Document Title	Specification of FlexRay Network Management
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	2.0.0
Document Status	Draft
Part of Release	2.1
Revision	0014

Document Change History

Date	Version	Changed by	Change Description
31.01.2007	2.0.0	AUTOSAR Administration	• FlexRay NM machine has been reworked completely • Support of Hardware NM Vector of communication controller • Separation of NM vote and data (transmission of NM vote and data can be done with different update intervals) • FlexRay NM State machine is now synchronised to FlexRay communication cycle • Legal disclaimer revised • Release Notes added • “Advice for users” revised • “Revision Information” added
10.07.2005	1.0.0	AUTOSAR Administration	Initial release

Release Notes

Errata and known deficiencies

All modifications planned in the scope of Release 2.1 for the incorporation into this document are completed. The document, however, has not yet undergone the necessary finalization.

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).
Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

Table of Figures

1 Introduction and functional overview

This document describes the concept, core functionality, optional features, interfaces and configuration issues of the AUTOSAR FlexRay Network Management (FrNm).

The AUTOSAR FlexRay Network Management is a hardware independent protocol that can only be used on FlexRay (for limitations see 4.1). Its main purpose is to coordinate the transition between normal operation and bus-sleep mode of the network.

In addition to the core functionality optional features are provided e.g. to implement a service to detect all present nodes or to detect if all other nodes are ready to sleep.

2 Acronyms, abbreviations and glossary

Acronym:	*Description:*
CC	Communication Controller
NM	Network Management
WCET	Worst Case Execution Time
DET	Development Error Tracer. AUTOSAR Module for detection and reporting of errors during development.
DEM	Diagnostic Event Manager. AUTOSAR Module which is a sub-component of the diagnostic module within AUTOSAR. It is responsible for processing and storing diagnostic events (errors) and associated freeze frame data. Further, the DEM provides fault information to the DCM (e.g. read all stored DTCs from the error memory).

Abbreviation:	*Description:*
FrIf	Abbreviation for the FlexRay Interface
FrNm	Abbreviation for the FlexRay specific Network Management
Nm	Abbreviation for the generic Network Management

Term:	*Definition:*
Bus-Sleep Mode	Network mode where all interconnected communication controllers are in the sleep mode.
NM Channel	Logical communication path associated with the NM-cluster.
NM Cluster	Logical part of the network coordinated with use of the NM algorithm.
NM Data Cycle	Number of cycles necessary for all nodes to be able to send NM Data at least once.
NM Message	Packet of information exchanged for purposes of the NM algorithm.
NM Repetition Cycle	Number of repetitions of an NM Voting Cycle. Used to improve the reliability of the voting.
NM Slot	Slot reserved for purposes of the network management.
NM Timeout	Timeout in the NM algorithm that initiates transition into Bus-Sleep Mode.
NM User Data	Supplementary application specific piece of data that is sent independent of the NM Vote on the bus.
NM Voting Cycle	Number of cycles necessary for all nodes to be able to vote at least once.
NM-Vote	Information transmitted using the FlexRay Bus indicating the vote of a ECU to keep the bus awake
NM-Data	Data related to NM transmitted using the FlexRay Bus.

3 Related documentation

3.1 Input documents

[1] AUTOSAR Layered Architecture
https://svn.autosar.org/repos/10Releases/
AUTOSAR_LayeredSoftwareArchitecture.pdf

[2] AUTOSAR General Requirements on Basic Software Modules
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_General.pdf

[3] Requirements on Network Management
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_NM.pdf

3.2 Related standards and norms

[1] FlexRay Communications System Specifications, V2.1

3.3 Related AUTOSAR documents

[4] Specification of Communication Manager
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_ComManager.pdf

[5] Specification of Generic Network Management Interface
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_NMInterface.d pdf

[6] Specification of FlexRay Interface
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_FlexRay_Interface. pdf

[7] Specification of ECU State Manager
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_ECU_StateManager.pdf

[8] Specification of Operating System
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_OS.pdf

[9] Specification of Diagnostics Event Manager
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_DEM.pdf

[10] Specification of Development Error Tracer

https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_DET.pdf

[11] Specification of Standard Types
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_StandardTypes.pdf

[12] Specification of Platform Types
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_PlatformTypes.pdf

[13] Specification of Compiler Abstraction
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_CompilerAbstraction.pdf

4 Constraints and assumptions

4.1 Limitations

1. FlexRay NM can be applied to FlexRay communication systems that support bus sleep mode and that are implemented with appropriate wakeup mechanisms.

2. One instance of FlexRay NM can be applied to only one instance of FlexRay Interface within the same ECU.

3. One instance of FlexRay NM can be applied to only one FlexRay NM-Cluster in one FlexRay network. One FlexRay NM-Cluster can have only one instance of FlexRay NM.

4. FlexRay NM can be applied to both channels of the same FlexRay Bus at the same time.

The Figure 4-1 (below) presents an AUTOSAR NM stack within an example ECU belonging to a FlexRay NM-clusters.

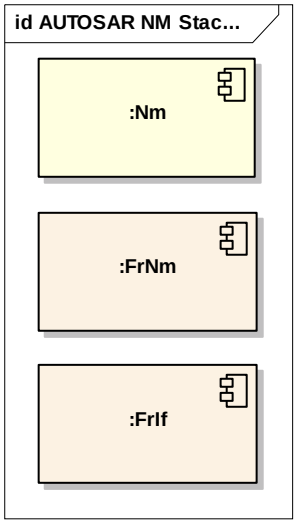

Figure 4-1 AUTOSAR NM Stack on FlexRay

4.2 Applicability to car domains

AUTOSAR NM can be applied to any car domain, wherever FlexRay technology is used, under limitations provided above.

5 Dependencies to other modules

FlexRay NM provides services to the Network Management Interface (Nm) and uses services of FlexRay Interface

