Added feature description 'NM user data via Com'

Oliver Hornung	2010-06-23	2.10.00	<p>ESCAN00040933: Added description for API 'EcuM_GeneratorCompatibilityError'</p> <p>ESCAN00043405: Added 'Check Limp Home' API description</p> <p>ESCAN00043406: Added feature description 'OSEK NM State Change Notifications'</p> <p>ESCAN00043439: Added feature description 'Extended OSEK NM Initialization'</p>
Oliver Hornung	2010-07-22	2.11.00	<p>ESCAN00043708: Added feature description 'Gateway Extension'</p>
Oliver Hornung	2010-08-27	2.12.00	<p>ESCAN00043781: Added feature description 'Passive Coordinator'</p>
Oliver Hornung	2010-09-03	2.13.00	<p>ESCAN00043776: Added feature description 'Car Wakeup'</p> <p>ESCAN00043778: Added feature description 'Set Nm State in User Data'</p> <p>ESCAN00043792: Added feature description 'Partial Networking'</p>
Oliver Hornung	2011-02-17	2.14.00	<p>ESCAN00045723 Removed Partial Networking feature description</p> <p>ESCAN00045745 Adapted description for Communication Control</p> <p>ESCAN00046522 Added description for Fiat Class B Nm Support</p> <p>ESCAN00046777 Added description for Multiple Configuration support</p>
Markus Drescher	2011-08-15	2.15.00	<p>ESCAN00048814 Adapted description for Multiple Configuration support</p> <p>ESCAN00051939 Adapted function prototypes in chapters 6.5.1, 6.6, 6.8</p>
Markus Drescher	2012-01-12	2.16.00	<p>ESCAN00053241 Removed VStdLib and CAN Driver critical section handling</p> <p>ESCAN00056049 Added limitation that the use case of a passive coordinator on AUTOSAR channels is not recommended</p>
Markus Drescher	2012-03-21	2.16.01	<p>ESCAN00055255 Removed limitation introduced with ESCAN00056049</p>
Markus Drescher	2012-06-19	2.17.00	<p>ESCAN00059493 Adapted API descriptions for availability of the functions in chapters 5.6.7.1, 5.6.8.1, 5.6.9.2</p>
Markus Drescher	2012-09-10	2.18.00	<p>ESCAN00055254 Specified use case descriptions more precisely in chapter 3.6, adapted limitation description in chapter 3.1.3</p> <p>ESCAN00059749 Added descriptions of the values written by Nm_GetState (chapter 5.6.1)</p> <p>ESCAN00061120 Documented AUTOSAR deviation of the Nm_Init function prototype (chapters 5.5.1, 3.1.1)</p>

Markus Drescher	2012-11-22	2.19.00	ESCAN00062789 Added description for Fiat Class C Nm Support ESCAN00062966 Removed occurrences of 'State Report Feature Enabled' ESCAN00062967 Added chapter 3.7 ESCAN00063231 Updated Architecture Overview Figure (chapter 2.2.1) ESCAN00063246 Added further deviations from [2] to chapter 6
Markus Drescher	2013-03-05	2.20.00	ESCAN00064216 Added Coordinator Support of Fiat Class B Nm and Fiat Class C Nm
Markus Drescher	2013-04-15	2.21.00	ESCAN00065792 Added deviation description for Nm_ConfigType (chapter 3.1.1) ESCAN00066111 Adapted chapter 3.1.1 ESCAN00066647 Added chapter 'Component History', Adapted chapters 2, 3, merged chapter 7 with chapter 3, swapped chapters 4 and 5, removed 'Compiler Abstraction and Memory Mapping' from chapter 4, various improvements
Markus Drescher	2013-10-01	2.22.00	ESCAN00069710 Adapted chapters that deal with State Change Notifications by OSEK NM ESCAN00070814 Extended description of NM States in 5.6.1

## Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	SRS AUTOSAR Network Management	3.0.0
[2]	AUTOSAR	SWS NM Interface	1.3.0
[3]	AUTOSAR	SWS CAN NM	3.4.0
[4]	AUTOSAR	SWS FlexRay NM	3.3.0
[5]	AUTOSAR	SWS CAN Interface	3.4.0
[6]	AUTOSAR	SWS Communication Manager	2.3.0
[7]	AUTOSAR	Specification of Development Error Tracer	2.2.2
[8]	AUTOSAR	Specification of Diagnostic Event Manager	3.3.0
[9]	AUTOSAR	Specification of BSW Scheduler	1.2.0
[10]	Vector	AN-ISC-8-1118 MICROSAR BSW Compatibility Check	1.0.0
[11]	Vector	Technical Reference MICROSAR Fiat Class B Network Management	1.01.00
[12]	Vector	Technical Reference MICROSAR Fiat Class C Network Management	1.01.00
[13]	Vector	Technical Reference MICROSAR ECUM	2.12.01

**Caution**

We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector's release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

## Contents

<b>1</b>	<b>Component History .....</b>	<b>12</b>
<b>2</b>	<b>Introduction .....</b>	<b>13</b>
2.1	Naming Conventions .....	13
2.2	Architecture Overview .....	14
2.2.1	Architecture of AUTOSAR Software .....	14
2.2.2	Architecture of AUTOSAR Network Management.....	14
<b>3</b>	<b>Functional Description .....</b>	<b>16</b>
3.1	Features.....	16
3.1.1	Deviations Against AUTOSAR 3.2 R 2.....	16
3.1.2	Additions/ Extensions .....	17
3.1.2.1	Filenames of NM Interface.....	17
3.1.2.2	Link-time Support .....	17
3.1.2.3	Link-time Support for Channels .....	18
3.1.2.4	Configurable Service Callback Functions.....	18
3.1.2.5	Development Error Detection.....	18
3.1.2.6	Error Reporting to Diagnostic Event Manager.....	18
3.1.2.7	Memory Initialization .....	18
3.1.2.8	Limp Home Indication.....	18
3.1.2.9	Synchronous Restart / Wake-up Behavior .....	18
3.1.2.10	OSEK NM Support .....	19
3.1.2.11	Selective NM Channel .....	19
3.1.2.12	Coordinator Extension .....	19
3.1.2.13	Coordination of Multiple NMs on One Channel .....	19
3.1.2.14	ComM 2.1 Support .....	19
3.1.2.15	Communication Control Support for OSEK NM.....	19
3.1.2.16	NM User Data via Com.....	19
3.1.2.17	OSEK NM State Change Notifications .....	19
3.1.2.18	OSEK NM Extended Initialization .....	19
3.1.2.19	Gateway Extension.....	19
3.1.2.20	Passive Coordinator .....	19
3.1.3	Limitations.....	20
3.1.3.1	Synchronized OSEK Channel.....	20
3.1.3.2	Extended Coordination Algorithm .....	20
3.2	Adaptation Layer.....	20
3.3	Macro Layer Optimization.....	20
3.4	Network Management Coordination .....	20
3.4.1	Selective Channels .....	21

3.4.2	Synchronous Channels .....	21
3.4.2.1	Application Request Handler .....	22
3.4.2.2	Network Request Handler .....	22
3.4.2.3	State Machine for Shutdown .....	22
3.4.3	Coordination of Multiple NMs within one Channel .....	23
3.5	Coordinator Extension .....	23
3.5.1	Active Coordinator Detection .....	24
3.5.2	Extended Shutdown Algorithm .....	24
3.5.3	Coordinator Loss Detection .....	25
3.5.3.1	Loss Detection Before Shutdown .....	25
3.5.3.2	Loss Detection During Shutdown .....	25
3.5.4	Algorithm Restrictions .....	25
3.5.4.1	Not Covered Scenarios .....	25
3.5.4.2	Synchronous Shutdown Delay .....	26
3.6	Passive Coordinator .....	26
3.6.1	Algorithm Restrictions .....	26
3.7	Provision of the NM State .....	27
3.7.1	Determining the NM State Using Nm_GetState .....	27
3.7.2	Using the 'State Change Ind Enabled' feature .....	27
3.8	OSEK NM Support .....	27
3.8.1	API Mapping .....	28
3.8.2	Callback Mapping .....	28
3.8.3	Limitations .....	29
3.8.4	NM Coordination .....	29
3.8.5	State Change Notifications .....	29
3.8.6	Extended Initialization .....	30
3.9	ComM 2.1 Support .....	31
3.10	Gateway Extension .....	31
3.10.1	Diagnostic Gateway Extension .....	32
3.10.2	NM Gateway Extension .....	32
3.10.2.1	Shutdown Criteria .....	32
3.10.2.2	Wakeup Handling .....	33
3.10.2.3	Coordination of Multiple NMs within One Channel .....	33
3.11	Car Wakeup .....	34
3.12	Set Nm State in User Data .....	34
3.13	Multiple ECU Support .....	34
3.14	Multiple Configuration Support .....	35
3.15	Fiat Class B NM and Fiat Class C NM Support .....	36
3.15.1	API Mapping .....	36
3.15.2	Callback Mapping .....	36
3.15.3	Limitations .....	36

3.16	Error Handling .....	37
3.16.1	Development Error Detection .....	37
3.16.2	Production Code Error Reporting .....	38
<b>4</b>	<b>Integration .....</b>	<b>39</b>
4.1	Files .....	39
4.2	Include Structure .....	40
4.3	Version Changes .....	40
4.4	Initialization .....	40
4.5	Main Function .....	41
4.6	Critical Sections .....	42
4.6.1	Critical Section Codes .....	42
<b>5</b>	<b>API Description .....</b>	<b>44</b>
5.1	API Categories .....	44
5.2	Data Types .....	44
5.3	Global Variables .....	44
5.4	Global Constants .....	45
5.4.1	AUTOSAR Specification Version .....	45
5.4.2	Component Versions .....	45
5.4.3	Vendor and Module ID .....	45
5.5	Administrative Functions Provided by NM Interface .....	46
5.5.1	Nm_Init: Initialization of NM Interface .....	46
5.5.2	Nm_MainFunction: Main Function of the NM Interface .....	47
5.6	Service Functions Provided by NM Interface .....	48
5.6.1	Nm_GetState: Get the State of the Network Management .....	48
5.6.2	Nm_GetVersionInfo: Version Information API .....	50
5.6.3	Nm_PassiveStartUp: Wake-up Network Management .....	51
5.6.4	Wake-up Registration .....	52
5.6.4.1	Nm_NetworkRequest: Request the Network .....	52
5.6.4.2	Nm_NetworkRelease: Release the Network .....	53
5.6.5	Communication Control Service .....	54
5.6.5.1	Nm_DisableCommunication: Disable NM Message Transmission ...	54
5.6.5.2	Nm_EnableCommunication: Enable NM Message Transmission .....	55
5.6.6	User Data Handling .....	56
5.6.6.1	Nm_SetUserData: Set User Data .....	56
5.6.6.2	Nm_GetUserData: Get User Data .....	57
5.6.6.3	Nm_GetPduData: Get NM Pdu Data .....	58
5.6.7	Node Detection .....	59
5.6.7.1	Nm_RepeatMessageRequest: Set Repeat Message Request Bit ....	59
5.6.7.2	Nm_GetNodeIdentifier: Get Source Node Identifier .....	60



5.6.7.3	Nm_GetLocalNodeIdentifier: Get Local Source Node Identifier .....	61
5.6.8	Remote Sleep Indication .....	62
5.6.8.1	Nm_CheckRemoteSleepIndication: Check for Remote Sleep Indication .....	62
5.6.9	Vector Extensions .....	63
5.6.9.1	Nm_InitMemory: Memory Initialization .....	63
5.6.9.2	Nm_RequestBusSynchronization: Request Bus Synchronization ....	64
5.6.9.3	Nm_CheckLimpHomeIndication: Check the Limp Home Status.....	65
5.6.9.4	Nm_SetGwRemoteSleepFilter: Set Remote Sleep Filter .....	66
5.6.9.5	Nm_SetGwRemoteWakeupFilter: Set Remote Wakeup Filter .....	67
5.6.9.6	Nm_WakeupNotification: Wakeup Notification .....	68
5.6.9.7	Nm_SetDiagGwReqId: Set Requested Diagnostic Node Identifier...	69
5.7	Service Functions Used by NM Interface.....	70
5.8	Callback Functions Provided by NM Interface .....	73
5.8.1	Nm_NetworkStartIndication: Network Start Indication .....	73
5.8.2	Nm_NetworkMode: Network Mode Indication.....	74
5.8.3	Nm_PrepareBusSleepMode: Prepare Bus Sleep Mode Indication .....	74
5.8.4	Nm_BusSleepMode: Bus Sleep Mode Indication .....	75
5.8.5	Nm_RemoteSleepIndication: Remote Sleep Indication .....	75
5.8.6	Nm_RemoteSleepCancellation: NM Remote Sleep Cancellation .....	76
5.8.7	Nm_PduRxIndication: NM Message Reception Indication .....	76
5.8.8	Nm_RepeatMessageIndication: Repeat Message Request Indication.....	77
5.8.9	Nm_StateChangeNotification: State Change Notification .....	78
5.8.10	Nm_TxTimeoutException: Transmission Timeout Exception .....	79
5.8.11	Nm_CarWakeUpIndication: Car Wake-up Indication .....	79
5.8.12	Vector Extensions .....	80
5.8.12.1	Nm_ActiveCoordIndication: Indication of an Active Coordinator Bit.....	80
5.8.12.2	Nm_LimpHomeIndication: Limp Home Indication .....	81
5.8.12.3	Nm_LimpHomeCancelation: NM Remote Sleep Cancellation.....	82
5.8.12.4	Nm_GwPduRxIndication: Extended Gateway Message Indication ..	83
5.8.12.5	OSEK NM Callback Functions .....	83
5.9	Callback Functions Used by NM Interface.....	85
5.9.1	Service Callback Functions of NM Interface .....	85
5.9.2	Configurable Service Callback Functions .....	85
5.9.3	Additional Service Callback Functions of NM Interface.....	86
5.9.3.1	Callbacks for Limp Home Indication.....	86
5.9.3.2	Callback for Diagnostic Gateway Extension.....	86
5.9.3.3	Generator Compatibility Error .....	86

## 6 Configuration..... 87

6.1	Configuration Variants .....	87
6.2	Configuration in Data Base.....	87
6.3	Configuration with GENy .....	88
6.3.1	Component Configuration .....	89
6.3.2	Channel Configuration.....	93
6.3.3	NM Coordination Restrictions.....	96
6.3.3.1	Needed NM Features .....	96
6.3.3.2	NM Main Function Cycle Time .....	96
6.3.3.3	Shutdown Timing .....	96
6.3.3.3.1	Shutdown Time Calculation .....	97
6.3.3.4	Synchronized Channels.....	97
6.3.4	NM Extended Coordination Restrictions .....	98
6.3.5	OSEK NM Configuration .....	98
6.3.6	Diagnostic Gateway Extension Configuration Restrictions .....	100
6.3.7	NM Gateway Extension Configuration Restrictions .....	100
6.3.8	Fiat Class B NM / Fiat Class C NM Configuration Restrictions .....	101
<b>7</b>	<b>Glossary and Abbreviations .....</b>	<b>102</b>
7.1	Abbreviations .....	102
7.2	Glossary.....	103
<b>8</b>	<b>Contact.....</b>	<b>104</b>

## Illustrations

Figure 2-1	Architecture of the AUTOSAR Stack .....	14
Figure 2-2	Interface to Adjacent Modules of the NM Interface .....	15
Figure 3-1	Request/Release Handling for Selective Channels .....	21
Figure 3-2	Request/Release Handling for Synchronous Channels .....	22
Figure 3-3	NM Coordination Scenario with 2 Coordinators .....	24
Figure 4-1	Include Structure of Nm Interface .....	40
Figure 6-1	Component Selection in GENy .....	88
Figure 6-2	Component Configuration NM Interface .....	89
Figure 6-3	Channel Configuration NM Interface .....	94
Figure 6-4	OSEK NM Configuration .....	99

## Tables

Table 1-1	Component history .....	12
Table 2-1	Module Prefixes .....	13
Table 2-2	Additional Supported Network Managements .....	13
Table 3-1	Supported AUTOSAR standard conform features .....	16
Table 3-2	Features provided beyond the AUTOSAR standard .....	17
Table 3-3	API Mapping for OSEK NM Channels .....	28
Table 3-4	Callback Mapping for OSEK NM Channels .....	28
Table 3-5	OSEK NM State Change Notification Arguments: nmPreviousState and nmCurrentState .....	30
Table 3-6	OSEK NM Extended Initialization States .....	31
Table 3-7	Nm State Change Signal Values .....	34
Table 3-8	Service IDs .....	38
Table 3-9	DET Error Codes .....	38
Table 4-1	File Names of the NM Interface .....	39
Table 4-2	Critical Section Codes .....	43
Table 5-1	Global NM Types .....	44
Table 5-2	NM Interface Data Types .....	44
Table 5-3	Specification Version API Data .....	45
Table 5-4	Component Version API Data .....	45
Table 5-5	Vendor and Module ID .....	45
Table 5-6	Services Used by the NM Interface .....	72
Table 5-7	OSEK NM Callback Functions .....	84
Table 6-1	Database Attributes .....	87
Table 6-2	Component Configuration NM Interface .....	93
Table 6-3	Channel Configuration NM Interface .....	95
Table 6-4	Required Configuration Items for the NM Coordination .....	96
Table 6-5	Common Shutdown Time Calculation .....	97
Table 6-6	Required Items of Involved Components for Proper Usage of the Coordinator Extension .....	98
Table 6-7	Required Configuration Items for Diagnostic Gateway Extension .....	100
Table 6-8	Required Configuration Items for NM Gateway Extension .....	100
Table 6-9	Required Configuration Settings for Fiat Class B NM / Fiat Class C NM Usage .....	101
Table 7-1	Abbreviations .....	102
Table 7-2	Glossary .....	103

## 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 2.1
1.03.00	Added Limp Home Indication Feature
2.00.00	Adaption for AUTOSAR 3
2.01.00	Added Coordinator Extension (Synchronization Support) Feature
2.03.00	Added Channel Link-Time Support
2.08.00	Added AUTOSAR 2.1 Communication Manager Support
2.14.00	Added 'User Data via Com' Feature
2.15.00	Added OSEK NM State Change Notification Support Added OSEK NM Extended Initialization Support
2.16.00	Added 'Gateway Extension' Feature
2.17.00	Added 'Passive Coordinator' Feature
2.18.00	Added 'Car Wake Up' Feature Added 'State Report' Feature
2.19.00	Added Fiat Class B NM Support Added Multiple Configuration Support
2.26.00	Added Fiat Class C NM Support
2.27.00	Added Coordinator Support for Fiat Class B NM and Fiat Class C NM
2.29.00	Adapted OSEK NM State Change Notification Support in order to support the NM message being the first message on the bus

Table 1-1 Component history

## 2 Introduction

This document describes the concept, features, API and the configuration of the AUTOSAR Network Management Interface. Also the integration of the Network Management (NM) into the Vector CANbedded stack is covered by this document. The FlexRay Network Management is not covered by this document.