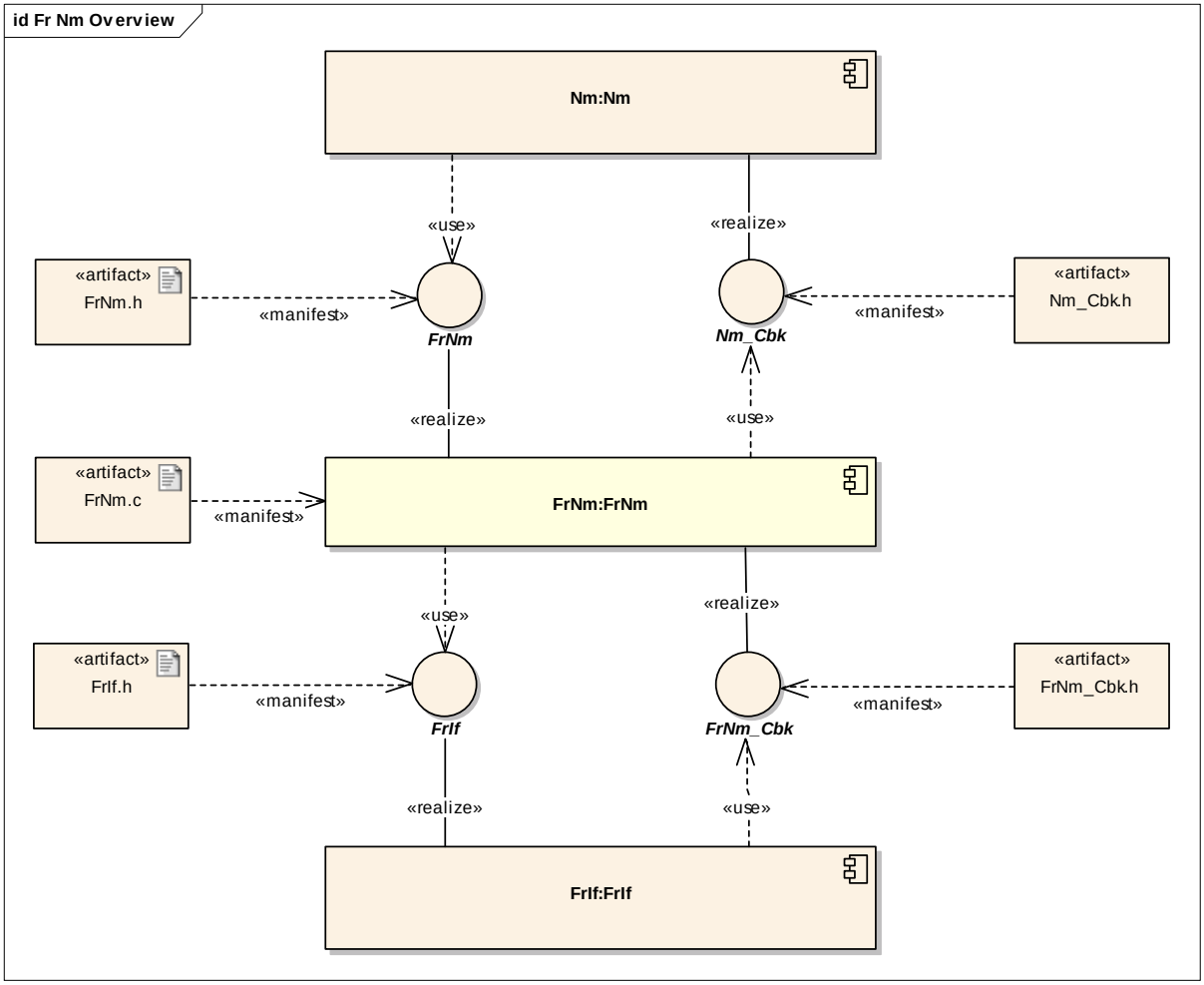

Figure 5-1 NM Overview

Note: In addition to the modules depicted in Figure 5-1(above), FlexRay Nm uses some additional modules (like the DET and DEM). A complete list can be found in 5.1.2.

FRNM220: The FlexRay NM shall use only OS objects and/or related OS services according to the table defined in [2] [BSW00429].

5.1 File structure

5.1.1 Code file structure

FRNM064: The following source code files shall be provided by the FrNm module.

- `FrNm.c` (for implementation of provided functionality)

- FrNm_Lcfg.c (for link time configurable parameters)
- FrNm_PBcfg.c (for post build time configurable parameters)

5.1.2 Header file structure

FRNM065: The following header files shall be provided and included within the FrNm module.

- FrNm.h (for declaration of provided interface functions)
- FrNm_Cbk.h (for declaration of provided call-back functions)
- FrNm_Cfg.h (for configuration parameters)
- FrNm_pCf.h (for pre-compile time parameters)

FRNM066: The following header files shall be included within the FrNm module.

- Std_Types.h (for AUTOSAR standard types - Note: Platform_Types.h (for platform specific types) and Compiler.h (for compiler specific language extensions) are indirectly included via AUTOSAR standard types)
- FrIf.h (for interface of FlexRay Interface)
- Nm_Cbk.h (for callbacks of Nm)
- Det.h (for interface of DET – optional, included only if Det is configured)
- Dem.h (for interface of DEM)
- Nm.h (for common NM types)
- SchM_FrNm.h (for services of the Basic Software Scheduler)

FRNM070: The following configuration files shall be included within the FRNM module.

- FrIf_Cfg.h (for configuration of FlexRay Interface)
- Det_Cfg.h (for configuration of DET – optional, included only if Det is configured)
- Dem_Cfg.h (for configuration of DEM)

6 Requirements traceability

Document: AUTOSAR General Requirements on Basic Software [2].

Requirement	***Satisfied by***
4.2 Functional Requirements	
4.2.1 Configuration	
[BSW00344] Reference to link-time configuration	OK, see 8.3.1
[BSW00404] Reference to post build time configuration	OK, see 8.3.1
[BSW00405] Reference to multiple configuration sets	OK, see 8.3.1
[BSW00345] Storage of Pre-compile-time configuration	OK, see FRNM018
[BSW159] Tool-based configuration	OK, see FRNM020
[BSW167] Static configuration checking	OK, see 10
[BSW171] Configurability of optional functionality	OK, see 10
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	OK, SWS tempalate used
[BSW00380] Separate C-Files for configuration parameters	OK, see FRNM064
[BSW00381] Separate H-File for configuration parameters	OK, see FRNM065
[BSW00412] Separate H-File for configuration parameters	OK, see FRNM065
[BSW00383] List dependencies of configuration files	OK, see 5
[BSW00384] List dependencies to other modules	OK, see 5
[BSW00387] Specify the configuration class of callback function	OK, see 8
[BSW00388] Introduce containers	OK, see 10
[BSW00389] Containers shall have names	OK, see 10
[BSW00390] Parameter content shall be unique within the module	OK, see 10
[BSW00391] Parameter shall have unique names	OK, see 10
[BSW00392] Parameters shall have a type	OK, see 10
[BSW00393] Parameters shall have a range	OK, see 10
[BSW00394] Specify the scope of the parameters	OK, see 10
[BSW00395] List the required parameters (per parameter)	OK, see 10
[BSW00396] Configuration classes	OK, see 10
[BSW00397] Pre-compile-time parameters	OK, see 10
[BSW00398] Link-time parameters	OK, see 10
[BSW00399] Loadable Post-build time parameters	OK, see 10
[BSW00400] Selectable Post-build time parameters	OK, see 10
[BSW00401] Creating multiplicity	OK, see 10
[BSW00402] Published information	OK, see 10
4.2.2 Wake-Up	
[BSW00375] Notification of wake-up reason	N/A, Bus-Interface is responsible for notification of the wakeup reason.
4.2.3 Initialization	
[BSW101] Initialization interface	OK, see FRNM028, FRNM033
[BSW00416] Sequence of initialization	OK, see FRNM029
[BSW00406] Check module initialization	OK, see FRNM071, FRNM073
4.2.4 Normal Operation	
[BSW168] Diagnostic Interface of SW components	N/A, FlexRay NM does not need to be tested by external devices.
[BSW00407] Function to read out published parameters	OK, see 8.3.16
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	N/A
[BSW00424] BSW main processing function task allocation	OK, see 7.7.3
[BSW00425] Trigger conditions for schedulable objects	OK, see 7.7.3
[BSW00426] Exclusive areas in BSW modules	N/A, FlexRay NM task activation is coupled to

	the FlexRay Schedule
[BSW00427] ISR description for BSW modules	N/A, FlexRay NM does not use ISR functions
[BSW00428] Execution order dependencies of main processing functions	OK, see 7.7.3
[BSW00429] Restricted BSW OS functionality access	OK, see FRNM220
[BSW00431] The BSW Scheduler module implements task bodies	N/A, FlexRay NM task activation is coupled to the FlexRay Schedule
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	OK, see FRNM176
[BSW00433] Calling of main processing functions	N/A, FlexRay NM task activation is coupled to the FlexRay Schedule
[BSW00434] The Schedule Module shall provide an API for exclusive areas	N/A, FlexRay NM does not support exclusive areas.
4.2.5 Shutdown Operation	
[BSW00336] Shutdown interface	N/A, FlexRay NM is used for coordinated shutdown of bus communication.
4.2.6 Fault Operation and Error Detection	
[BSW00337] Classification of errors	OK, see FRNM021
[BSW00338] Detection and Reporting of development errors	OK, see FRNM022, FRNM049
[BSW00369] Do not return development error codes via API	OK, see FRNM056, FRNM057
[BSW00339] Reporting of production relevant errors and exceptions	OK, see FRNM023
[BSW00417] Reporting of Error Events by Non-Basic Software	N/A, FlexRay NM is part of the Basic Software
[BSW00323] API parameter checking	OK, see FRNM024
[BSW004] Version check	OK, see FRNM074
[BSW00409] Header files for production code error IDs	OK, see 5
[BSW00385] List possible error notifications	OK, see 7.11
[BSW00386] Configuration for detecting an error	OK, see 7.11
4.3 Non-functional Requirements	
4.3.1 Software Architecture Requirements	
[BSW161] Microcontroller abstraction	N/A, already abstracted
[BSW162] ECU layout abstraction	N/A, already abstracted
[BSW005] No hard coded horizontal interfaces within MCAL	N/A, FlexRay NM is not located within MCAL
[BSW00415] User dependent include files	OK, see 5
4.3.2 Software Integration Requirements	
[BSW164] Implementation of interrupt service routines	N/A, no interrupt routine implemented
[BSW00325] Runtime of interrupt service routines	N/A, no interrupt routine implemented
[BSW00326] Transition from ISRs to OS tasks	N/A, no interrupt routine implemented
[BSW00342] Usage of source code and object code	OK, see 10
[BSW00343] Specification and configuration of time	OK, see 10
[BSW160] Human-readable configuration data	OK, see 10
4.3.3 Software Module Design Requirements	
4.3.3.1 Software quality	
[BSW007] HIS MISRA C	OK, HIS MISRA C is used

4.3.3.2 Naming conventions	
[BSW00300] Module naming convention	OK, naming convention used respectively
[BSW00413] Accessing instances of BSW modules	OK, FlexRay NM can be accessed in instances
[BSW00347] Naming separation of drivers	N/A, FlexRay NM is no driver module
[BSW00305] Self-defined data types naming convention	OK, naming convention used respectively
[BSW00307] Global variables naming convention	OK, naming convention used respectively
[BSW00310] API naming convention	OK, naming convention used respectively
[BSW00373] Main processing function naming convention	OK, naming convention used respectively
[BSW00327] Error values naming convention	OK, naming convention used respectively
[BSW00335] Status values naming convention	OK, naming convention used respectively
[BSW00350] Development error detection keyword	OK, keyword used
[BSW00408] Configuration parameter naming convention	OK, see 10
[BSW00410] Compiler switches shall have defined values	OK, see 10
[BSW00411] Get version info keyword	OK, see 10
4.3.3.3 Module file structure	
[BSW00346] Basic set of module files	OK, see 5.1
[BSW158] Separation of configuration from implementation	OK, see 5.1 and 10
[BSW00314] Separation of interrupt frames and service routines	N/A, no interrupt frames implemented
[BSW00370] Separation of callback interface from API	OK, see FRNM065
4.3.3.4 Standard header files	
[BSW00348] Standard type header	OK, see 5
[BSW00353] Platform specific type header	OK, see 5
[BSW00361] Compiler specific language extension header	OK, see 5
4.3.3.5 Module Design	
[BSW00301] Limit imported information	OK, see 5
[BSW00302] Limit exported information	OK, see 5
[BSW00328] Avoid duplication of code	OK, see 8
[BSW00312] Shared code shall be reentrant	OK, see 8
[BSW006] Platform independency	OK, see FRNM075
4.3.3.6 Types and keywords	
[BSW00357] Standard API return type	OK, see 8
[BSW00377] Module specific API return types	OK, see 8
[BSW00304] AUTOSAR integer data types	OK, see 8
[BSW00355] Do not redefine AUTOSAR integer data types	OK, see 8
[BSW00378] AUTOSAR boolean type	OK, see 8
[BSW00306] Avoid direct use of compiler and platform specific keywords	OK, see 8
4.3.3.7 Global data	
[BSW00308] Definition of global data	OK, see 10
[BSW00309] Global data with read-only constraint	OK, see 10
4.3.3.8 Interface and API	
[BSW00371] Do not pass function pointers via API	OK, see 8
[BSW00358] Return type of init() functions	OK, see 8.3.1
[BSW00414] Parameter of init function	OK, see 8.3.1
[BSW00376] Return type and parameters of main processing functions	OK, see 8.5.1
[BSW00359] Return type of callback functions	OK, see 8
[BSW00360] Parameters of callback functions	OK, see 8
[BSW00329] Avoidance of generic interfaces	OK, see 8

[BSW00330] Usage of macros / inline functions instead of functions	OK, see 8
[BSW00331] Separation of error and status values	OK, see FRNM021 and 8.2
4.3.4 Software Documentation Requirements	
[BSW009] Module User Documentation	N/A, implantation specific
[BSW00401] Documentation of multiple instances of configuration parameters	N/A, implantation specific
[BSW172] Compatibility and documentation of scheduling strategy	N/A, implantation specific
[BSW010] Memory resource documentation	N/A, implantation specific
[BSW00333] Documentation of callback function context	N/A, implantation specific
[BSW00374] Module vendor identification	OK, see 10
[BSW00379] Module identification	OK, see 10
[BSW003] Version identification	OK, see 10
[BSW00318] Format of module version numbers	OK, see 10
[BSW00321] Enumeration of module version numbers	OK, see 10
[BSW00341] Microcontroller compatibility documentation	N/A, implantation specific
[BSW00334] Provision of XML file	N/A, implantation specific

Document: AUTOSAR Requirements on Basic Software Modules (NM) [3]

Requirement	***Satisfied by***
[BSW150] Configuration of functionality Note: BSW150 contains a list of requirements which will be traced in detailed in the following. This entry can be used only as overview.	FRNM179, FRNM180, FRNM077, FRNM221, FRNM213, FRNM187
[BSW151] Integration into running system	FRNM033 together with FRNM175. Note: The FlexRay start-up and reintegration is handled by the ComM and the FrIf. FrNm requires a running FlexRay communication (7.2.2).
[BSW043] Bus Traffic without NM Initialization	FRNM221
[BSW044] Independency of Underlying Communication System	N/A, FlexRay NM is bus specific
[BSW045] Channel Selective Wake-Up/Shutdown	FRNM034
[BSW046] Trigger of startup of all Nodes at any Point in Time	FRNM168
[BSW047] Bus Sleep and Bus Keep Awake Service	FRNM144, FRNM145
[BSW048] Bus Sleep Mode	FRNM101
[BSW050] Bus State Information	FRNM104
[BSW051] Bus State Change Information	FRNM106
[BSW052] Notification that all other ECUs are ready to sleep	FRNM181
[BSW02509] Notification that at least one other node is not ready to sleep anymore	FRNM185
[BSW02503] Sending user data	FRNM043
[BSW02504] Reading user data	FRNM044
[BSW153] Detection of present nodes	OK, see 7.3.3.1
[BSW02508] Unambiguous node identification per bus	N/A, see bus specific SWS
[BSW02505] Sending node identifier	FRNM222
[BSW02506] Reading node identifier	FRNM047
BSW02507] Reading the configured NM Identifier of the local node	FRNM046
[BSW02511] Configurable Role in Cluster Shutdown	FRNM187
[BSW053] Deterministic Behavior in Case of Bus Unavailability	FRNM035, FRNM223, FRNM224
[BSW137] Communication system error handling	FRNM035
[BSW136] Coordination of coupled networks	N/A, FlexRay NM does not support coupled networks.
[BSW140] Compliance with OSEK NM on a gateway	N/A, see NM gateway SWS
[BSW054] Deterministic Time for Bus Sleep	FRNM101
[BSW142] Limitation of NM bus load	OK (limited by configured number of

	NM-messages per FlexRay communication cycle)
[BSW143] Deterministic average bus load	OK (guaranteed by the FlexRay media access control mechanism)
[BSW144] ECU cluster size	FRNM179
[BSW145] Robustness against NM message losses	N/A, currently not supported
[BSW146] Robustness against NM message jitter	N/A, FlexRay does not have a message jitters.
[BSW147] Processor independent algorithm	FRNM225
[BSW149] Configurable Timing	FRNM036
[BSW154] Bus independency of basic API	FRNM034
[BSW148] Separation of Communication system dependent parts	N/A, FlexRay NM is bus specific
[BSW139] Compliance with OSEK NM on one cluster	N/A, see NM gateway SWS
[BSW02510] Immediate Transmission Confirmation	N/A, CAN specific Requirement
[BSW02512] CommunicationControl (28 hex) service support	Not supported
[BSW02501] Number of hardware send buffer	FRNM002
[BSW02502] Number of hardware receive buffer	FRNM002, FRNM015
[BSW02508] Unambiguous node identification per bus	FRNM003

7 Functional specification

7.1 Coordination algorithm

The AUTOSAR FlexRay NM is based on a decentralized direct network management strategy, which means that every network node individually performs self-sufficient NM activities self-sufficient based only on the NM-messages that are received or transmitted within the communication system.