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 FlexRay NM.

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

### 2.1 Naming Conventions

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

In case of callback functions to the COM Manager the service starts with the term 'ComM\_Nm\_' to make clear it is a callback from the Network Management. E.g. 'ComM\_Nm\_' denominates a service, which is implemented within the COM Manager and called by NM Interface.

#### Naming conventions

Nm_	Services of NM Interface (Nm).
CanNm_	Services of CAN NM (CanNm).
FrNm_	Services of FlexRay NM (FrNm).
BusNm_	Services of the corresponding bus-specific NM
ComM_Nm_	Callback services of NM Interface that are implemented within the Communication Manager (ComM).
Det_	Services of Development Error Tracer (Det).
Dem_	Services of Diagnostic Event Manager (Dem).

Table 2-1 Module Prefixes

The term 'bus-specific NM' does not only refer to the AUTOSAR CAN NM and FlexRay NM but also to the additionally supported NMs listed in the following table with their according bus type.

Bus Type	Additional supported bus-specific NM (Short name)
CAN	OSEK NM (NmOsek)
CAN	Fiat Class B (NmFiatB)
CAN	Fiat Class C (NmFiatC)

Table 2-2 Additional Supported Network Managements

## 2.2 Architecture Overview

### 2.2.1 Architecture of AUTOSAR Software

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

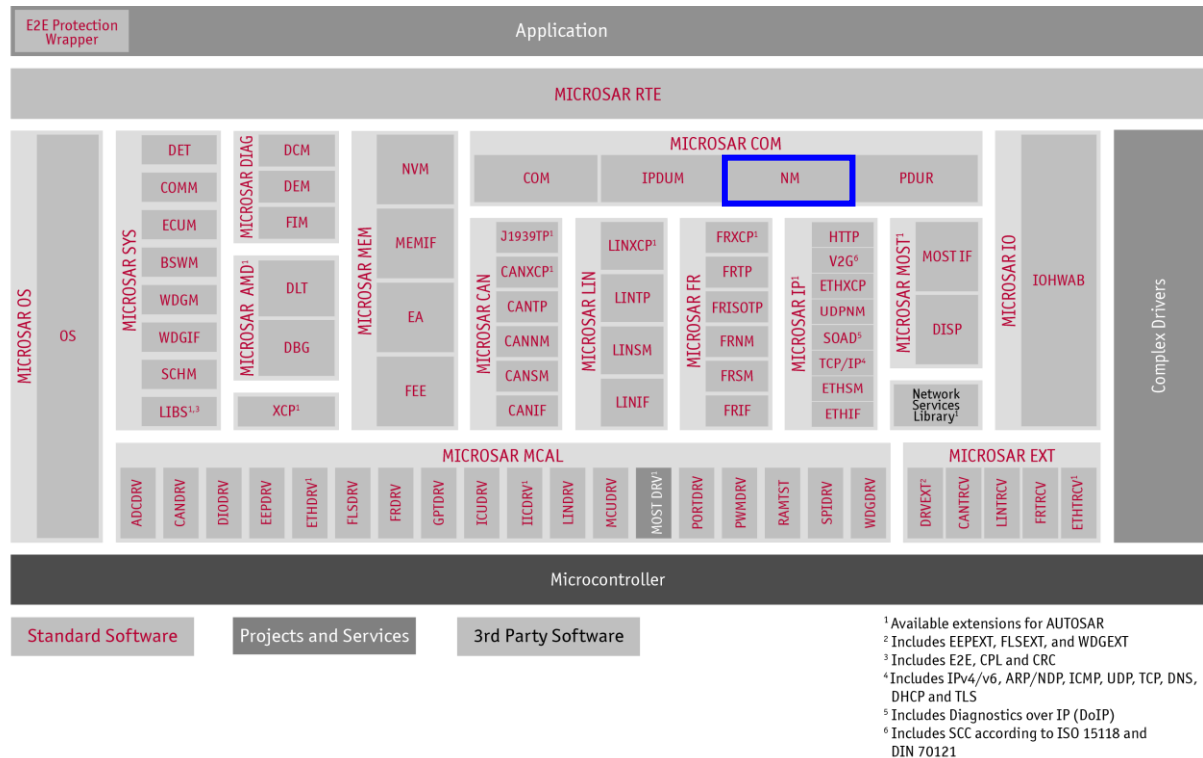


Figure 2-1 Architecture of the AUTOSAR Stack

### 2.2.2 Architecture of AUTOSAR Network Management

The AUTOSAR Network Management consists of three modules:

- > NM Interface
- > CAN NM<sup>1</sup>
- > FlexRay NM<sup>1</sup>

The NM Interface schedules function calls from the Communication Manager (ComM) to the respective module for each channel, e.g. for a CAN channel the corresponding CAN NM function will be called. It also provides the interface to ComM. NM Interface exclusively interacts with these modules.

The communication bus specific functionality is incorporated in the corresponding bus-specific NM. Refer to the corresponding Technical References of the bus-specific network managements for more information.

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

<sup>1</sup> Not covered by this document.

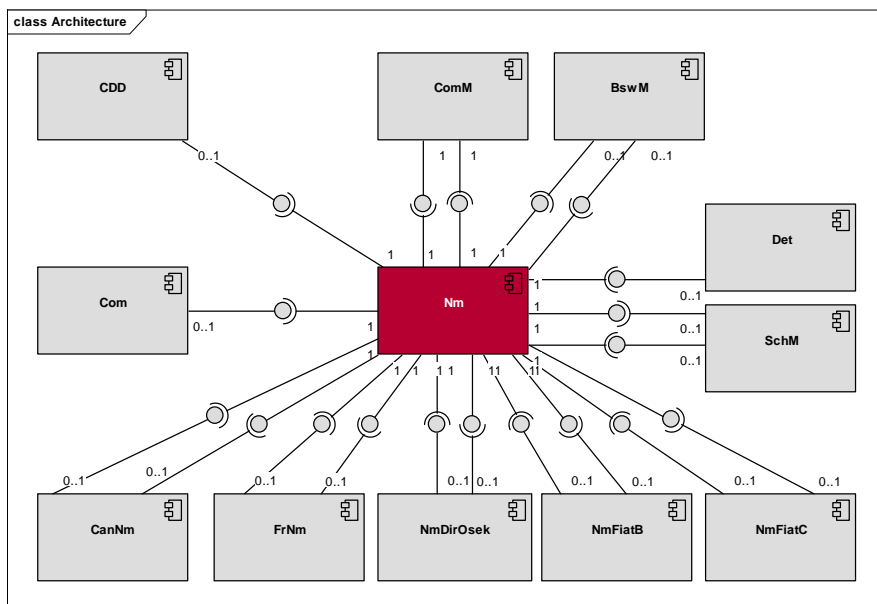


Figure 2-2 Interface to Adjacent Modules of the NM Interface

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 NM Interface has no service ports, the NM Interface cannot be accessed via RTE by the application.

## 3 Functional Description

### 3.1 Features

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 Nm.

The AUTOSAR standard functionality is specified in [2], the corresponding features are listed in the table

► Table 3-1 Supported AUTOSAR standard conform features

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

► Table 3-2 Features provided beyond the AUTOSAR standard

The following features specified in [2] are supported:

Supported AUTOSAR Standard Conform Features
Controlled transition of all ECU's to bus-sleep mode and vice versa.
User Data Handling
Node Detection
Remote Sleep Indication
Passive Mode Support
Coordination Functionality
Coordinator Synchronization Support
OSEK NM Support
NM User Data via Com
Car Wake-Up
Set Nm State in User Data

Table 3-1 Supported AUTOSAR standard conform features

#### 3.1.1 Deviations Against AUTOSAR 3.2 R 2

All deviations of the user requirements are documented within the System Requirements Specification [1]. The implementation is based on the System Requirements Specification.

The following list provides the limitations of the NM Interface:

- > Following additional dependencies between configuration parameters are added to avoid bad configurations:
  - `NM_COM_CONTROL_ENABLED` must be set to `OFF` for passive nodes.



- `NM_NODE_DETECTION_ENABLED` must be set to `OFF` for passive nodes.

Furthermore, the function prototype of `Nm_Init` deviates from the original AUTOSAR requirement in [2]. See also chapter 5.5.1.

The function prototype of `Nm_ActiveCoordIndication` differs from the original AUTOSAR requirement in [2]. Refer to chapter 5.8.12.1 for details.

The configuration parameter `NmGlobalCoordinatorTime` mentioned in [2] is called `NmCoordinationShutdownTimer` instead.

The data type definition of `Nm_ConfigType` is different to the definition mentioned in chapter 8.2.5 of [2].

### 3.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
Limp Home Indication
ComM 2.1 Support
Gateway Extension
Multiple ECU Support
Multiple Configuration Support
Fiat Class B Support
Fiat Class C Support

Table 3-2 Features provided beyond the AUTOSAR standard



#### Info

Additional non-AUTOSAR features are only available if they are explicitly ordered by the customer.

These and the following extensions of the NM Interface software specification ([2]) are available within the Network Management embedded software components. If required, the extensions have to be enabled during configuration time.

#### 3.1.2.1 Filenames of NM Interface

In order to allow the integration of NM Interface together with OSEK NM the pre-compile configuration files are renamed to '`NmIf_Cfg`' instead of '`Nm_Cfg`' in case of a non AUTOSAR CAN Driver is used and the OSEK NM Support is enabled.

For more information refer to chapter 4.1 'Files' for an overview of the file names.

#### 3.1.2.2 Link-time Support

Up to now, the software specifications of the NM Interface only specify a pre-compile variant. To allow the usage of object code or libraries the link-time variant is available.

For more information refer to chapter 6 'Configuration'.

### 3.1.2.3 Link-time Support for Channels

In [2] 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 add or remove channels at link-time. Therefore the NM Interface implementation supports link-time changes of the channels.

Note that the bus-specific NM types are not link-time able, i.e. only channels of already available bus-specific NMs can be added or removed (except the last channel of one bus-specific NM type) at link-time.

### 3.1.2.4 Configurable Service Callback Functions

As the handling for some optional callback functions from the underlying bus-specific NMs is not specified the NM Interface provides for them a configurable interface to the upper layer.

For more information refer to chapter 5.9.2 'Configurable Service Callback Functions'.

### 3.1.2.5 Development Error Detection

As the specified development errors in [2] are not sufficient the implementation will report additional development errors.

For more information refer to chapter 3.16.1 'Development Error Detection'.

### 3.1.2.6 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 `NM_PROD_ERROR_DETECT` has been added.

For more information refer to chapter 6.3.1 'Component Configuration'.

### 3.1.2.7 Memory Initialization

Not every start-up code of embedded targets and neither CANoe reinitialize all variables correctly. It thus may happen that the state of a variable may be not initialized, when this should be the case. It therefore needs to be initialized explicitly by the corresponding module.

For more information refer to chapter 5.6.9.1 'Nm\_InitMemory: Memory Initialization'.

### 3.1.2.8 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. Please note that this feature is OEM-specific and not available by default. If available it can be activated or deactivated in the configuration.

For more information refer to chapter 6 'Configuration'.

### 3.1.2.9 Synchronous Restart / Wake-up Behavior

The state machine for the synchronous shutdown must consider an abortion of the shutdown due to communication need on any of the coordinated channels. For keeping the algorithm simple the state machine does also cover the use case where at least one or all channels are asleep. Therefore this implementation also supports a synchronous restart and wake-up behavior.

For more information refer to chapter 3.4.2.3 'State Machine for Shutdown'

#### **3.1.2.10 OSEK NM Support**

OSEK NM Support is independently available from the NM Coordination Support.

#### **3.1.2.11 Selective NM Channel**

AUTOSAR only allows the NM Coordination for all channels of the ECU. To allow separate handling of channels that must not have a synchronized shutdown, a channel can be enabled to be selective. Selective NM Channels are not considered for the NM Coordination algorithm.

For more information refer to chapter 3.4.1 'Selective Channels'.

#### **3.1.2.12 Coordinator Extension**

For supporting scenarios where more than one coordinator is connected to the same channel this extended coordinator algorithm can be used.

For more information refer to chapter 3.5 'Coordinator Extension'.

#### **3.1.2.13 Coordination of Multiple NMs on One Channel**

The NM Coordinator is extended to work with different NMs on the same network.

For more information refer to chapter 3.4.3 'Coordination of Multiple NMs within one Channel'.

#### **3.1.2.14 ComM 2.1 Support**

The NM supports integration together with an AUTOSAR 2.1 Communication Manager.

For more information refer to chapter 3.9 'ComM 2.1 Support'.

#### **3.1.2.15 Communication Control Support for OSEK NM**

The communication control feature is also available for OSEK NM. Refer to chapter 3.8.1 'API Mapping' for the mapping to the corresponding OSEK NM functions.

#### **3.1.2.16 NM User Data via Com**

The MICROSAR NM supports the possibility to write the NM user data via Com signals. For details please refer to the Technical References of CAN NM and FlexRay NM.

#### **3.1.2.17 OSEK NM State Change Notifications**

State Change Notifications are provided in case of a state change of OSEK NM. Refer to chapter 3.8.5 'State Change Notifications' for further information.

#### **3.1.2.18 OSEK NM Extended Initialization**

OSEK NM can be initialized with initialization modes other than only 'Sleep'. See chapter 3.8.6 'Extended Initialization' for details.

#### **3.1.2.19 Gateway Extension**

The Gateway Extension contains two sub-features, Diagnostic Gateway Extension and Network Management Gateway Extension. For more information refer to chapter 3.10 'Gateway Extension'.

#### **3.1.2.20 Passive Coordinator**

The feature 'Passive Coordinator' allows in addition to the feature 'Coordinator Extension' having OSEK NM channels connected to more than one coordinator.

For details refer to chapter 3.6 'Passive Coordinator'.

### 3.1.3 Limitations

#### 3.1.3.1 Synchronized OSEK Channel

Since it is only possible to synchronize the NM Coordination to one OSEK channel this channel has to explicitly chosen in the configuration (see also chapter 6 'Configuration').

#### 3.1.3.2 Extended Coordination Algorithm

A coordinator in passive mode behaves only correct on all channels if all passively coordinated channels are connected to the same active coordinator.

For more information refer to chapter 3.5.4 'Algorithm Restrictions'.

The function `Nm_CoordReadyToSleepIndication` specified in [2] is not provided by Nm. The indication for sleep readiness shall be provided instead to Nm by `Nm_ActiveCoordIndication`.

The functions `CanNm_SetSleepReadyBit` and `FrNm_SetSleepReadyBit` are not used by Nm. Instead, the functions `CanNm_SetCoordBits` and `FrNm_SetCoordBits` are used instead to indicate whether the coordinator is ready to sleep or not.

The following chapters provide a detailed description of the NM functionality.

## 3.2 Adaptation Layer

The NM Interface shall act as a bus-independent adaptation layer between the Communication Manager and the bus-specific network managements. Currently supported are the CAN and the FlexRay NM. AUTOSAR plans to also support LIN NM in the future. Mainly for the coordination with channels where OSEK NM is running the NM Interface supports optionally OSEK NM (refer to chapter 3.8 'OSEK NM Support'). Additionally the usage of Fiat Class B NM and Fiat Class C NM is supported (see also chapter 3.15 'Fiat Class B NM and Fiat Class C NM Support').

## 3.3 Macro Layer Optimization

If the NM Interface provides no other functionality than mapping API calls from the higher layer to exclusively AUTOSAR CAN NM, exclusively AUTOSAR FlexRay NM, exclusively MICROSAR Fiat Class B NM or exclusively MICROSAR Fiat Class C NM, the complete layer can be optimized to be a macro layer in order to save precious resources (ROM, RAM and CPU load). Also refer to chapter 6 'Configuration'.

Please note that Macro Layer optimization can only be enabled in pre-compile configurations and in configurations where the NM Interface does not provide own algorithms (e.g. Coordinator or Gateway algorithm).

## 3.4 Network Management Coordination

The feature 'NM coordination' is optional and can be used in ECUs with gateway functionality and the requirement that at least two busses of the gateway have to be shut down synchronously. The gateway keeps all connected networks awake as long as at least one NM node is not ready for sleep. This could be either the gateway itself or any node on any synchronous network.

To allow additionally that some networks can still be handled asynchronously each channel can be configured to if it is synchronous or selective (refer also chapter 6.3.2 'Channel Configuration').

### 3.4.1 Selective Channels

Selective channels behave like un-coordinated channels. Network request/releases are directly processed and passed on to the underlying bus-specific NM. The following figure illustrates this handling for selective channels:

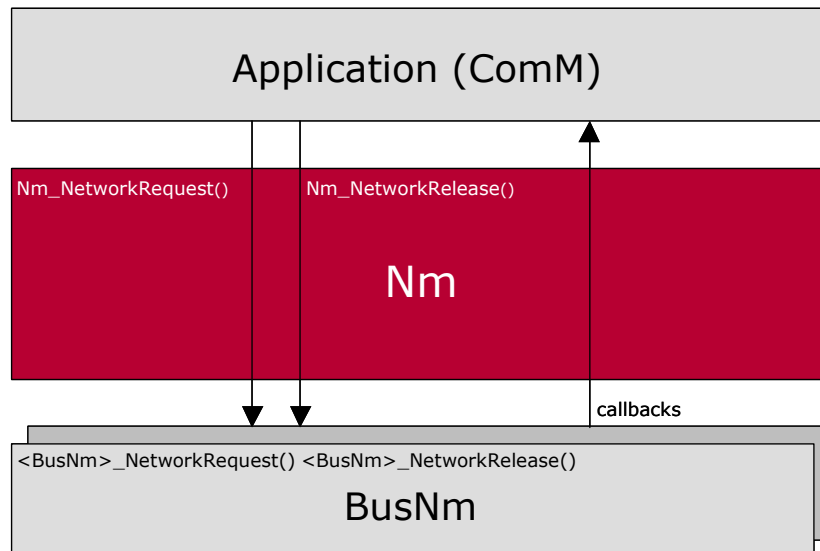


Figure 3-1 Request/Release Handling for Selective Channels

### 3.4.2 Synchronous Channels

Synchronous channels are coordinated. Coordination is done by the coordinator. The coordinator is based on a cyclic task function (see chapter 4.5 'Main Function').