The AUTOSAR FlexRay NM coordination algorithm is based on periodic NM-Vote messages received by all nodes in the cluster. Reception of an NM-Vote message indicates that the sending node wants to keep the NM-cluster awake. If any node is ready to go to the Bus-Sleep Mode, it stops sending NM-messages, but as long as NM-messages from other nodes are received, it postpones transition to the Bus-Sleep Mode. Ultimately, if a designated timer elapses because no NM-messages are received anymore, the node initiates transition to the Bus-Sleep Mode.

If any node in the NM-cluster requires bus-communication, it can “wake-up”[1] the NM-cluster from the Bus-Sleep Mode by transmitting NM-Vote messages. For more details concerning the wakeup procedure, please refer to the Mode Management (see[6], [7]).

FlexRay Network Management is responsible for the following functionalities:

- Periodic Update of FlexRay NM-PDU's
- Encoding and Decoding of FlexRay NM-PDU's
- Transmission Error Handling for FlexRay NM-PDU's
- Notification of the Network Management Interface (Nm) regarding changes of the FlexRay NM state machine

A special case is the possibility to configure the FrNm of a node as “passive”. Such a “passive node” will listen to the NM-messages on the FlexRay Bus (to determine whether to stay awake or to go to sleep), but will not send any NM-messages itself. Thus such a node will follow the decisions the network global consensus but will not influence it. A more detailed description of the requirements of passive nodes can be found in chapter 7.8.4 on page 36).

The main concept of the AUTOSAR FlexRay NM coordination algorithm can be defined by the following key-requirements:

FRNM100: Every network node shall transmit periodic messages as an indication that the node requires bus-communication.

Note: Since the transmission of frames in the static segment of FlexRay cannot be stopped, the presence of an NM-message cannot be directly used as an indication whether the bus is required or not. Other means must be used to signal the “non

[1] The “wake-up” of the NM-cluster shall not be confused with “wake-up” of the FlexRay cluster, which is not part of the wake-up procedure of the FlexRay NM, as the FlexRay NM requires an already started FlexRay cluster.

transmission" of NM-Data and NM-Vote, e.g. by usage of the FlexRay Protocol HW NW Management support.

FRNM101: If bus communication is released and there are no NM-messages on the bus for a configurable amount of time determined by **`FRNM_READY_SLEEP_CNT`** (configuration parameter), the transition into the Bus-Sleep Mode shall be performed.

Note: The FRNM_READY_SLEEP_CNT is a factor of the Repetition Cycle time. To get an absolute time this factor has to be multiplied by the time need for one Repetition Cycle
Example: FRNM_READY_SLEEP_CNT = 3; Repetition Cycle = 4 vote cycles; Vote Cycle = 1 FlexRay Cycle; FlexRay Cycle = 5 msec: Repetition Cycle time = 3* 4* 1* 5 msec = 60 msec)

FRNM102: The AUTOSAR FlexRay NM state machine shall contain states, transitions and triggers required for the AUTOSAR FlexRay NM coordination algorithm as seen from the point of view of one single node in the NM-cluster.

FRNM103: Transitions in the AUTOSAR FlexRay NM state machine shall be triggered by calls of selected interface functions or by expiration of internal timers or counters.

Note: The FlexRay NM will be exclusively controlled (via the NM interface) by the AUTOSAR Communication Manager Module (ComM), therefore all external introduced state changes will be triggered by the ComM. Internal timers of the FlexRay NM will be described in chapter 7.

FRNM001: The Module Generic Network Management Interface [5]) shall apply to FlexRay NM.

FRNM075: FlexRay NM realization shall be processor and compiler independent.

FRNM168: State changes in the FlexRay NM state machine shall be synchronized to the FlexRay periodic Schedule by locating these changes to the borders of an NM Repetition Cycle.

Rationale: FlexRay cluster wide synchronization (based on a periodic repetition of its communication scheme, the so called Cycle – see [1]), is used by the FlexRay NM to align the state changes to the borders of a NM Repetition Cycle to guarantee a synchronous behavior of the NM state machines of all ECUs in the NM cluster.

7.2 Interconnection to other Modules

7.2.1 Interconnection to Nm

The module Generic NM ([5]) is intended to be a "thin" layer which contains mainly c-macros for naming conventions. For example, **`Nm_Init`** will be directly translated to **FrNm_Init**. In addition Generic NM is stateless and hence provides the FrNm services unmodified to the upper layers (e.g., ComM).

7.2.2 Interconnection to ComM

The FlexRay NM relies on the fact that communication on FlexRay Bus shall be available before the FlexRay NM is started. This is needed to prevent race conditions caused by the inability to transmit NM-Messages. The startup of the FlexRay communication schedule is a complex mechanism which may take time. The AUTOSAR FlexRay Driver and Interface (FrIf) will oversee the startup of the FlexRay Bus, but ComM has to guarantee that the FlexRay Network has been started before starting FlexRay NM.

The states of the FlexRay NM and the ComM are interlinked as follows:

ComM WG please fill

7.2.3 Interconnection to FrIf

The FlexRay NM will use the services of the FrIf as described in chapter 8.4 and 8.6.1.1.

The FlexRay NM requires full communication on the FlexRay Bus (see also 7.2.2). Therefore all FlexRay NM States require the FrIf to be in the FRIF_ONLINE state.

7.3 Operational modes

In the following chapter operational modes of the AUTOSAR FlexRay NM coordination algorithm are described in detail. Figure 8-2 (in chapter 8.9 on page 57) shows the detailed UML state chart of the FlexRay NM.

FRNM104: The service call **`FrNm_GetState`** shall provide consistent information about the current state and the current mode of the NM state machine.

Note: Consistency between the provided values and the current values of the state and mode shall be guaranteed.

FRNM105: The AUTOSAR FlexRay NM shall consist of three operational modes:
- Bus-Sleep Mode
- Synchronize Mode
- Network Mode

FRNM106: Changes of the AUTOSAR FlexRay NM operational modes shall be notified to the upper layer by means of callback functions.

FRNM118: The FlexRay NM shall store the Repeat Message Request in a flag (**`FrNm_RepeatMessage`**).

Note: The **`FrNm_RepeatMessage`** flag is used to store the request only – it is not a status variable and will not be returned on a **`FrNm_GetState`** service call, as state changes will be done on Repetition Cycle boundaries.

FRNM167: The FlexRay NM shall store the Network Request in a flag (**`FrNm_NetworkRequested`**).

Note: The **`FrNm_NetworkRequested`** flag is used to store the request only – it is not a status variable and will not be returned on a **`FrNm_GetState`** service call, as state changes will be done on Repetition Cycle boundaries.