The coordinator consists of three elements

- > Application Request Handler
- > Network Request Handler
- > Statemachine for Shutdown

The following figure provides an overview of the network request/release handling for synchronous channels:

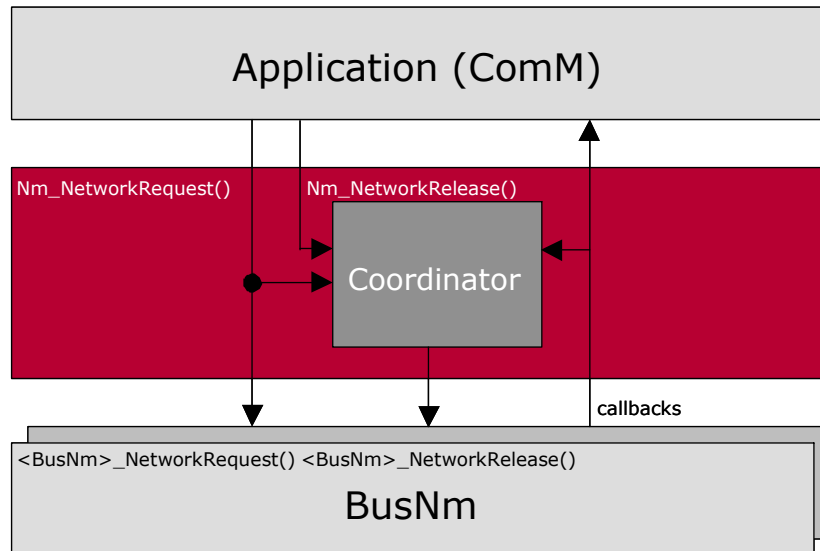


Figure 3-2 Request/Release Handling for Synchronous Channels

### 3.4.2.1 Application Request Handler

Network requests are directly performed. The request is also notified to the Application Request Handler.

Network releases for synchronous channels are redirected to the Application Request Handler. The network release is not directly performed.

The Application Request Handler combines the pending requests for all synchronous channels. The result influences the state machine for shutdown.

### 3.4.2.2 Network Request Handler

The bus-specific NM of each synchronous network checks if any of the other nodes on the network is not ready for sleep. This information is polled by the Network Request Handler.

If all nodes on a network signal readiness for sleep, the network is said to be Remote Sleep Indication.

The Network Request Handler combines the status of the Remote Sleep Indication of all synchronous channels. The result influences the state of the Statemachine for Shutdown.

### 3.4.2.3 State Machine for Shutdown

The coordinator checks the communication status of all synchronous channels. The communication status is derived from the status of the Application Request Handler and the Network Request Handler.

If no communication is required, the shutdown procedure starts. A network-specific delay time is determined for each synchronous channel. The timing depends on the used types of channels and their timings.

**Caution**

The shutdown time depends on the maximum shutdown timing of all channels.

If on one synchronous channel OSEK NM is running, the shutdown algorithm is based on the timing of this network, i.e. the size of the logical NM ring.

**Caution**

If multiple synchronous channels with OSEK NM are used, synchronous shutdown cannot be guaranteed. The timing has to be based on one of the NM OSEK channels. This decision is done by explicitly setting the OSEK NM channel that should be used as base for the synchronous shutdown in the configuration (see chapter 3.8.4 'NM Coordination').

If the coordinator detects any need for communication during or after the shutdown procedure, it ensures that all synchronous channels are restarted again. In case a channel is already asleep this channel will be woken up by calling

```
void ComM_Nm_RestartIndication (
    const NetworkHandleType nmChannelHandle ) (5.9.1)
```

Therefore the coordination algorithm also covers the synchronous restart/wake-up use case.

### 3.4.3 Coordination of Multiple NMs within one Channel

The coordination feature supports an AUTOSAR CAN NM and an OSEK NM running both on the same network. Both NMs are synchronized for shutdown as far as possible.

Externally this channel behaves like an AUTOSAR CAN NM channel, i.e. any API access or callback notification is related to the CAN NM. The OSEK NM is handled only internally.

**Info**

Channels with multiple NMs need to be coordinated, i.e. the coordination feature must be enabled and the channel must be synchronous.

## 3.5 Coordinator Extension

For supporting more than one coordinator connected to the same channel it is necessary to extend the coordination algorithm to prevent that the coordinators keep themselves awake one another. It has to be ensured that only one coordinator coordinates this

channel actively. All other coordinators act as passive coordinators and just synchronize their other channels with this one.

The following figure shows such a scenario:

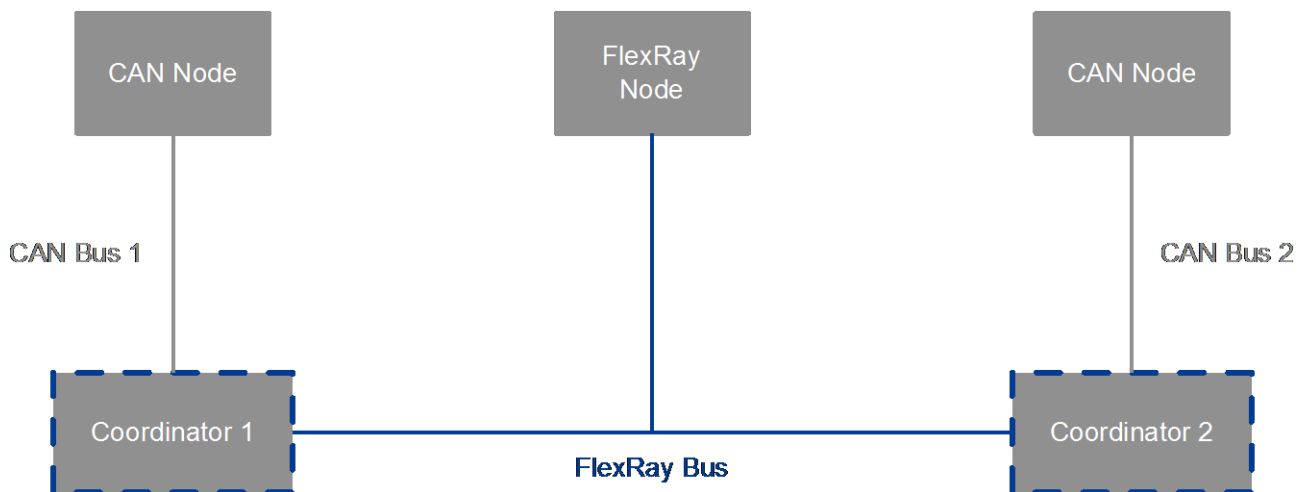


Figure 3-3 NM Coordination Scenario with 2 Coordinators

### 3.5.1 Active Coordinator Detection

Each Coordinator needs for the extension a priority ID that is unique on each channel. For the configuration of this ID please refer to chapter 6.3.1 'Component Configuration'. This ID is transmitted within the NM message and the coordinator with the highest ID will take over the coordination. All other coordinators reset their delay timer and switch to passive coordination for this channel. In passive coordination mode the coordinators won't transmit their priority ID.



#### Caution

As this priority is transmitted within the Control Bit Vector of the NM message and reserves only just two bits at most 3 coordinators at one channel are allowed.

### 3.5.2 Extended Shutdown Algorithm

Coordinators that act passively on one channel release the communication on this channel as soon as all their other channels and their applications are ready to sleep. Those coordinators then wait for the 'Announce Sleep Bit'. This bit is set by the active coordinator as soon as he detects readiness for sleep on all channels (including his application) and he starts the shutdown. This bit is forwarded as soon as received on the passively coordinated channel to all actively coordinated AUTOSAR NM channels and the shutdown is started.

If at least one channel on the coordinator node communication is needed again the shutdown is aborted. The active coordinator on this channel removes the sleep bit and sends its NM message with its priority. All passive coordinators recognize that the sleep bit has been deleted and abort their shutdown also.



Synchronous shutdown timing for all channels is achieved by setting the 'NM Coordination Shutdown Timer' (see chapter 6.3.1 'Component Configuration') to the same value on all coordinator nodes (refer also to chapter 6.3.4 'NM Extended Coordination Restrictions').

### 3.5.3 Coordinator Loss Detection

Coordinators that act passively on a channel must be able to detect if the active coordinator fails. In that case the coordinator with the next highest priority will take over the coordination. This detection mechanism is time based with two different timings depending whether the active coordinator already started the synchronous shutdown or not. Both scenarios are discussed in the corresponding sub-chapters.

#### 3.5.3.1 Loss Detection Before Shutdown

While the coordination is ongoing due to communication is needed on at least one channel the active coordinator sends its message with its priority normally. Therefore the timeout is set to two times the message cycle time<sup>2</sup> for having a short reaction time on a possible loss of the active coordinator.

#### 3.5.3.2 Loss Detection During Shutdown

When the active coordinator started the shutdown phase it will release communication and enter Ready Sleep state. It then stops sending its NM message<sup>3</sup> with priority and sleep bit set. Therefore the timeout is set to the shutdown time of this channel (i.e. the time from entering Ready Sleep state until Bus Sleep is reached). If this time elapses and the channel has not been shut down but also was not restarted by the active coordinator a further node has communication need and keeps the bus awake while the active coordinator failed.

### 3.5.4 Algorithm Restrictions

In this chapter restrictions of the usage of this extension are discussed. Configuration restrictions for this extension are discussed in chapter 6.3.4 'NM Extended Coordination Restrictions'.

#### 3.5.4.1 Not Covered Scenarios

This extension of the coordination algorithm cannot handle scenarios where more than one coordinator is connected to a non-AUTOSAR channel (i.e. a channel where OSEK NM is running).

The algorithm is not capable to synchronize shutdown correctly if the coordinator acts passively on more than one channel due to the conflict that could appear when on one of those channels the 'Announce Sleep Bit' is already set but not on another channel. In such a situation the coordinator keeps all actively coordinated channels awake until the sleep bit is received on all passively coordinated channels.

Apart from this restriction are scenarios where all passively coordinated channels are connected to the same active coordinator, e.g. backup coordinators. Here the sleep indication bit from the active coordinator will appear almost at the same time on all passive channels.

---

<sup>2</sup> Note that in case of FlexRay NM the data cycle time is taken as message cycle time due to the Control Bit Vector information is relevant for the loss detection.

<sup>3</sup> Note that in case of FlexRay data message is send in the static part NM message data is still transmitted in Ready Sleep. The loss detection mechanism in this case works also.

### 3.5.4.2 Synchronous Shutdown Delay

Coordinators that act passively on one channel will have a small delay on the start of their synchronous shutdown. This delay depends on the time difference between the point in time where the active coordinator sets the 'Announce Sleep Bit' and the point in time where this bit is then transmitted and received.

If the Coordinator that acts passively on one channel has to coordinate on another channel an OSEK NM actively then this delay will be increased for this OSEK NM channel by the time until the OSEK NM token is available again in the Coordinator node.

## 3.6 Passive Coordinator

The 'Passive Coordinator' feature is an add-on of the 'Coordinator Extension' (refer to chapter 3.5 'Coordinator Extension'). A channel that is configured for passive coordination behaves always as if a coordinator with a higher priority is active, regardless of the own priority and whether any other coordinator is active on that channel or not. Therefore no timeout monitoring is performed and loss detection of the active coordinator is not possible on such a channel.

This feature allows scenarios where a channel shall be regarded for coordination but shall not be kept awake if application and all actively coordinated channels are ready to sleep. It can be used for example to have more than one coordinator on non-AUTOSAR networks (i.e. OSEK NM) or for having more than three coordinators connected to the same bus.



#### Note

Vector recommends using a passive coordinator configuration on AUTOSAR networks only if more than three coordinators are connected to the same bus.

On a channel that is configured for passive coordination the 'Announce Sleep Bit' may not be available. Therefore the coordinator accepts additionally the transition to 'Bus Sleep' on this channel as trigger that the channel is ready for shutdown. The shutdown of all actively coordinated channels is started when all passively coordinated channels indicate readiness for shutdown.



#### Caution

On channels that are not configured to passive coordination the 'Announce Sleep Bit' is still required on passively coordinated as otherwise the timeout monitoring will prevent that the coordinator will shut down.

### 3.6.1 Algorithm Restrictions

When not having the 'Announce Sleep Bit' as trigger for shutdown readiness on all passively coordinated channels, the shutdown cannot be synchronized exactly to the

actively coordinated channels as their shutdown will start when the passively coordinated channel is already asleep.

### 3.7 Provision of the NM State

The NM State can be determined either by using the function

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

 (5.6.1)

or by activating the feature 'State Change Ind Enabled'. Both of these features are implemented according to [2].

Note that the NM state may also be optionally reported into Com signals using the 'State Report Enabled' feature (refer to chapter 3.12 for details).

#### 3.7.1 Determining the NM State Using Nm\_GetState

If Nm\_GetState (see also chapter 5.6.1) has been called, the current NM state and the current NM mode of the bus-specific Nm (e.g. CanNm, FrNm) associated with the system channel index nmChannelHandle are written to the passed pointer variables. Possible state and mode values that are returned into these variables can be seen in the definition of Nm\_StateType / Nm\_ModeType in NmStack\_Types.h.

#### 3.7.2 Using the 'State Change Ind Enabled' feature

The 'State Change Ind Enabled' feature enables a callback that is called each time the NM state has been changed. To use this feature, activate the 'State Change Ind Enabled' setting in the module settings of Nm in GENy (see also chapter 6.3.1 'Component Configuration'). Furthermore, enter the name of the function that shall be called in case of a NM state change into the 'State Change Indication Callback' field and provide the file name of the header file that contains the function prototype of this function in the 'Nm Callbacks Prototype Header' setting.

Note that the function prototype of the function given in 'State Change Indication Callback' has to be the same as (except the function name) Nm\_StateChangeNotification (refer to chapter 5.8.9 for details).

### 3.8 OSEK NM Support

The optional feature 'OSEK NM Support' allows controlling channels where OSEK NM is running. If this feature is enabled each channel can be configured as an OSEK NM channel.



#### Note

If GENy is used for the configuration the mapping for each channel to the correct NM is done automatically by recognizing which NM is running on which channel.

### 3.8.1 API Mapping

For OSEK NM channels the AUTOSAR NM Interface API is mapped to the corresponding OSEK NM API. Return values are evaluated and transformed to 'NM Interface'-compatible values.

The table below shows the mapping of the function and the corresponding return values (not listed NM Interface functions are not mapped):

NM Interface	OSEK NM	Return value
Nm_Init()	NmOsekInit( NM_SLEEPIND )	NM_E_OK
Nm_PassiveStartUp()	NmOsekInit( NM_SLEEPIND )	NM_E_OK
Nm_NetworkRequest()	GotoMode( Awake )	NM_E_OK
Nm_NetworkRelease()	GotoMode( BusSleep )	NM_E_OK
Nm_GetState()	NmGetState() <sup>4</sup>	NM_E_OK
Nm_CheckRemoteSleepIndication()	NmGetRemoteSleepInd()	NM_E_OK
Nm_DisableCommunication()	SilentNM()	NM_E_OK
Nm_EnableCommunication()	TalkNM()	NM_E_OK

Table 3-3 API Mapping for OSEK NM Channels

### 3.8.2 Callback Mapping

Control callbacks from the OSEK NM are implemented by NM Interface. Some of them are mapped to the corresponding callbacks within the NM Interface. This mapping is shown in the following table (rows that contain '---' could not be directly mapped):

OSEK NM	NM Interface
ApplNmCanNormal ApplNmWaitBusSleepCancel	Nm_NetworkMode
ApplNmWaitBusSleep	Nm_PrepareBusSleepMode
ApplNmCanSleep	Nm_BusSleepMode
ApplNmBusStart	---
ApplNmCanBusSleep	---
ApplNmBusOff	---
ApplNmBusOffEnd	---

Table 3-4 Callback Mapping for OSEK NM Channels



#### Caution

OSEK NM also offers data callbacks that are not implemented within the NM Interface.

<sup>4</sup> OSEK NM state information is converted to AUTOSAR NM state information.

**Caution**

OSEK NM has a different behavior when it receives a NM message in sleep state. While AUTOSAR NM notifies `Nm_NetworkStartIndication()` OSEK NM performs a start-up and will call `ApplNmCanNormal()`.

### 3.8.3 Limitations

The following NM features cannot be used when at least one channel with OSEK NM is used:

- > Macro Layer Optimization (refer also to chapter 3.3)
- > Passive Mode Enabled

The following NM features can additionally not be used when only channels with OSEK NM are used:

- > Node Detection Enabled
- > Node Id Enabled
- > Pdu Rx Indication
- > User Data Enabled
- > State Report Enabled
- > Car Wake Up Rx Enabled

### 3.8.4 NM Coordination

It is only possible to synchronize the NM Coordination with one OSEK NM channel due to the shutdown timing of an OSEK NM channel depends on the availability of the NM token and normally this token will not be available for all channels at one node at the same time. For the coordination it is necessary to know to which OSEK NM channel the other network management shall be synchronized. This is set in the configuration (refer to chapter 6.3.1 'Component Configuration').

### 3.8.5 State Change Notifications

This optional feature enables state change notification calls to the higher layer when OSEK NM changes its state. The NM Interface calls the function name that is configured with the following additional arguments:

`nmChannelHandle:` System Channel Handle

`nmPreviousState` and `nmCurrentState`: see Table 3-4

nmCurrentState					
	NM_STATE_BUS_SLEEP	NM_STATE_PREPARE_BUS_SLEEP	NM_STATE_READY_SLEEP	NM_STATE_NORMAL_OPERATION	NM_STATE_BUS_OFF
nmPreviousState					
NM_STATE_BUS_SLEEP			■	■	
NM_STATE_PREPARE_BUS_SLEEP	■		■	■	
NM_STATE_READY_SLEEP		■		■	■
NM_STATE_NORMAL_OPERATION			■		■
NM_STATE_BUS_OFF			■	■	

Table 3-5 OSEK NM State Change Notification Arguments: nmPreviousState and nmCurrentState

Note that BusOff may also occur on the CAN Bus in NM\_STATE\_PREPARE\_BUS\_SLEEP but this is explicitly not modeled into the state 'NM\_STATE\_BUS\_OFF'.

From the NM perspective, the TX path of the channel can be enabled if the current state is NM\_STATE\_NORMAL\_OPERATION or NM\_STATE\_READY\_SLEEP.

**Caution**

CANbedded CCL is incompatible with this feature.

For further information about the configuration refer to chapter 6.3.1 'Component Configuration'.

### 3.8.6 Extended Initialization

OSEK NM is initialized by the NM Interface. This optional feature allows configuring the state which OSEK NM shall enter during initialization. Note that the initialization state is only valid for channels where only OSEK NM is active.

The following initialization states are possible:

Initialization State	Description
NM_CANSLEEP	OSEK NM initializes in 'Bus Sleep', the communication shall be disabled. This is the default value.
NM_NORMAL	OSEK NM is started during initialization and a communication request is set.

Initialization State	Description
NM_SLEEPIND	OSEK NM is started during initialization but no communication request is set.

Table 3-6 OSEK NM Extended Initialization States

For more information about the OSEK NM initialization states refer also to the Technical Reference of the OSEK NM. Further information about the configuration can be found in chapter 6.3.1 'Component Configuration'.

**Caution**

This feature can only be used when an initialization mode other than 'Sleep' is allowed by the stack.

### 3.9 ComM 2.1 Support

To allow the integration of an AUTOSAR 2.1 Communication Manager in combination with AUTOSAR 3 Network Management this feature has been added. It adapts the API Interface to work with ComM 2.1.

Please note that the optional NM callbacks have to be configured depending on the needs of the ComM 2.1. For the configuration of this feature and the callbacks refer to chapter 6.3.1 'Component Configuration'.

**Caution**

This feature does not allow the integration with further AUTOSAR 2.1 components affecting the NM. Especially the CAN Interface and the EcuM (for the initialization mechanism) are expected to be implemented according to AUTOSAR 3.

### 3.10 Gateway Extension

The gateway extension functionality provides two optional features, the Diagnostic Gateway Extension and the Network Management Gateway Extension.

The Diagnostic Gateway Extension feature supports the application by reading and writing one special byte in the NM user data.

The NM Gateway Extension feature provides an extended network coordination handling where networks can be shut down if no node in any network requests it.

**Info**

The gateway extension can be used only for systems with a maximum of 8 NM channels.

### 3.10.1 Diagnostic Gateway Extension

The Diagnostic Gateway Extension informs the application about each received NM message with the following additional callback:

```
void ComM_Nm_DiagGwReqIdIndication (
    const NetworkHandleType nmChannelHandle,
    uint8 nmNodeId, uint8 nmReqId )
```

 (5.9.3.2)

Beside the channel the parameters this callback contains the 'node identifier' and the 'requested node identifier'. The 'requested node identifier' is located in byte 4 of the NM message.

The application can set the 'requested node identifier' by calling the following function:

```
Nm_ReturnType Nm_SetDiagGwReqId ( uint8 nmReqId )
```

 (5.6.9.7)

This function causes additionally the transmission of the own NM message on all channels. The transmission request is executed within the next CAN NM main function.

For the configuration of the Diagnostic Gateway Extension refer to chapter 6.3.1 'Component Configuration'.

**Info**

Note that diagnostic gateway functionality is only provided for channels where CAN NM is used.

### 3.10.2 NM Gateway Extension

The Network Management Gateway Extension provides an extended coordination handling:

Any network is kept awake if a communication request for that network is active or a node in another network needs the network to be active.

On a wakeup event further channels can be requested by the NM Gateway Extension algorithm.

For the configuration of the NM Gateway Extension refer to chapter 6.3.1 'Component Configuration'.

#### 3.10.2.1 Shutdown Criteria

A network can shut down if all of the following criteria are fulfilled:

1. Application has not requested communication for the network



2. No node in the network requests communication (detected via remote sleep mechanism)
3. No node in another network requests the network by setting corresponding information in byte 5 of the NM message
4. No node in another network is active and needs information, i.e. receives signals, from some nodes in the network (information is provided by the generation tool by evaluating routing relations of signals)

If any of the criteria is not fulfilled the network is kept awake.

Criterion 4 can be additionally influenced by a Remote Sleep Filter mask for each network. Other networks that are marked as inactive within this mask cannot be kept awake by the routing relations of active nodes in this network.

This mask is initially configured to keep all networks with routing relations awake.

The application can change the channel-specific Remote Sleep Filter mask by calling the following function:

```
void Nm_SetGwRemoteSleepFilter (
    const NetworkHandleType nmChannelHandle,
    uint8 nmFilterMask )
```

 (5.6.9.4)

### 3.10.2.2 Wakeup Handling

On any network activity on a sleeping network starts other networks when they are defined as active in a Remote Wakeup Filter mask for that network.

Default value of the remote wakeup filter is that all channels are started.

The application can change the remote wakeup filter by calling the following function:

```
void Nm_SetGwRemoteWakeupFilter (
    const NetworkHandleType nmChannelHandle,
    uint8 nmFilterMask )
```

 (5.6.9.5)

The NM Interface cannot detect an external wakeup event. Therefore the application has to inform the NM of any external wakeup event with the function:

```
void Nm_WakeupNotification (
    const NetworkHandleType nmChannelHandle )
```

 (5.6.9.6)

Note that the gateway algorithm ignores this notification for channels that are already requested by the NM Interface.

### 3.10.2.3 Coordination of Multiple NMs within One Channel

The Gateway Extension supports an AUTOSAR CAN NM and an OSEK NM running both on the same network. Both NMs are synchronized for shutdown as far as possible.

Externally this channel behaves like an AUTOSAR CAN NM channel, i.e. any API access or callback notification is related to the CAN NM. The OSEK NM is handled only internally.

**Info**

The gateway extension handling of multiple NMs within one channel does not influence the shutdown criteria for other networks. The coordination feature (refer to chapter 3.5 'Coordinator Extension') cannot be used and must be disabled.

### 3.11 Car Wakeup

If 'Car Wakeup' feature is enabled in the configuration the NM Interface provides a configurable callback to the application. This callback informs the application when a 'Car Wakeup' request has been detected by CAN NM or FlexRay NM. Refer also to the Technical References of CAN NM and FlexRay NM.

For the configuration of the feature refer to chapter 6.3.1 'Component Configuration'.

### 3.12 Set Nm State in User Data

The feature 'Set Nm State in User Data' writes state change notifications as bit-coded values in a configured Com signal. The following table provides all state changes (from previous state to current state) and the corresponding signal values that are set by the NM Interface.

Previous State	Current State	Signal Value
Bus Sleep Mode <sup>5</sup>	Repeat Message	1
Prepare Bus Sleep Mode <sup>6</sup>	Repeat Message	2
Repeat Message	Normal Operation	4
Ready Sleep	Normal Operation	8
Ready Sleep	Repeat Message	16
Normal Operation	Repeat Message	32

Table 3-7 Nm State Change Signal Values

This feature is optional and has to be enabled in the configuration. For the configuration of the feature refer to chapter 6.3.2 'Channel Configuration'.

Note that this feature should only be used for AUTOSAR NMs (e.g. CAN NM and FlexRay NM).

### 3.13 Multiple ECU Support

The NM Interface 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 as NM Interface functionality does not have any parameter relevant for a multiple ECU project.

<sup>5</sup> As FlexRay NM does not perform a transition directly from Bus Sleep Mode to Repeat Message State the value is set in the transition from Synchronize Mode to Repeat Message State.

<sup>6</sup> This transition is not available for FlexRay NM.

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

**Note**

As only one configuration is available it is possible to pass the null pointer to the initialization function. NM Interface will automatically initialize with this available configuration.

### 3.14 Multiple Configuration Support

The NM Interface supports Multiple Configuration setups when the configuration in GENy is done with an AUTOSAR ECU configuration file.

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

```
void Nm_Init (
    const Nm_ConfigType * nmConfigPtr ) (5.5.1)
```

The configuration type (Nm\_ConfigType) passed to the initialization function contains all relevant parameters for the multiple configuration support.

**Note**

The NM Interface provides defines for the different initialization configurations like other modules do. Please take a look at the Nm\_Cfg.h header file. Choose the relevant configuration define and use this for configuring the EcuM. Refer to [13] for further information.

**Note**

Multiple Configuration Support is only relevant when the feature 'Coordinator Support Enabled' is used. Otherwise it is currently not relevant which configuration is passed to the initialization function.

**Caution**

Although this feature uses the post-build initialization mechanism the NM Interface is still not post-build able.

### 3.15 Fiat Class B NM and Fiat Class C NM Support

The NM Interface supports channels with a Fiat Class B NM or Fiat Class C NM if the configuration in GENy is done with an AUTOSAR ECU configuration file.



---

**Note**

This feature can only be used with the MICROSAR Fiat Class B NM and/or the MICROSAR Fiat Class C NM which supports an AUTOSAR like API (similar to CAN NM) and the AUTOSAR architecture. Refer to [11] and/or [12] for details of the MICROSAR Fiat Class B NM / MICROSAR Fiat Class C NM.

---

#### 3.15.1 API Mapping

API calls for channels with this NM type are forwarded to the Fiat Class B NM / Fiat Class C NM similar as for CAN NM and FlexRay NM. Therefore the NM Interface expects that the Fiat Class B NM / Fiat Class C NM implements the corresponding service functions. A complete list, which functions are expected can be found in chapter 5.7 'Service Functions Used by NM Interface'.

#### 3.15.2 Callback Mapping

The Fiat Class B NM / Fiat Class C NM component must also provide at least the following mandatory callbacks to the NM Interface:

- > Nm\_NetworkStartIndication (refer also to chapter 5.8.1)
- > Nm\_NetworkMode (see also chapter 5.8.2)
- > Nm\_PrepareBusSleepMode (refer also to chapter 5.8.3)
- > Nm\_BusSleepMode (see also chapter 5.8.4)

All other callback functions are optional and must be only provided if the corresponding feature is fully supported by the Fiat Class B NM / Fiat Class C NM (refer to [11] and/or [12] for details).

#### 3.15.3 Limitations

The following features cannot be used when at least one channel with Fiat Class B NM / Fiat Class C NM is used:

- > Gateway Extension (see chapter 3.10)
- > Passive Mode Enabled

### 3.16 Error Handling

#### 3.16.1 Development Error Detection

Development errors are reported to the DET using the service `Det_ReportError()` as specified in [7], if development error reporting is enabled (i.e. pre-compile parameter `NM_DEV_ERROR_DETECT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported Nm ID is 29.

The reported service IDs identify the services which are described in chapter 5.5, chapter 5.6, and chapter 5.8. The following table presents the service IDs and the related services:

Service ID	Service
0x00	Nm_Init
0x01	Nm_PassiveStartUp
0x02	Nm_NetworkRequest
0x03	Nm_NetworkRelease
0x04	Nm_DisableCommunication
0x05	Nm_EnableCommunication
0x06	Nm_SetUserData
0x07	Nm_GetUserData
0x08	Nm_GetPduData
0x09	Nm_RepeatMessageRequest
0x0A	Nm_GetNodeIdentifier
0x0B	Nm_GetLocalNodeIdentifier
0x0D	Nm_CheckRemoteSleepIndication
0x0E	Nm_GetState
0x0F	Nm_GetVersionInfo
0x10	Nm_MainFunction
0x11	Nm_NetworkStartIndication
0x12	Nm_NetworkMode
0x13	Nm_PrepareBusSleepMode
0x14	Nm_BusSleepMode
0x15	Nm_PduRxIndication
0x16	Nm_StateChangeNotification
0x1C	Nm_ActiveCoordIndication
0x1D	Nm_CarWakeUpIndication
0x24	Nm_RemoteSleepIndication
0x25	Nm_RemoteSleepCancellation
0x28	Nm_RepeatMessageIndication

Service ID	Service
0x29	Nm_TxTimeoutException
0x30	Nm_LimpHomeIndication
0x31	Nm_LimpHomeCancelation
0x33	Nm_GwPduRxIndication
0xC0	Nm_RequestBusSynchronization
0xD0	Nm_CheckLimpHomeIndication
0xD1	Nm_SetGwRemoteSleepFilter
0xD2	Nm_SetGwRemoteWakeupFilter
0xD3	Nm_WakeupNotification
0xD4	Nm_SetDiagGwReqId

Table 3-8 Service IDs

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

Error Code	Description
0x01 NM_E_NO_INIT	API service used without module initialization.
0x02 NM_E_INVALID_CHANNEL	API service used with wrong channel handle.
0x13 NM_E_NULL_PTR	Null Pointer has been passed as argument.
0x20 NM_E_PRIORITY_COLLISION	Two Coordinators with the same priority are connected to the same bus
0x21 NM_E_SLEEPBIT_ERROR	Sleep Indication Bit received on an active channel

Table 3-9 DET Error Codes

### 3.16.2 Production Code Error Reporting

By default, production code related errors are reported to the DEM using the service `Dem_ReportErrorStatus()` as specified in [8], if production error reporting is enabled (i.e. pre-compile parameter `NM_PROD_ERROR_DETECT==STD_ON`).

If another module is used for production code error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Dem_ReportErrorStatus()`.

Currently the NM Interface does not report any production error.

## 4 Integration

### 4.1 Files

The NM Interface consists of the following files:

#### Files of NM Interface









Nm.c	Source code of NM Interface. The user <b>must not</b> change this file!	
Nm.h	API of NM Interface. The user <b>must not</b> change this file!	
Nm_Cbk.h	API of NM Interface callback functions. The user <b>must not</b> change this file!	
Nm_Cfg.c	Pre-compile variant Configuration source file. The user <b>must not</b> change this file!	
Nm_Cfg.h	Configuration header file for NM Interface. The user <b>must not</b> change this file!	
Nm_Lcfg.c	Link-time variant Configuration source file. The user <b>must not</b> change this file!	
Nm_Lcfg.h	Link-time variant Configuration header file. The user <b>must not</b> change this file!	
NmStack_Types.h	Provided header file for global NM Types. The user <b>must not</b> change this file!	

Table 4-1 File Names of the NM Interface



#### Note

If the NM is used with OSEK NM and a non-AUTOSAR CAN Driver the configuration files `Nm_Cfg.h` and `Nm_Cfg.c` will be generated as `NmIf_Cfg.h` and `NmIf_Cfg.c` to avoid a conflict with the generated OSEK NM configuration header file.

## 4.2 Include Structure

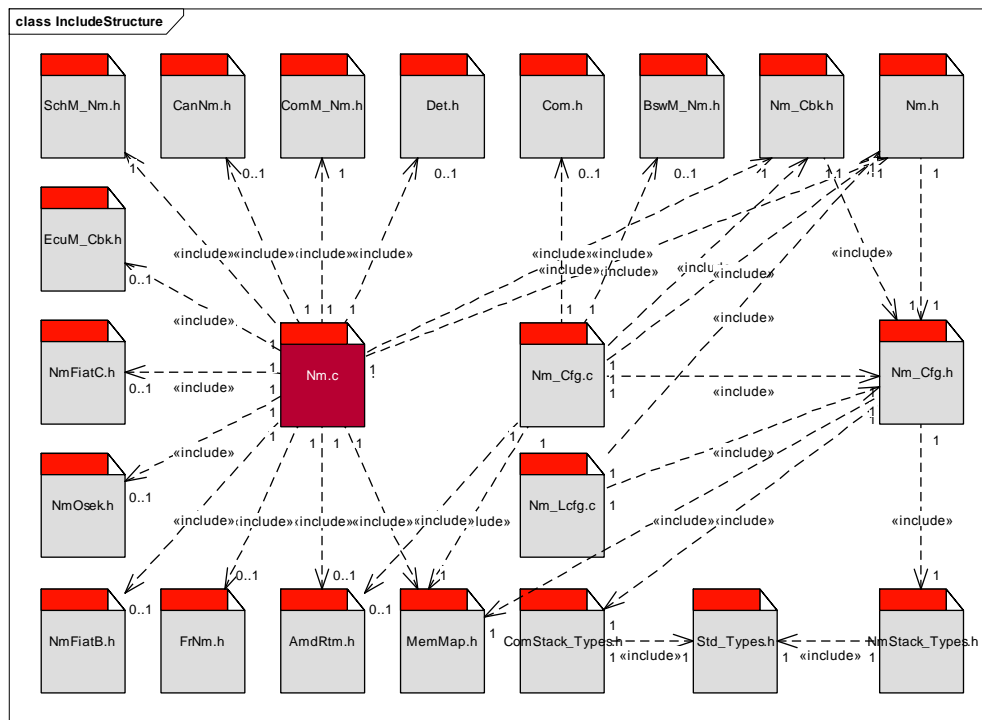


Figure 4-1 Include Structure of Nm Interface



### Note

If the feature 'OSEK NM Support' is enabled, also the configuration header file of the OSEK NM is included by the NM Interface.

If on at least one channel Fiat Class B and/or Fiat Class C is used, then also the configuration header file of the Fiat Class B NM / Fiat Class C NM is included by the NM Interface.

## 4.3 Version Changes

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

## 4.4 Initialization

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

Also refer to chapter 5.5.1 'Nm\_Init: Initialization of NM Interface'.



**Caution**

The NM Interface assumes that some variables are initialized with zero at start-up when the feature 'Coordinator Support Enabled' is enabled. If the embedded target does not initialize RAM within the start-up code the function 'Nm\_InitMemory' has to be called during start-up and before the initialization is performed. Refer also to chapter 3.1.2.7 'Memory Initialization'.

**Note**

In an AUTOSAR environment where the ECU Manager is used, the initialization is performed within the ECU Manager. Refer to the Technical Reference of the ECU Manager for further details.

When not using the feature 'Multiple Configuration Support' only one configuration exists that has to be passed to the initialization function. Alternatively the null pointer can be passed and the NM Interface will automatically initialize with this configuration. When using the feature 'Multiple Configuration Support' multiple configurations exist and depending on which identity shall be active the correct one has to be chosen. Refer to chapter 3.14 'Multiple Configuration Support' for further information about this feature.

## 4.5 Main Function

The NM Interface provides a main function when the NM Coordinator is used or the Gateway Extension is enabled. This main function has to be called cyclic on task level. Default Timing value is 10 milliseconds. The call cycle time value has to be set in the component configuration (refer to chapter 6.3.1 'Component Configuration').

For Further Information refer also to chapter 5.5.2 'Nm\_MainFunction: Main Function of the NM Interface'.

**Caution**

If no BSW Scheduler is used for the critical sections (refer to chapter 4.6 'Critical Sections') the NM main function must not be interrupted by any other task.