7.3.1 Bus-Sleep Mode

The Bus Sleep Mode is the default mode on the start of the FrNm State Machine, where it remains unless the NM is started (either with a passive startup request or with a network request) or the power of the CPU is switched off.
In Bus Sleep Mode the communication controller can be switched into the sleep mode where wakeup mechanisms are activated and power consumption is reduced to a minimal level. The corresponding functionality (shut down of FlexRay, power down the CPU) will be implemented in other modules. The FlexRay NM will only signal the readiness for Sleep Mode.

FRNM134: When Bus-Sleep Mode is entered, the FlexRay NM shall notify the upper layer by calling **`Nm_BusSleepMode`**, except when Bus-Sleep Mode is entered by default at initialization.

Note: The FlexRay NM will notify the ComM via the Generic NM Interface (macro adaptation layer).

FRNM135: When the Bus Sleep Mode is entered **`FrNm_RepeatMessage`** shall be set to FALSE.

FRNM136: When the Bus Sleep Mode is entered **`FrNm_NetworkRequest`** shall be set to FALSE.

FRNM137: When the Bus Sleep Mode is entered the transmission and reception of NM-Data and NM-Vote shall be deactivated.

FRNM138: If the service **`FrNm_PassiveStartUp`** is called in the Bus Sleep Mode, the Bus-Sleep Mode shall be left and the Synchronize Mode shall be entered.

FRNM139: If the service **`FrNm_NetworkRequest`** is called in the Bus Sleep Mode, the Bus-Sleep Mode shall be left, the networkNetwork Request Flag shall be set to requested (**`FrNm_NetworkRequested`** shall be set to TRUE) and the Synchronize Mode shall be entered.

FRNM175: At successful reception of a NM-message in the Bus-Sleep Mode, the FlexRay NM shall notify the upper layer by calling **`Nm_NetworkStartIndication`**

Rationale: This is required to avoid race conditions and state inconsistency between Network and Mode Management. NM-message reception handling in Bus-Sleep Mode is dependent on the current state of the ECU shutdown/startup process.

Note: The FlexRay NM will notify the ComM via the Generic NM Interface (macro adaptation layer).

7.3.2 Synchronize Mode

In the Synchronize Mode the FrNm state machine is synchronized to the FrNm Repetition Cycle. This is necessary as the FlexRay NM is dependent on state changeds being synchronized across the NM Cluster.

FRNM140: When Synchronize Mode is entered, the FlexRay NM shall notify the upper layer by calling **`Nm_SynchronizeMode.`**

Note: The FlexRay NM will notify the ComM via the Generic NM Interface (macro adaptation layer).

FRNM141: If the network is requested in Synchronize Mode (**`FrNm_NetworkRequest`**) **`FrNm_NetworkRequested`** shall be set to TRUE

FRNM142: If the network is released in Synchronize Mode (**`FrNm_NetworkRelease`**) **`FrNm_NetworkRequested`** shall be set to FALSE

FRNM143: Synchronize Mode shall be left at the first boundary between two NM Repetition Cycles and Network Mode shall be entered

7.3.3 Network Mode

FRNM107: The Network Mode shall consist of three internal states:
- Repeat Message State
- Normal Operation State
- Ready Sleep State

FRNM115: State changes into and within the Network Mode shall be synchronized to the boundary between two NM Repetition Cycles.

Rationale: The FlexRay NM defines a number of FlexRay cycles as NM Repetition Cycle to improve the reliability of the NM vote transmission. Within a NM Repetition Cycle the NM is not allowed to change the NM-vote. For details see chapter 7.9 on page 37.

FRNM108: When the Network Mode is entered the Repeat Message State shall be entered.

FRNM109: When the Network Mode is entered **`FrNm_RepeatMessage`** shall be set to TRUE.

FRNM110: When the Network Mode is entered, the FlexRay NM shall notify the upper layer by calling **`Nm_NetworkMode`**.

Note: The FlexRay NM will notify the ComM via the Generic NM Interface (macro adaptation layer).

FRNM133: When the Network Mode is left the Bus Sleep Mode shall be entered.

FRNM111: At successful reception of a Repeat Message Request in the Network Mode, **FrNm_RepeatMessage** shall be set to TRUE.

Note: FlexRay NM will detect Repeat Message Request only if the Node detection service is activated (**FRNM_NODE_DETECTION_ENABLED**).

FRNM112: If the NM Repeat Message Timer (**FrNm_RepeatMessageTimer**) has expired in the Network Mode, .**FrNm_RepeatMessage** shall be set to FALSE.

FRNM113: If the network is requested in Network Mode (**FrNm_NetworkRequest**) **FrNm_NetworkRequested** shall be set to TRUE.

Note: The **FrNm_NetworkRequest** service will not be available if the FrNm is configured as Passive Node (see 7.8.4)

FRNM114: If the network is released in Network Mode (**FrNm_NetworkRelease**) **FrNm_NetworkRequested** shall be set to FALSE.

Note: The **FrNm_NetworkRelease** service will not be available if the FrNm is configured as Passive Node (see 7.8.4)

FRNM119: The NM Repeat Message Timer shall be active for a configurable amount of time determined by **FRNM_REPEAT_MESSAGE_TIME** (configuration parameter).

FRNM226: When the **FrNm_RepeatMessageRequest** service is called in the Network Mode, **FrNm_RepeatMessage** shall be set to TRUE if the Node detection service is activated (**FRNM_NODE_DETECTION_ENABLED**).

FRNM228: The **FrNm_RepeatMessageRequest** service shall be ignored if the Node detection service is NOT activated (**FRNM_NODE_DETECTION_ENABLED** set to **OFF**).

7.3.3.1 Repeat Message State

For nodes that are not configured as passive the Repeat Message State ensures, that any transition from Bus Sleep to the Network Mode becomes visible to the other nodes on the network. Additionally it ensures that any node stays active for a minimum amount of time. Optionally it can be used for detection of present nodes.

FRNM116: When Repeat Message State is entered the transmission of NM-Data shall be activated if the node is configured for NM-Data transmissions and the node shall vote to keep the cluster awake if the node is configured to permit voting.

FRNM117: When the Repeat Message State is entered the NM Repeat Message Timer (**`FrNm_RepeatMessageTimer`**) shall be restarted.

FRNM120: The Repeat Message State shall be left at the boundary between two NM Repetition Cycles if **`FrNm_RepeatMessage`** is set to FALSE

FRNM121: When Repeat Message State is left (see FRNM120), the Normal Operation State shall be entered if the network has been requested (**`FrNm_NetworkRequested`** is set to TRUE – see FRNM113)

FRNM122: When Repeat Message State is left (see FRNM120), the Ready Sleep State shall be entered if the network has been released (**`FrNm_NetworkRequested`** is set to FALSE – see FRNM114)

7.3.3.2 Normal Operation State

The Normal Operation State ensures that any node can keep the NM-cluster awake as long as the network is requested.

Note: This State will not be reached if the node is configured as “passive node” (FRNM187). It is up to the implementation to optimize this state and remove the code corresponding to this state.
FRNM123: When the Normal Operation State is entered the transmission of NM-Data shall be activated and the Node shall vote for keeping the cluster awake (send “positive” NM-Votes).

FRNM124: The Normal Operation State shall be left at the boundary between two NM Repetition Cycles and the Repeat Message State shall be entered if a Repeat Message Request has been detected (**`FrNm_RepeatMessage`** is set to TRUE – see FRNM111)

FRNM125: The Normal Operation State shall be left at the boundary between two NM Repetition Cycles and the the Ready Sleep State shall be entered if no Repeat Message Request is active (**`FrNm_RepeatMessage`** is set to FALSE) and the network has been released (**`FrNm_NetworkRequested`** is set to FALSE – see FRNM114)

7.3.3.3 Ready Sleep State

The Ready Sleep State ensures that any node in the NM-cluster waits to transition to the Bus-Sleep Mode as long as any other node keeps the NM-cluster awake.