**Note**

If the BSW Scheduler is used, the main function calls are executed by the BSW Scheduler. When the BSW Scheduler is available in GENy the main function timing is automatically adapted when changing the value in the component configuration. Refer to [7] for more details about the BSW Scheduler.

## 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.

The NM Interface calls the following function when entering a critical section:

```
void SchM_Enter_Nm ( uint8 ExclusiveArea ) ( )
```

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

```
void SchM_Exit_Nm ( uint8 ExclusiveArea ) ( )
```

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 4.6.1 'Critical Section Codes'.

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

**Note**

Note that critical sections are only relevant if the NM coordinator is used.

### 4.6.1 Critical Section Codes

Since the BSW Scheduler is used for critical sections (see also chapter 4.6 'Critical Sections') the Nm provides two critical section codes which must lead to corresponding interrupt locks, described in the following table:

Critical Section Define	Interrupt Lock
NM_EXCLUSIVE_AREA_0	No interruption by any interrupt is allowed. Therefore this section must always lock global interrupts.
NM_EXCLUSIVE_AREA_1	This section must lock task interrupts so that no interruption by the task of any bus-specific NM is possible. If the system architecture ensures this already otherwise (e.g. in non-preemptive systems) this section must not lead to any interrupt lock.

Table 4-2 Critical Section Codes



**Note**

Note that critical section codes are only relevant if the NM coordinator is used.

## 5 API Description

### 5.1 API Categories

The AUTOSAR Network Management supports a multi-channel indexed API.

When only one bus-specific NM Type is used the API can be optimized in order to save runtime (refer to chapter 3.3 'Macro Layer Optimization').

### 5.2 Data Types

The software module NM Interface 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`.

NM Interface provides a header file for global NM Types, `NmStack_Types.h`. Following types are specified in this file:

Name	Type	Description
Nm_ReturnType	uint8	Return type for NM functions.
Nm_StateType	enum	State of the network management state machine.
Nm_ModeType	enum	Mode of the network management state machine.

Table 5-1 Global NM Types

Furthermore the following software module specific data types are used:

Name	Type	Description
Nm_ConfigType <sup>7</sup>	struct	Structure for configuration parameters.
Nm_ChannelConfigType <sup>7</sup>	struct	Structure for the channel specific configuration parameters.
Nm_BusNmType	enum	Enumeration that provides information about the bus-specific NM type of the channel.
Nm_SyncNmType <sup>8</sup>	enum	Enumeration that provides information if the channel is selective or synchronous

Table 5-2 NM Interface Data Types

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

### 5.3 Global Variables

There are no global variables within NM Interface.

<sup>7</sup> These types are used for configuration purposes only.

<sup>8</sup> Only available if the NM Coordinator is enabled.

## 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 constants:

Name	Type	Description
NM_AR_MAJOR_VERSION	BCD	Contains the major specification version number.
NM_AR_MINOR_VERSION	BCD	Contains the minor specification version number.
NM_AR_PATCH_VERSION	BCD	Contains the patch level specification version number.

Table 5-3 Specification Version API Data

### 5.4.2 Component Versions

The source code version of NM Interface is provided by three BCD coded constants:

Name	Type	Description
NM_SW_MAJOR_VERSION	BCD	Contains the major component version number.
NM_SW_MINOR_VERSION	BCD	Contains the minor component version number.
NM_SW_PATCH_VERSION	BCD	Contains the patch level component version number.

Table 5-4 Component Version API Data

### 5.4.3 Vendor and Module ID

NM Interface provides the vendor and module identifier according to HIS:

Name	Type	Description
NM_VENDOR_ID	-	Vendor ID according to HIS.
NM_MODULE_ID	-	Module ID according to HIS.

Table 5-5 Vendor and Module ID

## 5.5 Administrative Functions Provided by NM Interface

### 5.5.1 Nm\_Init: Initialization of NM Interface

Nm\_Init

#### Prototype

```
void Nm_Init ( const Nm_ConfigType * nmConfigPtr )
```

#### Parameter

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

#### Return code

-	-
---	---

#### Service ID

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

#### Functional Description

Initializes the NM Interface.  
This function calls additionally the initialization function of OSEK NM, if configured.

#### Particularities and Limitations

- The function `Nm_Init()` has to be called before any other NM functionality is used.
- Call context: task level
- Not re-entrant
- This service is called by EcuM.
- The GENy code generator provides macro definitions for the `nmConfigPtr` parameter. Please take a look at `Nm_Cfg.h` after code generation. The name of these macro definitions may vary if you have a multiple ECU or a multiple configuration setup. Please refer to [13] for details about the Initialization of Basic Software Modules. This applies only to AUTOSAR ECU configurations.
- Note that this function prototype deviates from the AUTOSAR document [2]. The original AUTOSAR requirement requires actually this prototype:  

```
void Nm_Init(Nm_ConfigType* const nmConfigPtr)
```

## 5.5.2 Nm\_MainFunction: Main Function of the NM Interface

### Nm\_MainFunction

Prototype	
void <b>Nm_MainFunction</b> ( void )	
Parameter	
nmConfigPtr	Configuration Pointer
Return code	
-	-
Service ID	
Service ID	0x10
Functional Description	
Main function of the NM Interface. This function controls the NM Coordination.	
Particularities and Limitations	
<ul style="list-style-type: none"><li>■ This function has to be called cyclically on task level.</li><li>■ Call context: task level.</li><li>■ Not re-entrant.</li><li>■ If this service is called by the Basic Scheduler or the CCL it must not be called by the application.</li><li>■ This function is only available if <code>NM_COORDINATOR_SUPPORT_ENABLED</code> is <code>STD_ON</code>.</li></ul>	

## 5.6 Service Functions Provided by NM Interface

### 5.6.1 Nm\_GetState: Get the State of the Network Management

Nm\_GetState

Prototype	
<pre>Nm_ReturnType Nm_GetState ( const NetworkHandleType nmChannelHandle,                              Nm_StateType* const      nmStatePtr,                              Nm_ModeType* const      nmModePtr )</pre>	
Parameter	
nmChannelHandle	Identification of the physical channel
nmStatePtr	Pointer where state of the network management shall be copied to
nmModePtr	Pointer where network management mode 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	0x0E



## Functional Description

Returns the current state and the current mode of the network management by calling the GetState function for the respective bus-specific NM.

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)<sup>9</sup>
- > NM\_STATE\_READY\_SLEEP (Ready Sleep)
- > NM\_STATE\_NORMAL\_OPERATION (Normal Operation)
- > NM\_STATE\_REPEAT\_MESSAGE (Repeat Message)<sup>10</sup>
- > NM\_STATE\_SYNCHRONIZE (Synchronized State)<sup>11</sup>
- > NM\_STATE\_WAIT\_CHECK\_ACTIVATION (Wait Check Activation)<sup>12</sup>
- > NM\_STATE\_WAIT\_NETWORK\_STARTUP (Wait Network Startup)<sup>12</sup>
- > NM\_STATE\_BUS\_OFF (BusOff)<sup>13</sup>

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 the documentation for the corresponding BusNm (e.g. [3], [4], [11], [12]).

As a state value for channels with OSEK NM only,

- > NM\_STATE\_BUS\_SLEEP is returned, if it is in NMBusSleep (NM\_ACTION\_BUS\_SLEEP),
- > NM\_STATE\_PREPARE\_BUS\_SLEEP is returned, if it is in NMTwbsNormal or NMTwbsLimpHome (NM\_ACTION\_GO\_BUSSLEEP),
- > NM\_STATE\_READY\_SLEEP is returned, if it is not in NMBusSleep, NMTwbsNormal or NMTwbsLimpHome and GotoMode(BusSleep) has been called and it has not got any BusOff indication by CanSM,
- > NM\_STATE\_NORMAL\_OPERATION is returned, if it is not in NMBusSleep, NMTwbsNormal or NMTwbsLimpHome and GotoMode(Awake) has been called and it has not got any BusOff indication by CanSM,.
- > NM\_STATE\_BUS\_OFF is returned, if it is not in NMBusSleep, NMTwbsNormal or NMTwbsLimpHome and it has got a BusOff indication by CanSM.

## Particularities and Limitations

- Call context: task and interrupt level
- Re-entrant

<sup>9</sup> Not used by FrNm

<sup>10</sup> Only used by CanNm and FrNm

<sup>11</sup> Only used by FrNm

<sup>12</sup> Only used by NmFiatB and NmFiatC

<sup>13</sup> Only used by OSEK NM

## 5.6.2 Nm\_GetVersionInfo: Version Information API

### Nm\_GetVersionInfo

Prototype	
void <b>Nm_GetVersionInfo</b> ( Std_VersionInfoType* NmVerInfoPtr )	
Parameter	
NmVerInfoPtr	Pointer where the Version Information shall be copied to
Return code	
-	-
Service ID	
Service ID	0x0F
Functional Description	
This service returns the version information of this module. The version information includes the Module Id, the Vendor Id and the vendor specific version numbers (BSW00407).	
Particularities and Limitations	
<ul style="list-style-type: none"><li>■ Call context: task level</li><li>■ Re-entrant</li><li>■ This function is enabled if <code>NM_VERSION_INFO_API</code> is <code>STD_ON</code></li></ul>	

### 5.6.3 Nm\_PassiveStartUp: Wake-up Network Management

Nm\_PassiveStartUp

#### Prototype

```
Nm_ReturnType Nm_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 network management. This function calls the PassiveStartUp function for the respective bus-specific NM.

#### Particularities and Limitations

- Call context: task and interrupt level
- Re-entrant
- The wake-up handling is done by ComM.

## 5.6.4 Wake-up Registration

### 5.6.4.1 Nm\_NetworkRequest: Request the Network

#### Nm\_NetworkRequest

##### Prototype

```
Nm_ReturnType Nm_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

Request the network if the ECU needs to communicate on the bus by calling the NetworkRequest function for the respective bus-specific NM.

##### Particularities and Limitations

- Call context: task and interrupt level
- Re-entrant
- This service is called by ComM.
- Not available for passive nodes (NM\_PASSIVE\_MODE\_ENABLED is STD\_ON)

### 5.6.4.2 Nm\_NetworkRelease: Release the Network

#### Nm\_NetworkRelease

##### Prototype

```
Nm_ReturnType Nm_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	Network Release 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 by calling the NetworkRelease function for the respective bus-specific NM.

##### Particularities and Limitations

- Call context: task and interrupt level
- Re-entrant
- This service is called by ComM.
- Not available for passive nodes (NM\_PASSIVE\_MODE\_ENABLED is STD\_ON)

## 5.6.5 Communication Control Service

### 5.6.5.1 Nm\_DisableCommunication: Disable NM Message Transmission

#### Nm\_DisableCommunication

##### Prototype

```
Nm_ReturnType Nm_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	0x04
------------	------

##### Functional Description

Disable transmission of NM messages by calling the corresponding bus-specific function. Note that FlexRay NM does not support this feature and therefore this function returns for FlexRay NM channels NM\_E\_NOT\_EXECUTED.

##### Particularities and Limitations

- Call context: task and interrupt level
- Only re-entrant with different channel handles
- This function is enabled if NM\_COM\_CONTROL\_ENABLED is STD\_ON

### 5.6.5.2 Nm\_EnableCommunication: Enable NM Message Transmission

#### Nm\_EnableCommunication

##### Prototype

```
Nm_ReturnType Nm_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	0x05
------------	------

##### Functional Description

Enable transmission of NM messages by calling the corresponding bus-specific function. Note that for FlexRay NM this feature is not available and therefore this function just returns NM\_E\_NOT\_OK.

##### Particularities and Limitations

- Call context: task and interrupt level
- Only re-entrant with different channel handles
- This function is enabled if `NM_COM_CONTROL_ENABLED` is `STD_ON`

## 5.6.6 User Data Handling

### 5.6.6.1 Nm\_SetUserData: Set User Data

#### Nm\_SetUserData

##### Prototype

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

##### Parameter

nmChannelHandle	Identification of the physical channel
nmUserDataPtr	Pointer to the user data for the next transmitted NM message

##### Return code

NM_E_OK	No error
NM_E_NOT_OK	Setting of user data has failed

##### Service ID

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

##### Functional Description

Set user data for NM messages transmitted next on the bus by calling the SetUserData function for the respective bus-specific NM.

##### Particularities and Limitations

- Call context: task and interrupt level
- Only re-entrant with different channel handles
- This function is enabled if NM\_USER\_DATA\_ENABLED is STD\_ON
- Not available for passive nodes (NM\_PASSIVE\_MODE\_ENABLED is STD\_ON) or if NM\_COM\_USER\_DATA\_ENABLED is STD\_ON



### 5.6.6.2 Nm\_GetUserData: Get User Data

#### Nm\_GetUserData

##### Prototype

```
Nm_ReturnType Nm_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	0x07
------------	------

##### Functional Description

Get user data of the last NM message received previously on the bus by calling the GetUserData function for the respective bus-specific NM.

##### Particularities and Limitations

- Call context: task and interrupt level
- Re-entrant
- This function is enabled if `NM_USER_DATA_ENABLED` is `STD_ON`

### 5.6.6.3 Nm\_GetPduData: Get NM Pdu Data

#### Nm\_GetPduData

##### Prototype

```
Nm_ReturnType Nm_GetPduData (
    const NetworkHandleType nmChannelHandle,
    uint8* const nmPduData )
```

##### Parameter

nmChannelHandle	Identification of the physical channel
nmPduData	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	0x08
------------	------

##### Functional Description

Get the whole PDU data out of the last received NM message by calling the GetPduData function for the respective bus-specific NM.

##### Particularities and Limitations

- Call context: task and interrupt level
- Re-entrant
- This function is enabled if NM\_NODE\_ID\_ENABLED is STD\_ON or NM\_USER\_DATA\_ENABLED is STD\_ON

## 5.6.7 Node Detection

### 5.6.7.1 Nm\_RepeatMessageRequest: Set Repeat Message Request Bit

Nm\_RepeatMessageRequest

Prototype	
Nm_ReturnType <b>Nm_RepeatMessageRequest</b> ( 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	0x09
Functional Description	
Set Repeat Message Request Bit for NM messages transmitted next on the bus by calling the RepeatMessageRequest function for the respective bus-specific NM.	
Particularities and Limitations	
<ul style="list-style-type: none"> <li>■ Call context: task and interrupt level</li> <li>■ Only re-entrant with different channel handles</li> <li>■ This function is enabled if NM_NODE_DETECTION_ENABLED is STD_ON and if NM_PASSIVE_MODE_ENABLED is STD_OFF.</li> </ul>	

### 5.6.7.2 Nm\_GetNodeIdentifier: Get Source Node Identifier

#### Nm\_GetNodeIdentifier

##### Prototype

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

##### Parameter

nmChannelHandle	Identification of the physical channel
nmNodeIdPtr	Pointer where node identifier from 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 received NM message has failed

##### Service ID

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

##### Functional Description

Get the node identifier from the last received NM message by calling the GetNodeIdentifier function for the respective bus-specific NM.

##### Particularities and Limitations

- Call context: task and interrupt level
- Re-entrant
- This function is enabled if NM\_NODE\_ID\_ENABLED is STD\_ON

### 5.6.7.3 Nm\_GetLocalNodeIdentifier: Get Local Source Node Identifier

#### Nm\_GetLocalNodeIdentifier

##### Prototype

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

##### Parameter

nmChannelHandle	Identification of the physical channel
nmNodeIdPtr	Pointer where node identifier of the local node is copied

##### Return code

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

##### Service ID

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

##### Functional Description

Get the node identifier configured for the local node by calling the GetLocalNodeIdentifier function for the respective bus-specific NM.

##### Particularities and Limitations

- Call context: task and interrupt level
- Re-entrant
- This function is enabled if NM\_NODE\_ID\_ENABLED is STD\_ON

## 5.6.8 Remote Sleep Indication

### 5.6.8.1 Nm\_CheckRemoteSleepIndication: Check for Remote Sleep Indication

#### Nm\_CheckRemoteSleepIndication

##### Prototype

```
Nm_ReturnType Nm_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	0x0D
------------	------

##### Functional Description

Check if remote sleep indication takes place or not by calling the CheckRemoteSleepIndication function for the respective bus-specific NM.

##### Particularities and Limitations

- Call context: task and interrupt level
- Re-entrant
- This function is enabled if NM\_REMOTE\_SLEEP\_INDICATION\_ENABLED is STD\_ON and if NM\_PASSIVE\_MODE\_ENABLED is STD\_OFF.

## 5.6.9 Vector Extensions

### 5.6.9.1 Nm\_InitMemory: Memory Initialization

Nm\_InitMemory

#### Prototype

```
void Nm_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.

#### Particularities and Limitations

- > The function Nm\_InitMemory() has to be called with disabled global interrupts.
- > Call context: task level.
- > Not re-entrant.
- > This function is only available if NM\_COORDINATOR\_SUPPORT\_ENABLED is STD\_ON.

### 5.6.9.2 Nm\_RequestBusSynchronization: Request Bus Synchronization<sup>14</sup>

#### Nm\_RequestBusSynchronization

##### Prototype

```
Nm_ReturnType Nm_RequestBusSynchronization (
    const NetworkHandleType nmChannelHandle )
```

##### Parameter

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

##### Return code

NM_E_OK	No error
NM_E_NOT_OK	Checking of remote sleep indication has failed
NM_E_NOT_EXECUTED	Requesting of node detection is currently not executed

##### Service ID

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

##### Functional Description

Request Bus Synchronization by calling the RequestBusSynchronization function for the respective bus-specific NM.

##### Particularities and Limitations

- Call context: task level only
- Re-entrant
- This function is enabled if `NM_ENABLE_COMM21_SUPPORT` is defined and `NM_BUS_SYNCHRONIZATION_ENABLED` is `STD_ON` and if `NM_PASSIVE_MODE_ENABLED` is `STD_OFF`.

<sup>14</sup> If `NM_ENABLE_COMM21_SUPPORT` is not defined this is only an internal API.



### 5.6.9.3 Nm\_CheckLimpHomeIndication: Check the Limp Home Status

#### Nm\_CheckLimpHomeIndication

##### Prototype

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

##### Parameter

nmChannelHandle	Identification of the physical channel
nmLimpHomeIndPtr	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 limp home status has failed

##### Service ID

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

##### Functional Description

Check the limp home status of the respective bus-specific NM. Note that only CAN NM and OSEK NM are supported. For channels where CAN NM and OSEK NM are both assigned true (limp home) is returned when both NMs are in limp home, otherwise false is returned.

##### Particularities and Limitations

- Call context: task level only
- Re-entrant
- This function is enabled if `NM_LIMP_HOME_INDICATION` is defined

#### 5.6.9.4 Nm\_SetGwRemoteSleepFilter: Set Remote Sleep Filter

Nm\_SetGwRemoteSleepFilter

##### Prototype

```
Nm_ReturnType Nm_SetGwRemoteSleepFilter (
    const NetworkHandleType nmChannelHandle,
    uint8 nmFilterMask )
```

##### Parameter

nmChannelHandle	Identification of the physical channel
nmFilterMask	Filter mask that shall be set for the remote sleep filter. The bit position corresponds to the channel number. If the bit on a position is set the corresponding channel is kept awake by routing relations.

##### Return code

NM_E_OK	No error
NM_E_NOT_OK	Setting of Remote Sleep Filter has failed

##### Service ID

Service ID	0xD1
------------	------

##### Functional Description

API sets the remote sleep filter for a channel. The filter influences the shutdown behavior of the NM gateway algorithm on that channel.

##### Particularities and Limitations

- Call context: task level only
- Only re-entrant for different channel handles
- This function is enabled if 'NM Gateway Extension' is enabled

### 5.6.9.5 Nm\_SetGwRemoteWakeupFilter: Set Remote Wakeup Filter

#### Nm\_SetGwRemoteWakeupFilter

##### Prototype

```
Nm_ReturnType Nm_SetGwRemoteWakeupFilter (
    const NetworkHandleType nmChannelHandle,
    uint8 nmFilterMask )
```

##### Parameter

nmChannelHandle	Identification of the physical channel
nmFilterMask	Filter mask that shall be set for the remote wakeup filter. The bit position corresponds to the channel number. If the bit on a position is set the corresponding channel is (re-)started during wakeup.

##### Return code

NM_E_OK	No error
NM_E_NOT_OK	Setting of Remote Wakeup Filter has failed

##### Service ID

Service ID	0xD2
------------	------

##### Functional Description

API sets the remote wakeup filter for a channel. The filter influences the wakeup behavior of the NM gateway algorithm on that channel.

##### Particularities and Limitations

- Call context: task level only
- Only re-entrant for different channel handles
- This function is enabled if 'NM Gateway Extension' is enabled

### 5.6.9.6 Nm\_WakeupNotification: Wakeup Notification

#### Nm\_WakeupNotification

##### Prototype

```
Nm_ReturnType Nm_WakeupNotification (
    const NetworkHandleType nmChannelHandle )
```

##### Parameter

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

##### Return code

NM_E_OK	No error
NM_E_NOT_OK	Notification for an external wakeup has failed
NM_E_NOT_EXECUTED	Notification for an external wakeup will be ignored due to not allowed in current NM state.

##### Service ID

Service ID	0xD3
------------	------

##### Functional Description

API that informs the NM that an external wakeup has been detected by the application.

##### Particularities and Limitations

- Call context: task or interrupt level
- Only re-entrant for different channel handles
- This function is enabled if 'NM Gateway Extension' is enabled

### 5.6.9.7 Nm\_SetDiagGwReqId: Set Requested Diagnostic Node Identifier

Nm\_SetDiagGwReqId

#### Prototype

```
Nm_ReturnType Nm_SetDiagGwReqId ( uint8 nmReqId )
```

#### Parameter

nmReqId	Requested Diagnostic Node Identifier
---------	--------------------------------------

#### Return code

NM_E_OK	No error
NM_E_NOT_OK	Setting requested diagnostic node identifier has failed

#### Service ID

Service ID	0xD4
------------	------

#### Functional Description

API sets the fourth byte in the NM transmission message on all channels and requests an additional transmission in the next CAN NM main function.

#### Particularities and Limitations

- Call context: task or interrupt level
- Not re-entrant
- This function is enabled if 'Diagnostic Gateway Extension' is enabled

## 5.7 Service Functions Used by NM Interface

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

Component	API
DET	Det_ReportError
SchM	SchM_Enter_Nm SchM_Exit_Nm
CanNm	CanNm_PassiveStartUp CanNm_NetworkRequest CanNm_NetworkRelease CanNm_GetState CanNm_SetUserData CanNm_GetUserData CanNm_GetPduData CanNm_RepeatMessageRequest CanNm_GetNodeIdentifier CanNm_GetLocalNodeIdentifier CanNm_CheckRemoteSleepIndication CanNm_DisableCommunication CanNm_EnableCommunication CanNm_RequestBusSynchronization
FrNm	FrNm_PassiveStartUp FrNm_NetworkRequest FrNm_NetworkRelease FrNm_GetState FrNm_SetUserData FrNm_GetUserData FrNm_GetPduData FrNm_RepeatMessageRequest FrNm_GetNodeIdentifier FrNm_GetLocalNodeIdentifier FrNm_CheckRemoteSleepIndication FrNm_DisableCommunication FrNm_EnableCommunication FrNm_RequestBusSynchronization

Component	API
NmOsek	NmOsekInit GotoMode GetConfig NmGetStatus NmGetRemoteSleepInd TalkNM SilentNM NmStateLimphone NmStateBusSleep NmStateWaitBusSleep NmStateBusSleepInd
NmFiatB	NmFiatB_PassiveStartUp NmFiatB_NetworkRequest NmFiatB_NetworkRelease NmFiatB_GetState NmFiatB_SetUserData NmFiatB_GetUserData NmFiatB_GetNodeIdentifier NmFiatB_GetLocalNodeIdentifier NmFiatB_RepeatMessageRequest NmFiatB_GetPduData NmFiatB_CheckRemoteSleepIndication NmFiatB_EnableCommunication NmFiatB_DisableCommunication
NmFiatC	NmFiatC_PassiveStartUp NmFiatC_NetworkRequest NmFiatC_NetworkRelease NmFiatC_GetState NmFiatC_SetUserData NmFiatC_GetUserData NmFiatC_GetNodeIdentifier NmFiatC_GetLocalNodeIdentifier NmFiatC_RepeatMessageRequest NmFiatC_GetPduData NmFiatC_CheckRemoteSleepIndication NmFiatC_EnableCommunication NmFiatC_DisableCommunication

Component	API
ComM (CCL)	ComM_Nm_NetworkStartIndication ComM_Nm_NetworkMode ComM_Nm_PrepareBusSleepMode ComM_Nm_BusSleepMode ComM_Nm_RestartIndication ComM_Nm_LimpHomeIndication ComM_Nm_LimpHomeCancelation ComM_Nm_DiagGwReqIdIndication

Table 5-6 Services Used by the NM Interface



## 5.8 Callback Functions Provided by NM Interface

The following callback functions are called from CAN NM, FlexRay Nm, Fiat Class B or Fiat Class C.

### 5.8.1 Nm\_NetworkStartIndication: Network Start Indication

#### Nm\_NetworkStartIndication

##### Prototype

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

##### Parameter

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

##### Return code

-	-
---	---

##### Service ID

Service ID	0x20
------------	------

##### Functional Description

Notification that a NM-message has been received in the Bus-Sleep Mode indicating that some nodes in the network have already entered the Network Mode.

##### Particularities and Limitations

- Call context: task level
- Re-entrant
- This service may only be called by CAN NM, FlexRay NM, Fiat Class B or Fiat Class C

### 5.8.2 Nm\_NetworkMode: Network Mode Indication

Nm\_NetworkMode

Prototype	
<pre>void Nm_NetworkMode (     const NetworkHandleType nmChannelHandle )</pre>	
Parameter	
nmChannelHandle	Identification of the physical channel
Return code	
-	-
Service ID	
Service ID	0x21
Functional Description	
Notification that the network management has entered Network Mode.	
Particularities and Limitations	
<ul style="list-style-type: none"><li>■ Call context: task level</li><li>■ Re-entrant</li><li>■ This service may only be called by CAN NM, FlexRay NM, Fiat Class B or Fiat Class C</li></ul>	

### 5.8.3 Nm\_PrepareBusSleepMode: Prepare Bus Sleep Mode Indication

Nm\_PrepareBusSleepMode

Prototype	
<pre>void Nm_PrepareBusSleepMode (     const NetworkHandleType nmChannelHandle )</pre>	
Parameter	
nmChannelHandle	Identification of the physical channel
Return code	
-	-
Service ID	
Service ID	0x22
Functional Description	
Notification that the network management has entered Prepare Bus Sleep Mode.	
Particularities and Limitations	
<ul style="list-style-type: none"><li>■ Call context: task level</li><li>■ Re-entrant</li><li>■ This service may only be called by CAN NM, Fiat Class B or Fiat Class C</li></ul>	

### 5.8.4 Nm\_BusSleepMode: Bus Sleep Mode Indication

Nm\_BusSleepMode

Prototype	
<pre>void Nm_BusSleepMode (     const NetworkHandleType nmChannelHandle )</pre>	
Parameter	
nmChannelHandle	Identification of the physical channel
Return code	
-	-
Service ID	
Service ID	0x23
Functional Description	
Notification that the network management has entered Bus Sleep Mode.	
Particularities and Limitations	
<ul style="list-style-type: none"><li>■ Call context: task level</li><li>■ Re-entrant</li><li>■ This service may only be called by CAN NM, FlexRay NM, Fiat Class B or Fiat Class C</li></ul>	

### 5.8.5 Nm\_RemoteSleepIndication: Remote Sleep Indication

Nm\_RemoteSleepIndication

Prototype	
<pre>void Nm_RemoteSleepIndication (     const NetworkHandleType nmChannelHandle )</pre>	
Parameter	
nmChannelHandle	Identification of the physical channel
Return code	
-	-
Service ID	
Service ID	0x24
Functional Description	
Notification that the network management has detected that all other nodes are ready to sleep.	
Particularities and Limitations	
<ul style="list-style-type: none"><li>■ Call context: task level</li><li>■ Re-entrant</li><li>■ This service may only be called by CAN NM, FlexRay NM, Fiat Class B or Fiat Class C</li><li>■ This function is enabled if NM_REMOTE_SLEEP_IND_ENABLED is STD_ON</li></ul>	

### 5.8.6 Nm\_RemoteSleepCancellation: NM Remote Sleep Cancellation

Nm\_RemoteSleepCancellation

Prototype	
<pre>void Nm_RemoteSleepCancellation (     const NetworkHandleType nmChannelHandle )</pre>	
Parameter	
nmChannelHandle	Identification of the physical channel
Return code	
-	-
Service ID	
Service ID	0x25
Functional Description	
Notification that the network management has detected that no more all other nodes are ready to sleep.	
Particularities and Limitations	
<ul style="list-style-type: none"> <li>■ Call context: task level</li> <li>■ Re-entrant</li> <li>■ This service may only be called by CAN NM, FlexRay NM, Fiat Class B or Fiat Class C</li> <li>■ This function is enabled if NM_REMOTE_SLEEP_IND_ENABLED is STD_ON</li> </ul>	

### 5.8.7 Nm\_PduRxIndication: NM Message Reception Indication

Nm\_PduRxIndication

Prototype	
<pre>void Nm_PduRxIndication (     const NetworkHandleType nmChannelHandle )</pre>	
Parameter	
nmChannelHandle	Identification of the physical channel
Return code	
-	-
Service ID	
Service ID	0x26
Functional Description	
Notification that a NM message has been received.	
Particularities and Limitations	
<ul style="list-style-type: none"> <li>■ Call context: task level</li> <li>■ Re-entrant</li> <li>■ This service may only be called by CAN NM or FlexRay NM</li> <li>■ This function is enabled if NM_PDU_RX_INDICATION_ENABLED is STD_ON</li> </ul>	

### 5.8.8 Nm\_RepeatMessageIndication: Repeat Message Request Indication

Nm\_RepeatMessageIndication

#### Prototype

```
void Nm_RepeatMessageIndication (
    const NetworkHandleType nmChannelHandle )
```

#### Parameter

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

#### Return code

-	-
---	---

#### Service ID

Service ID	0x28
------------	------

#### Functional Description

Notification that a NM message has been received where the Repeat Message Bit was set.

#### Particularities and Limitations

- Call context: task level
- Re-entrant
- This service may only be called by CAN NM or FlexRay NM
- This function is enabled if `NM_NODE_DETECTION_ENABLED` is `STD_ON`

### 5.8.9 Nm\_StateChangeNotification: State Change Notification

#### Nm\_StateChangeNotification

##### Prototype

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

##### Parameter

nmChannelHandle	Identification of the physical channel
nmPreviousState	Previous State
nmCurrentState	Current State

##### Return code

-	-
---	---

##### Service ID

Service ID	0x27
------------	------

##### Functional Description

Notification that the NM state has changed.

##### Particularities and Limitations

- Call context: task level
- Re-entrant
- This service may only be called by CAN NM, FlexRay NM, Fiat Class B or Fiat Class C
- This function is enabled if `NM_STATE_CHANGE_IND_ENABLED` is `STD_ON`

### 5.8.10 Nm\_TxTimeoutException: Transmission Timeout Exception

Nm\_TxTimeoutException

Prototype	
<pre>void Nm_TxTimeoutException (     const NetworkHandleType nmChannelHandle )</pre>	
Parameter	
nmChannelHandle	Identification of the physical channel
Return code	
-	-
Service ID	
Service ID	0x29
Functional Description	
Notification that the transmission of network management was not successful for a configurable amount of time.	
Particularities and Limitations	
<ul style="list-style-type: none"> <li>■ Call context: task level</li> <li>■ Re-entrant</li> <li>■ This service may only be called by CAN NM or FlexRay NM</li> <li>■ This function is disabled if NM_PASSIVE_MODE_ENABLED is STD_ON</li> </ul>	

### 5.8.11 Nm\_CarWakeUpIndication: Car Wake-up Indication

Nm\_CarWakeUpIndication

Prototype	
<pre>void Nm_CarWakeUpIndication (     const NetworkHandleType nmChannelHandle )</pre>	
Parameter	
nmChannelHandle	Identification of the physical channel
Return code	
-	-
Service ID	
Service ID	0x2A
Functional Description	
Notification that a message with the car wake-up bit set has been received.	
Particularities and Limitations	
<ul style="list-style-type: none"> <li>■ Call context: task level</li> <li>■ Re-entrant</li> <li>■ This service may only be called by CAN NM or FlexRay NM</li> <li>■ This function is only enabled if NM_CAR_WAKE_UP_RX_ENABLED is STD_ON</li> </ul>	

## 5.8.12 Vector Extensions

### 5.8.12.1 Nm\_ActiveCoordIndication: Indication of an Active Coordinator Bit

#### Nm\_ActiveCoordIndication

##### Prototype

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

##### Parameter

nmChannelHandle	Identification of the physical channel
nmCoordPrio	Coordinator priority received
nmSleepInd	Sleep Indication Bit

##### Return code

-	-
---	---

##### Service ID

Service ID	0x32
------------	------

##### Functional Description

This callback function is called when a NM message with a priority from another coordinator is received. The handling of the information is done in the NM Interface.

##### Particularities and Limitations

- Call context: Interrupt and task level
- Only re-entrant for different channel handles
- This service may only be called by CAN NM or FlexRay NM
- This function is enabled if Coordinator Extension feature is available and `NM_ENABLE_COORD_SYNC_SUPPORT` is defined



### 5.8.12.2 Nm\_LimpHomeIndication: Limp Home Indication

#### Nm\_LimpHomeIndication

##### Prototype

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

##### Parameter

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

##### Return code

-	-
---	---

##### Service ID

Service ID	0x30
------------	------

##### Functional Description

Notification that the network management has detected that no NM message was received and no message could be transmitted for a certain amount of time.

##### Particularities and Limitations

- Call context: task level
- Re-entrant
- This service may only be called by CAN NM
- This function is enabled if Limp Home Indication feature is available and `NM_LIMP_HOME_INDICATION` is defined

### 5.8.12.3 Nm\_LimpHomeCancelation: NM Remote Sleep Cancellation

#### Nm\_LimpHomeCancelation

##### Prototype

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

##### Parameter

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

##### Return code

-	-
---	---

##### Service ID

Service ID	0x31
------------	------

##### Functional Description

Notification that the network management has successfully received or transmitted a NM message after Nm\_LimpHomeIndication was called.

##### Particularities and Limitations

- Call context: task level
- Re-entrant
- This service may only be called by CAN NM
- This function is enabled if Limp Home Indication feature is available and NM\_LIMP\_HOME\_INDICATION is defined

### 5.8.12.4 Nm\_GwPduRxIndication: Extended Gateway Message Indication

Nm\_GwPduRxIndication

#### Prototype

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

#### Parameter

nmChannelHandle	Identification of the physical channel
nmNodeId	Node identifier in the received NM PDU
nmReqId	Requestor ID in the received NM PDU (byte 4)
nmReqCh	Requested channels to be active in the received NM PDU (byte 5)

#### Return code

-	-
---	---

#### Service ID

Service ID	0x33
------------	------

#### Functional Description

This callback function is called each time when a NM message is received. The additional information in that callback is required for the gateway extension algorithm.

#### Particularities and Limitations

- Call context: Interrupt or task level
- Only re-entrant for different channel handles
- This service may only be called by CAN NM
- This function is enabled if Gateway Extension feature is enabled

### 5.8.12.5 OSEK NM Callback Functions

The following table provides an overview which OSEK NM callbacks are implemented within the NM Interface. For more information about the callbacks refer to the Technical Reference of OSEK NM.

OSEK NM Callback Function	Description
ApplNmCanNormal	Notification that OSEK NM entered 'Network Mode' from 'Sleep'
ApplNmBusStart	Notification of OSEK NM Bus Start
ApplNmWaitBusSleep	Notification that OSEK NM entered 'Prepare Bus Sleep'
ApplNmWaitBusSleepCancel	Notification that OSEK NM restarted from 'Prepare Bus Sleep' and entered 'Network Mode'
ApplNmCanBusSleep	Notification that OSEK NM entered 'Bus Sleep'
ApplNmCanSleep	Notification that OSEK NM entered 'Bus Sleep'
ApplNmBusOff	Notification that OSEK NM was informed about BusOff occurrence
ApplNmBusOffEnd	Notification that OSEK NM was informed about a BusOff

OSEK NM Callback Function	Description
	recovery attempt
AppINmGwPduRxIndication	Extended gateway notification that a NM message with cleared sleep indication bit was received

Table 5-7 OSEK NM Callback Functions

## 5.9 Callback Functions Used by NM Interface

Refer also to chapter 5.8 'Callback Functions Provided by NM Interface' for more information about these callback functions.