FRNM126: When the Ready Sleep State is entered the transmission of NM-Data shall be deactivated and the Node shall not vote for keeping the cluster awake (send “negative” NM-Votes).

FRNM127: When the Ready Sleep State is entered from another state the Ready Sleep Counter (**FrNm_ReadySleepCnt**) shall be initialized with the start value **FRNM_READY_SLEEP_CNT** (configurable parameter) – see FRNM101

FRNM128: When a NM-Vote to keep the cluster awake is detected the Ready Sleep Counter (**FrNm_ReadySleepCnt**) shall be initialized with the start value **FRNM_READY_SLEEP_CNT** (configurable parameter).

FRNM129: The Ready Sleep State (and the Network Mode) shall be left and the Bus Sleep Mode shall be entered if at the end of the NM Repetition Cycle the Ready Sleep Counter has expired (**FrNm_ReadySleepCnt** < 1).

FRNM130: The Ready Sleep State shall be left and the Repeat Message State shall be entered if at the boundary between two NM Repetition Cycles a Repeat Message Request is active (**FrNm_RepeatMessage** is set to TRUE – see FRNM111)

FRNM131: The Ready Sleep State shall be left and the Normal Operation State shall be entered if at the boundary between two NM Repetition Cycles no Repeat Message Request is active (**FrNm_RepeatMessage** is set to FALSE) and the network has been requested (**FrNm_NetworkRequested** is set to TRUE – see FRNM113)

FRNM132: The Ready Sleep Counter (**FrNm_ReadySleepCnt**) shall be decremented at the end of an NM Repetition Cycle if no Repeat Message Request is active (**FrNm_RepeatMessage** is set to FALSE) and the network has been released (**FrNm_NetworkRequested** is set to FALSE).

7.4 Network states

Network states (i.e. ‘requested’ and ‘released’) are two additional “states” of the AUTOSAR FlexRay NM state machine that exist in parallel to the state machine described in chapter 8.9. Network states distinguish whether the software components need to communicate on the bus (the network state is then ‘requested’); or not (the bus network state is then ‘released’). Note that if the network is released an ECU may still communicate because some other ECU still requests the network.

FRNM144: Service call **FrNm_NetworkRequest** shall request the network, i.e., the network requested flag shall be changed to ‘requested’ (see FRNM113).

Note: The **FrNm_NetworkRequest** service will not be available if the FrNm is configured as Passive Node (see 7.8.4)

FRNM145: Service call **FrNm_NetworkRelease** shall release the network, i.e., the network requested flag shall be changed to ‘released’ (see FRNM114).

Note: The **FrNm_NetworkRelease** service will not be available if the FrNm is configured as Passive Node (see 7.8.4)

7.5 Initialization and Startup

FRNM071: The FlexRay NM shall store the initialization status in a private variable

FRNM072: After reset the initialization status shall be set to **`NM_UNINIT`**.

FRNM073: After successful initialization the initialization status shall be set to **`NM_INIT`**, otherwise it shall be set to **`NM_UNINIT`**.

FRNM028: Service call **`FrNm_Init`** shall initialize the FlexRay NM.

FRNM059: The function **`FrNm_Init`** shall select active configuration set provided set by means of a configuration pointer parameter.

FRNM029: The FlexRay NM shall be initialized after the corresponding FlexRay Interface is initialized and before any other FlexRay NM service is called.

FRNM030: After initialization the periodic transmission of NM-Vote and NM-Data shall be deactivated. (see FRNM100, FRNM137)

FRNM032: If FlexRay NM is not initialized, a call of any FlexRay NM service shall be rejected and respective error code shall be returned.

FRNM042: After initialization the Reserved Bytes in the NM-Data and NM-Vote PDU shall be set to 0.

FRNM045: After initialization the User Data bytes shall be set to FFh.

FRNM033: Service call **`FrNm_Init`** shall re-initialize the FlexRay NM.

FRNM221: Bus traffic should not be prohibited if AUTOSAR FlexRay NM is not initialized

7.6 Communication

Using NM-Messages FlexRay NM provides mechanisms for information exchange for purposes of shutdown coordination. These messages are sent according to the schedule configured within each ECU and coordinated across the NM Cluster.

7.6.1 General requirements

FRNM015: For every node the FrIf shall be configured to receive all NM vote messages that are not aggregated by the FlexRay controller.

FRNM058: The FlexRay NM decisions shall be influenced by every received NM-Vote and every NM-Vote aggregated by the FlexRay controller.

FRNM205: A FlexRay NM Message shall be configurable to contain NM-Vote, NM-Data or both.

FRNM147: FlexRay NM shall be able to transmit NM Data and NM Vote separately.

Rationale: The voting algorithm of FlexRay is kept independent of the transmission of the NM data as the FlexRay Protocol provides a HW support for sending and receiving NM votes (see [1]). To use this feature and to increase the update rate of NM-Votes (compared to the update rate of the NM-Data), the transmission of NM-Data and NM-Vote may be separated.

FRNM160: It shall be configurable which NM-message transmission formats (NM-Data, NM-Vote and the combined NM-Data/Vote format) are recognized by the FlexRay NM.

Rationale: As the FlexRay NM of every node must be capable to receive and process only the NM-messages which are on the FlexRay bus (FRNM058) it must be configurable which formats are recognized by the FlexRay NM in order to avoid the overhead of “unused” formats.

FRNM148: Every FlexRay NM node shall be independently configurable to either send the NM-Vote in frames of the static or the dynamic segment of the FlexRay bus schedule.

FRNM169: Every FlexRay NM node shall be independently configurable to use the FlexRay NM HW support for receptions of NM-Votes which are transmitted in the static segment.

Note: To use this feature the “other” nodes have to send their NM-Vote at the start of designated FlexRay frames – see also FRNM165.

FRNM150: For a node sending NM-Votes in the dynamic segment it shall be configurable in which FlexRay cycles the node sends the NM-Vote.

FRNM151: Every FlexRay NM node shall be independently configurable to either send the NM-Data in a static slot or in a dynamic slot.

Note: Although it is possible to use multiple FlexRay slots to transmit NM-Data, only one slot should be used in order to limit the number of FlexRay buffers needed for the reception of the NM-Data from different nodes.

7.6.2 FlexRay NM-PDU format

As specified in FRNM147 the FlexRay NM is capable to send NM-Vote and NM-Data independently. Therefore several corresponding PDU formats exist for the NM-Vote and for the NM-Data. To also support an associated transmission of NM-Vote and NM-Data in the static segment the NM-Data PDU contains an optional Voting Bit.

7.6.2.1 FlexRay NM-Data PDU format

FRNM006: FlexRay NM-Data PDU format shall be defined as follows:

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

Table 7-1 FlexRay NM-Data PDU Format

FRNM076: Support of Control Bit Vector shall be configurable by means of the **FRNM_CONTROL_BIT_VECTOR_ENABLED** switch.

FRNM222: Support of Source Node Identifier shall be configurable by means of the **FRNM_SOURCE_NODE_IDENTIFIER_ENABLED** switch

FRNM154: The length of the NM Data PDU shall be configurable to any integer value between (and including) 0 and 8 by means of **FRNM_PDU_LENGTH**

FRNM227: The FlexRay NM shall not send an NM-Data PDU if configured not to do so by the means of the **FRNM_NM_DATA_DISABLED** switch.

FRNM155: The difference between applied standardized bytes and NM Data PDU length shall be user data.