### 5.9.1 Service Callback Functions of NM Interface

The following service callback functions of NM Interface are implemented within ComM. Please refer to documentation of the ComM component for a detailed description of these services:

```
void ComM_Nm_NetworkStartIndication( const NetworkHandleType
nmChannelHandle )

void ComM_Nm_NetworkMode( const NetworkHandleType nmChannelHandle
)

void ComM_Nm_PrepareBusSleepMode( const NetworkHandleType
nmChannelHandle )

void ComM_Nm_BusSleepMode( const NetworkHandleType
nmChannelHandle )

void ComM_Nm_RestartIndication( const NetworkHandleType
nmChannelHandle )15
```

These functions are called from task level.

### 5.9.2 Configurable Service Callback Functions

For the following service functions called optionally from the NM Interface an arbitrary callback function to the upper layer can be configured (see also chapter 6.3.1 'Component Configuration'):

```
void Ul_Nm_RepeatMessageIndication( const NetworkHandleType
nmChannelHandle )

void Ul_Nm_RemoteSleepIndication( const NetworkHandleType
nmChannelHandle )

void Ul_Nm_RemoteSleepCancellation( const NetworkHandleType
nmChannelHandle )

void Ul_Nm_PduRxIndication( const NetworkHandleType
nmChannelHandle )

void Ul_Nm_StateChangeNotification( const NetworkHandleType nmChannelHandle,
const Nm_StateType nmPreviousState,
const Nm_StateType nmCurrentState )

void Ul_TxTimeoutException( const NetworkHandleType
nmChannelHandle )
```

The naming of these functions can be configured arbitrarily.

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

---

<sup>15</sup> This callback function will only be called if the Coordination Feature is enabled.

### 5.9.3 Additional Service Callback Functions of NM Interface

The service callbacks described in this are additionally provided by the NM Interface when the corresponding feature is available. They have to be provided by the upper layer.

#### 5.9.3.1 Callbacks for Limp Home Indication

The following two callbacks are provided for limp home detection / cancellation if limp home feature is enabled:

```
void ComM_Nm_LimpHomeIndication( const NetworkHandleType  
nmChannelHandle )  
  
void ComM_Nm_LimpHomeCancelation( const NetworkHandleType  
nmChannelHandle )
```

These functions are called from task level.

#### 5.9.3.2 Callback for Diagnostic Gateway Extension

The following callback is provided by the NM Interface if the diagnostic gateway extension feature is enabled:

```
void ComM_Nm_DiagGwReqIdIndication( const NetworkHandleType  
nmChannelHandle,  
uint8 nmNodeId, uint8 nmReqId )
```

This function is called during NM message reception indication function, i.e. in most cases from interrupt level.

#### 5.9.3.3 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 NM Interface if the Generator Version Check or the CRC Check fails.

Refer to [10] for more information.

## 6 Configuration

The configuration of the AUTOSAR NM Interface component is done by pre-compile configuration or post-compile configuration. Pre-compile configuration can only be carried out if source code is available. Post-compile configuration is 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 post-compile configuration with object code and library has to be used.

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

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

### 6.1 Configuration Variants

The Nm supports the configuration variants

- > VARIANT-PRE-COMPILE
- > VARIANT-LINK-TIME

The configuration classes of the Nm parameters depend on the supported configuration variants. For their definitions please see the Nm\_bswmd.arxml file.

### 6.2 Configuration in Data Base

The following table provides a list of database attributes defined in a DBC file that apply to NM Interface.

Attribute	Object	Type	Values	Default	Description
NmType	Network	String	"NmAsr", "<OEM>- OSEK"	None	This attribute defines the type of the NM. For OSEK-NM this type is OEM-specific.
NmAsrNode <sup>16</sup>	Node	Enumeration	No, yes	No	This attribute defines whether the corresponding node uses the AUTOSAR NM ("Yes") or not ("No").

Table 6-1 Database Attributes

<sup>16</sup> Database attribute is required if GENy is used for the configuration.

### 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 NM Interface. Please note that the configuration items gathered from the database are disabled and grayed and cannot be overwritten by the user.

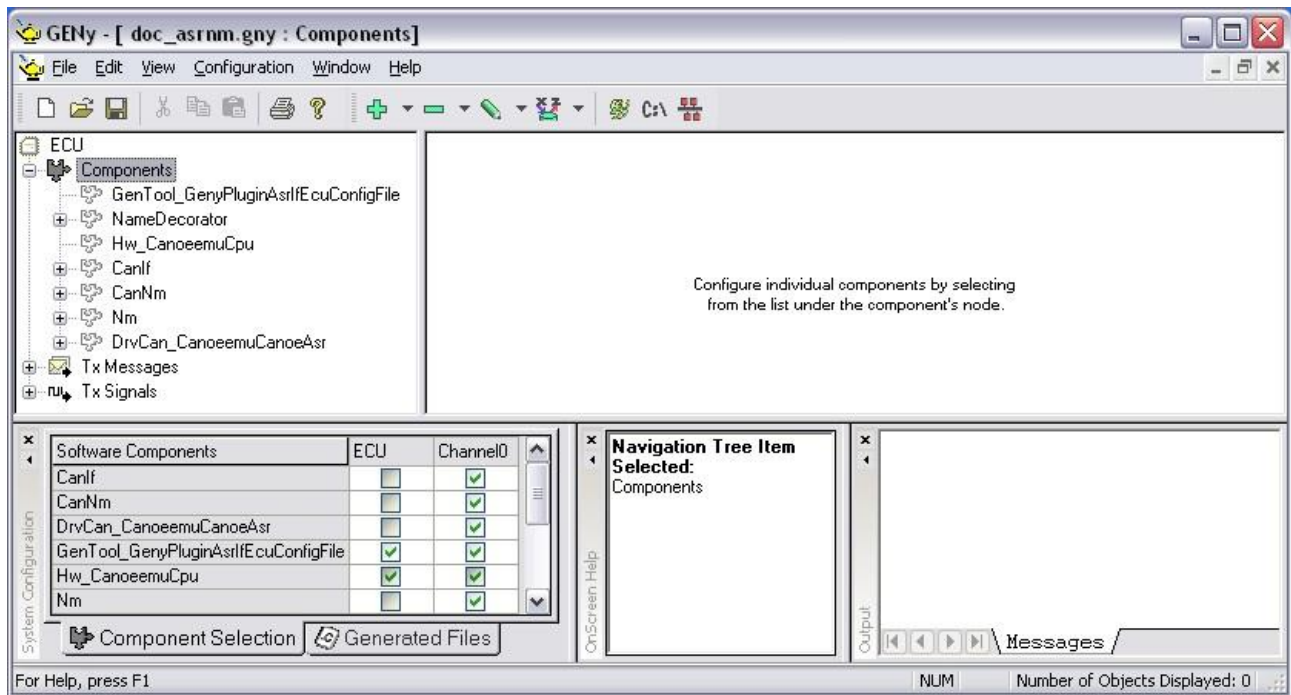


Figure 6-1 Component Selection in GENy

In order to configure the AUTOSAR Network Management Interface component, the component 'Nm' has to be activated on the designated channels.

NM Interface can be activated independently of the underlying bus specific network management. The activation of a bus-specific NM requires the activation of NM Interface on the same channel.

The configuration of the component is done on separate pages due to there are ECU specific and channel specific settings that have to be customized separately.

Depending on the used features and bus-specific NMs some restrictions apply to the configuration of the NM Interface and all used subjacent network managements. Additionally if OSEK Support is enabled (refer to chapter 3.8 'OSEK NM Support') some additional configuration items have to be considered to the OSEK NM configuration.

If a mixed-mode NM is used, all underlying bus-specific NMs have to be enabled on that channel.



### 6.3.1 Component Configuration

The component configuration of NM Interface is done as follows:

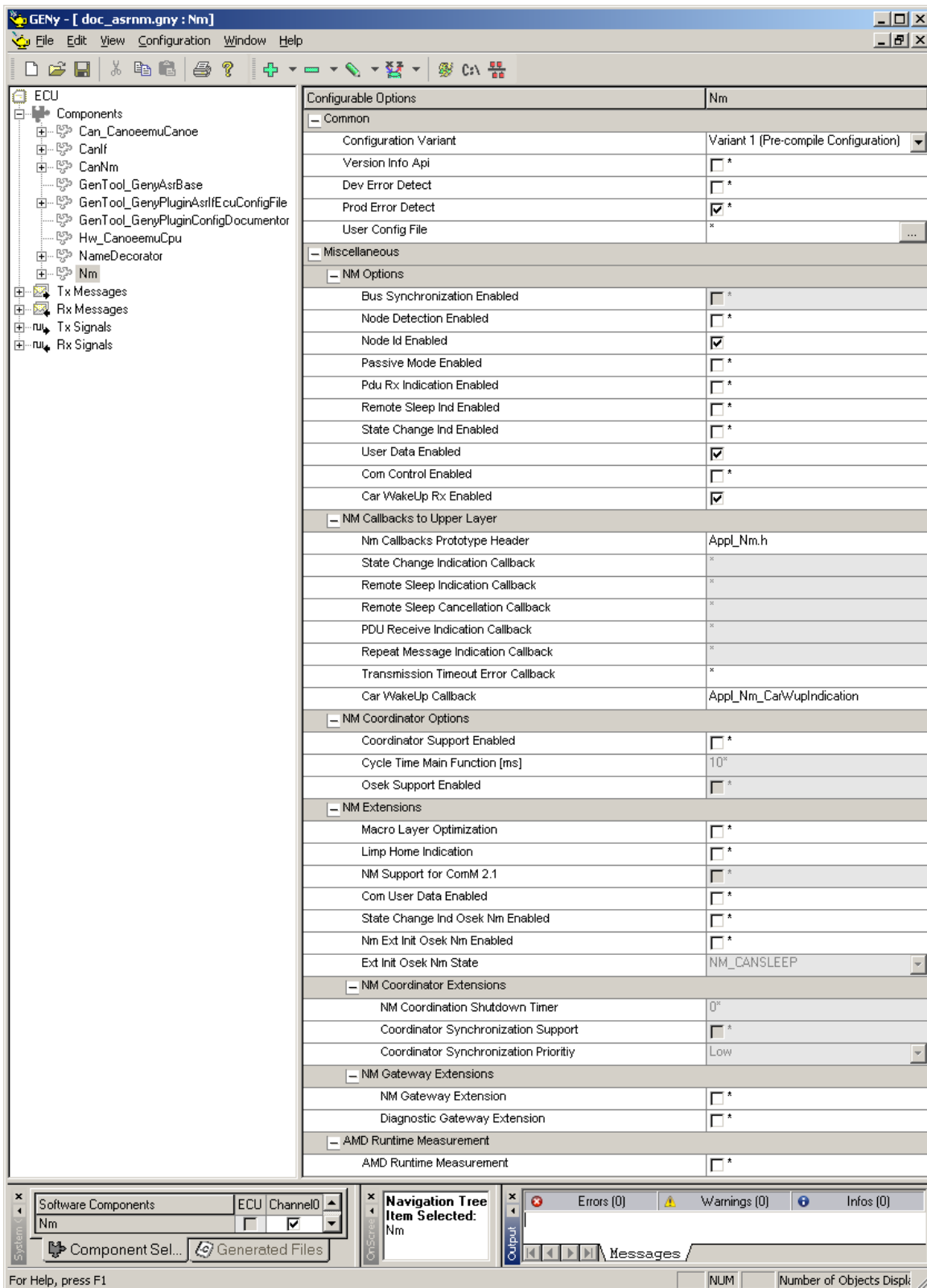


Figure 6-2 Component Configuration NM Interface

Configuration options	Value	Description
Common		
Configuration Variant	<ul style="list-style-type: none"> <li>&gt; Variant 1</li> <li>&gt; Variant 2</li> </ul>	Set the AUTOSAR Configuration Variant (Variant 1: Pre-compile configuration; Variant 2: Link-time configuration).
Version Info API <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	Activate/Deactivate the version information functionality.
Dev Error Detection <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	<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 <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	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 NM Interface configuration file during the generation process.</p> <p>It can be used to manually customize the NM Interface.</p>
Miscellaneous		
Network Management Interface Options		
Bus Synchronization Enabled <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	Activate/Deactivate bus synchronization (gateway functionality). It is automatically configured depending on the feature 'Coordinator Support'.
Node Detection Enabled <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	Activate/Deactivate node detection support. It can only be enabled if Passive Mode is disabled and Node Identifier is enabled.
Node Id Enabled <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	Activate/Deactivate node identifier support.
Passive Mode Enabled <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	Activate/Deactivate passive mode.
PDU Rx Indication Enabled <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	Activate/Deactivate PDU receive indication functionality.
Remote Sleep Ind Enabled <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	Activate/Deactivate remote sleep indication (gateway functionality). It can only be enabled if Passive Mode is disabled.
State Change Ind Enabled <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	Activate/Deactivate state change notification functionality.

<sup>17</sup> Pre-compile parameter which is read-only in post-compile configurations.

User Data Enabled <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	Activate/Deactivate the user data support.
Com Control Enabled <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	Activate/Deactivate communication control service. It can only be enabled if Passive Mode is disabled.
Car Wake Up Rx Enabled <sup>17</sup>	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	Activate/Deactivate the feature 'Car Wakeup'. Feature details can be found in chapter 3.11 'Car Wakeup'.
NM Callbacks to Upper Layer		
Nm Callbacks Prototype Header	string	Enter the header file name that contains the prototypes for all used callback functions (i.e. all callback functions below where a function name is entered). Return Type and API parameters for the function prototypes have to be the same as for the callback functions provided by the NM (see chapter 5.8). This field can be left blank if no callback to the upper layer is used.
State Change Indication Callback	string	Enter the function name of the service that should be called to the upper layer in case of a state transition. This field can be left blank if a callback to the upper layer is not required.
Remote Sleep Indication Callback	string	Enter the function name of the service that should be called to the upper layer in case of a remote sleep indication. This field can be left blank if a callback to the upper layer is not required.
Remote Sleep Cancellation Callback	string	Enter the function name of the service that should be called to the upper layer in case of a remote sleep cancellation. This field can be left blank if a callback to the upper layer is not required.
PDU Receive Indication Callback	string	Enter the function name of the service that should be called to the upper layer in case of a NM message reception. This field can be left blank if a callback to the upper layer is not required.
Repeat Message Indication Callback	string	Enter the function name of the service that should be called to the upper layer in case of a repeat message indication. This field can be left blank if a callback to the upper layer is not required.
Transmission Timeout Error Callback	string	Enter the function name of the service that should be called to the upper layer in case a transmission timeout occurs. This field can be left blank if a callback to the upper layer is not required.
Car Wake Up Callback	string	Enter the function name of the service that

		should be called to the upper layer in case a car wakeup is detected. This field can be left blank if a callback to the upper layer is not required.
<b>NM Coordinator Options</b>		
Coordinator Support Enabled <sup>17</sup>	> Enable > Disable	Activate/Deactivate Network Management Coordinator Support for synchronous shutdown.
NM Coordination Shutdown Timer	0..65535	The shutdown timer used for the computation of the shutdown timing for each channel. This parameter is only editable if Coordination Synchronization Support is enabled. Refer to chapter 6.3.3 'NM Coordination Restrictions' for more information.
Cycle Time Main Function [ms]	1..xxxx	The call cycle time value of the NM main function has to be set here. The default value is 10. Refer to chapter 6.3.3 'NM Coordination Restrictions' for more information.
OSEK Support Enabled <sup>17</sup>	Read Only	Activate/Deactivate Support for OSEK Network Management. The feature is handled automatically depending whether OSEK NM is enabled on at least one channel together with the NM.
Synchronized OSEK NM channel <sup>17</sup>	0..255	System channel handle to identify the OSEK NM channel that is relevant for the NM Coordination. This parameter is only visible and editable if more than one OSEK NM channel is available. In all other cases this value is automatically computed.
<b>NM Extensions</b>		
Macro Layer Optimization <sup>17</sup>	> Enable > Disable	In some configurations the NM Interface can be optimized to be a macro layer. Refer to chapter 3.3 'Macro Layer Optimization' for more details.
Limp Home Indication <sup>18</sup>	> Enable > Disable	Activate/Deactivate the limp home indication. It can only be enabled if Passive Mode is disabled.
NM Support for ComM 2.1	> Enable > Disable	Activate/Deactivate NM support for AUTOSAR 3 NM in combination with AUTOSAR 2.1 ComM.
NM User Data via Com	> Enable > Disable	Activate/Deactivate the feature 'NM User Data via Com'. It can only be enabled if Passive Mode is disabled.
State Change Ind Osek Nm Enabled	> Enable > Disable	Activate/Deactivate state change notifications when OSEK NM changes its state.
Nm Ext Init Osek Nm	> Enable	Activate/Deactivate the OSEK NM Initialization

<sup>18</sup> This configuration item is only relevant if the Limp Home Indication feature is available.

Enabled	> Disable	Extension feature.
Ext Init Osek Nm State	> NM_CANSLEEP > NM_NORMAL > NM_SLEEPIND	Set the OSEK NM Initialization state. Refer to chapter 3.8.6 'Extended Initialization' for further information.
<b>NM Coordinator Extensions</b>		
Coordinator Synchronization Support <sup>17</sup>	> Enable > Disable	Activate/Deactivate Coordinator Synchronization Support for supporting more than one coordinator connected to one channel.
Coordinator Synchronization Priority	> Low > Middle > High	Set the coordination synchronization priority used for the detection of the active coordinator.
<b>NM Gateway Extensions</b>		
NM Gateway Extension <sup>17</sup>	> Enable > Disable	Activate/Deactivate NM Gateway Extension. Refer to chapter 3.10.2 'NM Gateway Extension' for further information. Please consider also the configuration restrictions in chapter 6.3.7 'NM Gateway Extension Configuration Restrictions'.
Diagnostic Gateway Extension <sup>17</sup>	> Enable > Disable	Activate/Deactivate Diagnostic Gateway Extension. Refer to chapter 3.10.1 'Diagnostic Gateway Extension' for further information. Please consider also the configuration restrictions in chapter 6.3.6 'Diagnostic Gateway Extension Configuration Restrictions'.
<b>AMD Runtime Measurement</b>		
AMD Runtime Measurement <sup>17</sup>	> Enable > Disable	This parameter defines if AMD Runtime Measurement (RTM) is enabled for this module.

Table 6-2 Component Configuration NM Interface

### 6.3.2 Channel Configuration

The channel configuration of NM Interface is done as follows:

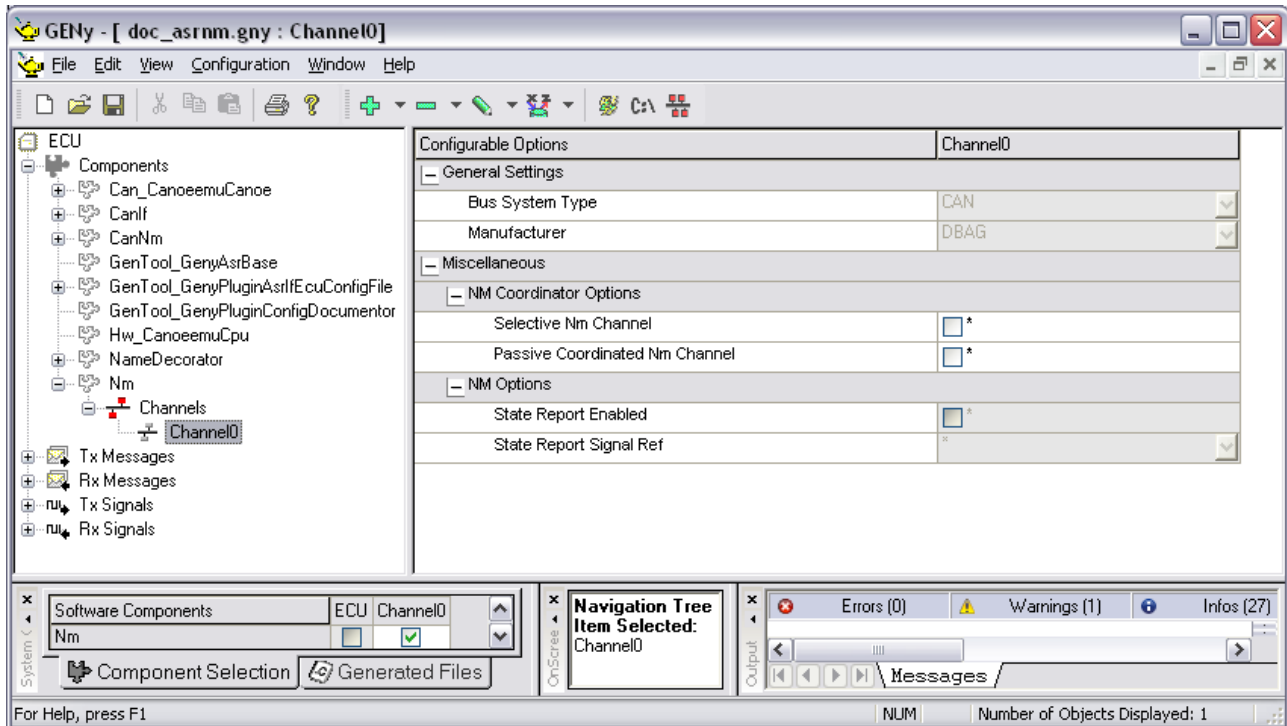


Figure 6-3 Channel Configuration NM Interface

Configuration options	Value	Description
<b>General Settings</b>		
Bus System Type	Read Only	Bus system type of the respective channel.
Manufacturer	Read Only	Value is currently not available for FlexRay channels. Therefore it will always be set to 'Unknown'.
<b>Miscellaneous</b>		
<b>NM Coordinator Options</b>		
Selective NM Channel	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	If enabled the corresponding channel is selective and therefore not part of the coordinated channels. If disabled the channel is synchronous and is considered in the NM Coordination. Refer to chapter 3.4.1 'Selective Channels' for further information.
Passive Coordinated Nm Channel	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	When enabling this feature the corresponding channel does not coordinate the channel actively and behaves always as if another coordinator with a higher priority is active. Refer to chapter 3.6 'Passive Coordinator' for more information.

NM Options		
State Report Enabled	<ul style="list-style-type: none"> <li>&gt; Enable</li> <li>&gt; Disable</li> </ul>	Activate/Deactivate the feature 'Set Nm State in User Data' on the corresponding channel. Refer to chapter 3.12 'Set Nm State in User Data' for further information.
State Report Signal Ref	Signal Name	This parameter defines the signal where the state change information is set. Note that this feature can only be enabled if 'State Report Enabled' is enabled. Refer to chapter 3.12 'Set Nm State in User Data' for further information.

Table 6-3 Channel Configuration NM Interface

### 6.3.3 NM Coordination Restrictions

#### 6.3.3.1 Needed NM Features

For the proper usage of the NM Coordination some configuration items of the NMs are required to be enabled. They are listed for all possibly involved components in the following table:

Component	Required Configuration Item
NM Interface / CAN NM / FlexRay NM	Remote Sleep Ind Enabled
NM Interface / CAN NM / FlexRay NM	Bus Synchronization Enabled
FlexRay NM / CAN NM	Ext Remote Sleep Indication
OSEK NM	Remote Sleep
OSEK NM	Node Monitoring
OSEK NM	Extended Callbacks
OSEK NM	Token Monitoring
OSEK NM	Indexed Component
Fiat Class B	Nm Type = 'Master'
Fiat Class C	Nm Type = 'Master'

Table 6-4 Required Configuration Items for the NM Coordination

For a detailed description of these descriptions and their configuration please refer to the Technical Reference of the corresponding network management.

#### 6.3.3.2 NM Main Function Cycle Time

The NM coordination algorithm is performed within the cyclic NM task. To ensure an immediate reaction to changes in the bus-specific NMs and the correct triggering of the shutdown phase the following timing restrictions must be fulfilled:

- > NM Main Function Cycle Time must be less or equal than the CAN NM Main Function Cycle Time
- > NM Main Function Cycle Time must be less or equal than the OSEK NM Main Function Cycle Time
- > NM Main Function Cycle Time must be less or equal than the FlexRay NM Repetition Cycle Time<sup>19</sup>

Additionally it is recommended that the bus-specific NM tasks are executed after the NM task.

#### 6.3.3.3 Shutdown Timing

For the computation of the shutdown timing of each channel a maximum value is required. This value can be configured with the parameter 'NM Coordination Shutdown Timer' (see chapter 6.3.1 'Component Configuration'). If this value is set to 0, GENy will automatically compute this value as maximum of the shutdown timing of all AUTOSAR NM channels. Any other value entered has to be greater or equal as this maximum value.

<sup>19</sup> FlexRay NM Repetition Cycle Time: (Repetition Cycle) \* (FlexRay Cycle Time)



If OSEK NM Support is enabled the 'NM Coordination Shutdown Timer' has also to be at least greater than the OSEK NM Wait Bus Sleep Time. If the shutdown timer value is set to 0, GENy will automatically generate a valid value.



#### Info

Adding a manual shutdown timing is most likely only necessary when the Extended Coordination feature is enabled, since in a scenario where multiple coordinators are connected to the same bus and having further independent channels each of the coordinator does not know all shutdown timings.

The calculation of a common minimum shutdown time is described in detail in the following sub-chapter.

#### 6.3.3.3.1 Shutdown Time Calculation

The common shutdown timing can be calculated as maximum shutdown timing of all coordinated channels from all coordinators. The shutdown timing of each channel depends on the NM type (OSEK NM, CAN NM, FlexRay NM, Fiat Class B or Fiat Class C). The computation formula for each type is listed in the following table.

NM Type	Shutdown Timing
CAN NM	$\text{CanNmWaitBusSleepTime}^{20} + \text{CanNmTimeoutTime}^{20} + \text{CanNmMainFunctionPeriod}^{20}$
FlexRay NM	$((\text{FrNmReadySleepCnt}^{21} + 2) * \text{FrNmRepetitionCycle}^{22} * \text{FrCycleTime}^{23})$
OSEK NM	$\text{OsekNmWaitBusSleepTime}^{20} + (\text{OsekNmRingTime}^{24} * \text{nrOfOsekNodes}^{25})$
Fiat Class B	$\text{NmFiatBSilentTime}^{26}$
Fiat Class C	$\text{NmFiatCWaitBusSleepTime}^{27}$

Table 6-5 Common Shutdown Time Calculation

#### 6.3.3.4 Synchronized Channels

For a reasonable coordination at least two NMs must be available that are coordinated. Therefore the minimum number of synchronous channels is one, if this synchronous channel is a channel with multiple NMs (refer to chapter 3.4.3 'Coordination of Multiple NMs within one Channel'). In all other cases at least two channels must be available that are not set to 'Selective'.

<sup>20</sup> Timing value given in ms

<sup>21</sup> Ready Sleep Counter is given in number of Repetition Cycles

<sup>22</sup> Repetition Cycle is given in number of FlexRay Cycles

<sup>23</sup> FlexRay Cycle Time (duration of one FlexRay cycle) given in ms

<sup>24</sup> Time interval between two ring messages (TTyp) given in ms

<sup>25</sup> (Maximum) Number of OSEK NM Nodes in the network

<sup>26</sup> Silent Time value given in s

<sup>27</sup> Wait Bus Sleep Timeout Time given in s

### 6.3.4 NM Extended Coordination Restrictions

In the following table the required enabled configuration items of all possibly involved components for a proper usage of the Coordinator Extension are listed:

Component	Required Configuration Item
NM Interface / CAN NM / FlexRay NM	Coordinator Synchronization Support <sup>28</sup>
CAN NM	Pdu Cbv Position
FlexRay NM	Control Bit Vector Enabled

Table 6-6 Required Items of Involved Components for Proper Usage of the Coordinator Extension

Additionally the following listed timing restrictions are required:

- > For each channel: (Ready Sleep Timeout Time)<sup>29</sup> > 2 \* (Message Cycle Time)
- > For all coordinators that are connected (directly or indirectly) the 'NM Coordination Shutdown Timer' has to be set explicitly (no automatic computation via setting the value to 0) to the same value. Refer to chapter 6.3.3.3 'Shutdown Timing' for details about the 'NM Coordination Shutdown Timer'.
- > If the CAN NM Feature 'Busload Reduction' is used all coordinators must have smaller 'Reduced Message Cycle Times' than all other nodes and a coordinator with a higher priority must have always a smaller 'Reduced Message Cycle Time' than another coordinator with a lower priority. For a detailed description of the 'Busload Reduction' Feature please refer to [3] and the Technical Reference of the CAN NM.

### 6.3.5 OSEK NM Configuration

For the usage of an OSEK NM channel the feature 'OSEK Support' has to be enabled (refer also to chapter 6.3.1 'Component Configuration'). For the configuration of this OSEK NM channel the configuration options as described in chapter 6.3.3 'NM Coordination Restrictions' have to be made.



#### Caution

For the correct usage of OSEK NM within the AUTOSAR stack the OSEK NM feature 'Extended Callback' must be enabled in all cases (not only for coordination scenario).

The following figure shows an OSEK NM configuration which is configured correctly for the OSEK Support within NM Interface and additionally the settings necessary for the NM Coordination are activated:

<sup>28</sup> When GENy is used for configuration this is automatically handled by GENy for CAN NM and FlexRay NM.

<sup>29</sup> CAN: NM Timeout Time; FlexRay: (Ready Sleep Counter + 1) \* (Repetition Cycle) \* (FlexRay Cycle Time)

Configurable Options	Nm_DirOsek
<b>General Settings</b>	
Indexed Component	<input checked="" type="checkbox"/>
User Config File	\$(ProjectDir)\Nm_DirOsek.cfg <input data-bbox="1305 342 1353 376" type="button" value="..."/>
Extended Callback	<input checked="" type="checkbox"/> *
Node Monitoring	<input checked="" type="checkbox"/>
Number of Nodes	64
<b>Extended Specification</b>	
Immediate Alive	<input type="checkbox"/>
Fast Bus Off Recovery	<input checked="" type="checkbox"/>
<b>NM Extensions</b>	
AUTOSAR Environment Usage	<input checked="" type="checkbox"/> *
Remote Sleep Indication	<input checked="" type="checkbox"/>
Nm Type	derived <input data-bbox="1329 763 1353 797" type="button" value="v"/>
BusOff Notification	<input checked="" type="checkbox"/>

Figure 6-4 OSEK NM Configuration

### 6.3.6 Diagnostic Gateway Extension Configuration Restrictions

For the proper usage of the diagnostic gateway extension some NM configuration items are required to be enabled. They are listed for all possibly involved components in the following table:

Component	Required Configuration Item
NM Interface / CAN NM	Node Id Enabled
CAN NM	Diagnostic Gateway Extension Support
OSEK NM	NM Message Rx Indication
OSEK NM	Indexed Component

Table 6-7 Required Configuration Items for Diagnostic Gateway Extension

Additionally the PDU length of the CAN NM message must be greater 5 and the node identifier position must not be configured off. For details of the CAN NM or OSEK NM configuration refer to the corresponding Technical Reference.



#### Caution

Diagnostic Gateway Extension cannot be used when FlexRay NM, Fiat Class B or Fiat Class C is used.

### 6.3.7 NM Gateway Extension Configuration Restrictions

For the proper usage of the NM gateway extension some NM configuration items are required to be enabled. The feature has the same restrictions as the diagnostic gateway extension (refer to chapter 6.3.6 'Diagnostic Gateway Extension Configuration Restrictions'). Additionally required configuration items listed in the following table:

Component	Required Configuration Item
NM Interface / CAN NM / OSEK NM	Remote Sleep Ind Enabled
NM Interface / CAN NM	Bus Synchronization Enabled
CAN NM	Extended Remote Sleep Indication
CAN NM	NM Gateway Extension Support
OSEK NM	Remote Sleep
OSEK NM	Node Monitoring
OSEK NM	Extended Callbacks
OSEK NM	Ring Data Access

Table 6-8 Required Configuration Items for NM Gateway Extension

Additionally the same restrictions to the NM main function cycle time as for the 'Coordinator Support' have to be taken into account. Refer to chapter 6.3.3.2 'NM Main Function Cycle Time' for details.

### 6.3.8 Fiat Class B NM / Fiat Class C NM Configuration Restrictions

For the usage of Fiat Class B NM / Fiat Class C NM some features must be set accordingly in the NM Interface. The following table contains those configuration items and the required setting.

Feature	Required Setting
Passive Mode Enabled	Disabled
State Change Ind Enabled	Enabled
NM Callbacks Prototype Header	BswM_Nm.h
State Change Indication Callback	BswM_Nm_StateChangeNotification

Table 6-9 Required Configuration Settings for Fiat Class B NM / Fiat Class C NM Usage



#### Note

These configuration restrictions should be already preconfigured when using a configuration with Fiat Class B NM / Fiat Class C NM.

Note that no other callback function except 'State Change Indication Callback', 'Remote Sleep Indication Callback'<sup>30</sup> and 'Remote Sleep Cancellation Callback'<sup>30</sup> are used by Fiat Class B NM / Fiat Class C NM.

<sup>30</sup> Only if 'Nm Type' is 'Master' in Fiat Class B / Fiat Class C

## 7 Glossary and Abbreviations

### 7.1 Abbreviations

Abbreviations	Complete expression
<b>API</b>	<b>A</b> pplication <b>P</b> rogramming <b>I</b> nterface
<b>AUTOSAR</b>	<b>A</b> utomotive <b>O</b> pen <b>S</b> ystem <b>A</b> rchitecture
<b>BSW</b>	<b>B</b> asic <b>S</b> oftware
<b>BSWM</b>	<b>B</b> asic <b>S</b> oftware <b>M</b> ode <b>M</b> anager
<b>CAN</b>	<b>C</b> ontroller <b>A</b> rea <b>N</b> etwork
<b>CBV</b>	<b>C</b> ontrol <b>B</b> it <b>V</b> ector
<b>CCL</b>	<b>C</b> ommunication <b>C</b> ontrol <b>L</b> ayer
<b>ComM</b>	<b>C</b> OM <b>M</b> anager
<b>DEM</b>	<b>D</b> iagnostic <b>E</b> vent <b>M</b> anager
<b>DET</b>	<b>D</b> evelopment <b>E</b> rror <b>T</b> racer
<b>DLC</b>	<b>D</b> ata <b>L</b> ength <b>C</b> ode (Number of data bytes of a CAN message)
<b>DLL</b>	<b>D</b> ata link layer
<b>EAD</b>	<b>E</b> mbedded <b>A</b> rchitecture <b>D</b> esigner (Generation tool for MICROSAR components)
<b>ECU</b>	<b>E</b> lectronic <b>C</b> ontrol <b>U</b> nit
<b>FIBEX</b>	<b>F</b> ield <b>B</b> us <b>E</b> xchange
<b>ID</b>	<b>I</b> dentifier (of a CAN message)
<b>IL</b>	<b>I</b> nteraction <b>L</b> ayer
<b>ISR</b>	<b>I</b> nterrupt <b>S</b> ervice <b>R</b> outine
<b>KL15</b>	<b>K</b> lemme <b>15</b>
<b>KL30</b>	<b>K</b> lemme <b>30</b>
<b>KL31</b>	<b>K</b> lemme <b>31</b>
<b>MICROSAR</b>	<b>M</b> icrocontroller <b>O</b> pen <b>S</b> ystem <b>A</b> rchitecture (the Vector AUTOSAR solution)
<b>MISRA</b>	<b>M</b> otor <b>I</b> ndustry <b>S</b> oftware <b>R</b> eliability <b>A</b> ssociation
<b>NID</b>	<b>N</b> ode <b>I</b> dentifier
<b>NM</b>	<b>N</b> etwork <b>M</b> anagement
<b>PDU</b>	<b>P</b> rotocol <b>D</b> ata <b>U</b> nit
<b>RAM</b>	<b>R</b> andom <b>A</b> ccess <b>M</b> emory
<b>RI</b>	<b>R</b> eference <b>I</b> mplementation (Reference Implementation of the CAN-Driver High Level part)
<b>ROM</b>	<b>R</b> ead <b>O</b> nly <b>M</b> emory
<b>SRS</b>	<b>S</b> oftware <b>R</b> equirement <b>S</b> pecification
<b>SWS</b>	<b>S</b> oftware <b>S</b> pecification

Table 7-1 Abbreviations

## 7.2 Glossary

Glossary	Description
<b>Confirmation</b>	Notification by the data link layer on asynchronous successful transmission of a CAN message.
<b>Identifier</b>	Identifies a message.
<b>Indication</b>	Notification by the data link layer on asynchronous reception of a message.
<b>Message</b>	One or more signals are assigned to each message.
<b>NM Channel</b>	Logical communication path associated with one NM cluster.
<b>NM Cluster</b>	Instance of the NM to handle one physical bus.
<b>Signal</b>	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 7-2 Glossary

## 8 Contact

Visit our website for more information on

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

**[www.vector.com](http://www.vector.com)**