FRNM156: The NM-Data PDU Control Bit Vector format shall be defined as follows:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte	Not available	Res	Res	Res	Res	Res	Res	RptMsg Request

Table 7-2 Control Bit Vector Format

FRNM055: The Control Bit Vector shall contain a Repeat Message Request Bit (RptMsgRequest) with the following meaning:
0: Repeat Message State not requested
1: Repeat Message State requested

FRNM213: Support of the Repeat Message Bit shall be configurable by means of the **FRNM_REPEAT_MESSAGE_BIT_ENABLED** switch (configuration parameter)

FRNM157: Bit 7 of the Byte containing the Control Bit Vector shall not be used.

Rationale: This bit is used for the NM-Vote when an NM-Data message is sent in the static segment of the FlexRay together with an NM-Vote.

FRNM214: Bit 7 of the Byte containing the Control Bit Vector shall be set 0_b.

Rationale: For processing purpose Bit 7 must be set to a value. The value of 0_b has been chosen as the NM-Vote mechanism uses an OR algorithm where 0_b does not influence the result.

FRNM161: Bit 1 to 6 of the Control Bit Vector shall be reserved for future extension and shall be set to 0_b.

7.6.2.2 FlexRay NM-Vote PDU format

FRNM215: The NM-Vote PDU format shall be defined as follows:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	Vote	Not available						

Table 7-3 NM-Vote PDU Format

FRNM216: The NM-Vote PDU format shall contain a Voting Bit (Vote) with the following meaning:
0: vote against keeping awake
1: vote for keeping awake

FRNM218: Bits 0-6 of the NM-Vote PDU shall not be used.

Rationale: Bits 0-6 are used for the Control Bit Vector of the NM-Data PDU when an NM-Data message is sent in the static segment of the FlexRay together with an NM-Vote.

FRNM219: Bits 0-6 of the NM-Vote PDU shall be set 0_b.

Rationale: For processing purposes Bits 0-6 must be set to a value. The value of 0_b has been chosen because it does not influence the Control Bit Vector when an OR algorithm is used to overlay the NM-Vote with the Control Bit Vector of the NM-Data PDU.

FRNM165: The NM-Vote PDU shall be placed at the beginning of the payload of the FlexRay frame if the NM-Vote is transmitted in the static segment of the FlexRay Schedule.

Rationale: To use the FlexRay NM Vector hardware support the NM-Vote PDU has to be placed at the start of the FlexRay frame. Regardless of whether a given node uses the FlexRay NM-vector hardware support, the node has to place its NM-Vote at this position to support the use of the hardware support by other nodes.

7.6.2.3 Combination of NM-PDUs

When the NM-Vote and NM-Data are combined within one PDU (see chapter 7.9 on page 37) the content of the NM-Vote will be combined with the content of the Control Bit Vector (CBV) Byte of the NM-Data as shown in Table 7-4 below. The following requirements specify this combination.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
NM-Data PDU - CBV	Not available	Res	Res	Res	Res	Res	Res	RptMsg Request
NM-Vote PDU	Vote	Not available						
Combined CBV and Vote	Vote	Res	Res	Res	Res	Res	Res	RptMsg Request

Table 7-4 Combined NM-Vote and NM-Data CBV Format

FRNM162: The NM-Vote shall be combined with the Control Bit Vector of the NM-Data PDU in case the NM-Vote is transmitted in the same PDU as the NM-Data in the static segment of the FlexRay schedule.

FRNM163: The PDU containing the NM-Vote shall be placed at the start of the FlexRay frame if the NM-Vote is transmitted in the static segment of the FlexRay schedule.

Rationale: To use the FlexRay NM Vector hardware support the NM Vector has to be placed at the start of the FlexRay frame. Regardless of whether a given node uses the FlexRay NM-vector hardware support, the node has to place its NM-Vote at this position to support the use of the hardware support by other nodes.

7.6.3 FlexRay NM-PDU transmission

For the FlexRay NM-PDU transmission both decoupled or immediate buffer access can be used. For more details see FlexRay Interface SWS [5].

7.6.4 FlexRay NM-PDU reception

For the FlexRay NM-PDU reception the FlexRay Reception Indication is used. For more details see FlexRay Interface SWS [5].

7.6.5 Functional requirements on FrNm API

The following requirements define the available FlexRay NM services.

FRNM037: The Source Node Identifier shall be set with the configuration parameter **FRNM_NODE_ID**.

FRNM170: Support of node detection shall be configurable by means of the **FRNM_NODE_DETECTION_ENABLED** switch (configuration parameter).

FRNM046: If node detection is enabled the service **FrNm_GetLocalNodeIdentifier** shall provide the node identifier configured for the local host node (**FRNM_NODE_ID**)

FRNM047: If node detection is enabled the service **FrNm_GetNodeIdentifier** shall provide the node identifier out of the most recently received NM-message.

FRNM171: If node detection is enabled the service **FrNm_RequestNodeDetection** shall request the node detection on the FlexRay Bus NM nodes.

FRNM172: If node detection is enabled the service **FrNm_RepeatMessageRequest** shall request the node detection on the FlexRay Bus NM nodes.

7.6.6 Constraints on FrNm API

FRNM174: The FlexRay NM shall support the service **FrNm_RequestBusSynchronization** by returning **NM_E_OK** without executing any functionality.

Rationale: As the FlexRay Bus is a synchronous bus, the FlexRay NM cannot influence the bus synchronization. This functionality is handled by the FlexRay Controller and FlexRay Driver. Since this service is used by future extensions (e.g. Gateway) the service shall at least be available.

7.7 Execution

The FlexRay NM State machine and hence the NM-task execution has to be synchronized with the bus schedule. Since most of the FlexRay NM decisions and state changes have to be aligned to the end of the FlexRay Bus Cycle (specifically at the boundary between two NM Repetition Cycles) this point in time is optimal to trigger the task activation. As the relative time for a cycle may vary due to the FlexRay clock rate correction, and the FlexRay NM algorithm is dependent on the synchronisation to the FlexRay Bus, it is not recommended to use a CPU time service. Instead this can (for example) be achieved by using the FlexRay Time Services provided by the FlexRay Interface [6].

7.7.1 General requirements

FRNM225: The FlexRay NM coordination algorithm shall be processor independent, which means it shall not rely on any processor specific hardware support and thus shall be realizable on any processor architecture that is in the scope of AUTOSAR.

FRNM176: The FlexRay NM shall realize FlexRay NM functions for Receive, FlexRay Cycle End events and optional for Transmit.

Note: It is could be up to the implementation if there is only one NM-Task (which is activated by the **`FrNm_<Rx|Tx|CycleStart>Confirmation`** callbacks) or several NM-Tasks with specific functionality, but [2][BSW00432] requires the split of the functions.

Note: The FlexRay NM will realize a Transmit function only if the node is an active node, i.e, **`FRNM_PASSIVE_NODE_ENABLED`** (configuration parameter) is set to OFF.

FRNM007: The FlexRay NM shall be executed once a FlexRay communication cycle.

7.7.2 FlexRay NM-Task structure

The FlexRay NM-Task will hold the “automated” functionality of the FlexRay NM - as there are the periodic transmission of NM-Messages, the processing of the received of NM-Messages and periodic processing of the FlexRay NM state machine (at the boundary between two NM-Repetition Cycles).

FRNM010: If transmission of cyclic NM-messages is started the FlexRay Interface function **`FrIf_Transmit`** shall be called to transmit NM-Vote and NM-Data.

FRNM012: If an NM-message is successfully transmitted, then FlexRay NM-function **`FrNm_TxConfirmation`** shall be called by the FlexRay Interface.

FRNM011: If an NM-message is received then the FlexRay NM-function **`FrNm_RxIndication`** shall be called by the FlexRay Interface.

FRNM013: If **`FrNm_RxIndication`** is called then FlexRay NM shall handle the data from the NM-message.

Note: It is up to the implementation how the data from the NM-message is handled. It can be immediately processed (to reduced the memory consumption), or it can be stored to be available for a later processing (to reduce computing time). However, NM-Votes received in a given cycle must be processed before the node transmits its vote in the subsequent cycle.

7.7.3 FlexRay NM-Task execution

7.7.3.1 Synchronous FlexRay NM-Task execution

FRNM048: The FlexRay NM task shall be scheduled synchronously to the FlexRay communication cycle such that the execution occurs within a window starting at the time where all NM-Votes of a cycle are available and ending at the time where the frame data for the next NM-Vote PDU is needed.

Rationale: This is necessary because FlexRay NM state changes may influence whether the NM-Vote PDU should be transmitted in the subsequent cycle. Since state changes are influenced by the aggregated NM vote in a given cycle and the state change subsequently influences the NM-Vote, the task must execute in a time window bounded by the cycle end and the transmission slot for the NM-Vote PDU. (FRNM168).

Note: The FlexRay interface shall notify the NM when all votes are theoretical available. This can either be achieved by setting an absolute timer or the cycle start event of the FlexRay communication controller (via the FrIf).

7.7.3.2 Asynchronous FlexRay NM-Task execution

FRNM177: FlexRay NM shall not use asynchronous NM-Task activation

Note: An alternative solution is under investigation which guarantees transition into Bus Sleep Mode at the same point in time when FlexRay NM task is not properly synchronized with the cycle timing of the FlexRay bus (i.e. within the same FlexRay communication cycle) but no solution has been found yet.

7.8 Additional Features

7.8.1 Cluster size

FRNM179: The AUTOSAR FlexRay NM algorithm shall support up to 64 nodes per NM-Cluster.

Note: The AUTOSAR FlexRay NM algorithm can support an arbitrary number of nodes per NM-cluster (even more than the maximum of 64 nodes per FlexRay cluster). It is only a matter of configuration, since the upper limit is not fixed and depends on the trade off between response time, fault-tolerance and resulted bus load configured for the AUTOSAR FlexRay NM coordination algorithm.

7.8.2 Detection of Remote Sleep Indication (optional)

The “Remote Sleep Indication” signals a situation where a node detects that all other nodes are ready to sleep, but the node where the indication occurs is still keeping the bus awake.

FRNM180: Detection of remote sleep indication shall be statically configurable with use of the **`FRNM_REMOTE_SLEEP_INDICATION_ENABLED`** switch (configuration parameter).

Note: if a node is configured as Passive (FRNM187) the remote sleep indication shall not be used because the node does not vote so it is incapable of being the only node keeping the NM Cluster awake. Consequently the remote sleep indication simply cannot occur.

FRNM181: If no NM-messages with an indication to keep the bus awake are received in the Normal Operation State for a configurable amount of time determined by the **FRNM_REMOTE_SLEEP_IND_TIME** (configuration parameter), the NM shall notify the NM Interface that all other nodes in the cluster are ready to sleep (the 'Remote Sleep Indication') by calling **Nm_RemoteSleepIndication**.

FRNM185: The FlexRay NM service call **FrNm_CheckRemoteSleepIndication** shall provide the information about current status of Remote Sleep Indication (i.e. already detected or not).

FRNM186: The FlexRay NM shall reject a check of Remote Sleep Indication (**FrNm_CheckRemoteSleepIndication**) when not in Network Mode. The service shall not be executed and the rejection return value **NM_E_NOT_EXECUTED** shall be returned.

FRNM229: If Remote Sleep Indication has been previously detected and if an NM-message with an indication to keep the bus awake is received in the Normal Operation State again, the NM shall notify the NM Interface that some nodes in the cluster are not ready to sleep anymore (the 'Remote Sleep Cancellation') by calling **Nm_RemoteSleepCancellation**.

FRNM230: If Remote Sleep Indication has been previously detected and Repeat Message State is entered from Normal Operation State, the NM shall notify the NM Interface that some nodes in the cluster are not ready to sleep anymore (the 'Remote Sleep Cancellation') by calling **Nm_RemoteSleepCancellation**.

7.8.3 User data (optional)

FRNM077: Support of user data shall be configurable by means of the **FRNM_USER_DATA_ENABLED** switch.

FRNM043: If user data handling is enabled, the user data shall be set by the service call **FrNm_SetUserData**.

FRNM044: If user data handling is enabled, the service **FrNm_GetUserData** shall provide the User Data from the last received NM-Message.

7.8.4 Passive Node Configuration (optional)

Nodes that are configured "Passive" Mode participate in the cluster NM only in a passive way. They only receive NM-Votes, but do not transmit votes to keep the cluster awake. Such a passive node never changes to the Normal Operation State (in the State Machine). It would be a configuration error if the ComM would call the **Nm_NetworkRequest** for such a node.

FRNM187: Passive Node Configuration shall be statically configurable with use of the **FRNM_PASSIVE_NODE_ENABLED** switch (configuration parameter).

FRNM188: If Passive Node Configuration is enabled (FRNM187) the Remote Sleep Indication options (FRNM180) shall not be used.

Note: configuration parameter **`FRNM_REMOTE_SLEEP_INDICATION_ENABLED`** shall be set to false.

7.8.5 NM PDU Rx Indication (optional)

The PDU Rx Indication could be used by the upper layers to detect NM acitivity (reception of a NM-Vote, NM-Data or combined NM-Data/Vote PDU) on the FlexRay bus. However, since a lot of NM PDUs will be received, especially when using the static segment for PDU transmission, it is not recommended to use this service.

FRNM189: NM PDU Reception indication shall be statically configurable with the use of the **`FRNM_PDU_RX_INDICATION_ENABLED`** switch (configuration parameter).

FRNM190: If NM PDU Reception indication is enabled (FRNM189) the FlexRay NM shall call **`Nm_PduRxIndication`** callback service at the successful reception of an NM-PDU

7.8.6 State change notification (optional)

FRNM191: The optional state change notification service shall be statically configurable with the use of the **`FRNM_STATE_CHANGE_INDICATION_ENABLED`** switch (configuration parameter).

FRNM192: If the optional state change notification service is enabled (FRNM191) all changes of the AUTOSAR FlexRay NM states shall be notified to the upper layer by calling **`Nm_StateChangeNotification`**.

7.8.7 Dual Channel PDU support (optional)

As described in more detail in 7.9, the FlexRay NM shall support the send and transmit of PDU on both FlexRay channels (A and B). For the static segment this feature is supported by the FrIf, where for the dynamic segment this feature must be provided by the FlexRay NM itself. The following requirements describe the required functionality.

FRNM231: The dual channel PDU support of the FlexRay NM shall be statically configurable with the use of the **`FRNM_DUAL_CHANNEL_PDU_ENABLE`** switch (configuration parameter).

7.9 Schedule details

The following sections describe requirements for the scheduling of NM PDUs on the FlexRay bus - for both dynamic segment and static segment.

As mentioned in chapter 7.6 the FlexRay NM is configurable to transmit the NM-Vote and NM-Data in different PDUs. Therefore the FlexRay NM offers six possibilities for the transmission of NM-messages. They are enumerated below and summarized in the subsequent table.

1. NM-Vote and NM Data transmitted within one PDU in static segment. The NM-Vote has to be realized as separate bit within the PDU
2. NM-Vote and NM-Data transmitted within one PDU in dynamic segment. The presence (or non-presence) of the frame corresponds to the NM-Vote
3. NM-Vote and NM-Data are transmitted in the static segment in separate PDUs. This is not recommended -→ Alternative 1 shall be used instead.
4. NM-Vote transmitted in static and NM-Data transmitted in dynamic segment. The usage of the FlexRay Network Management Vector Hardware support is possible.
5. NM-Vote is transmitted in dynamic and NM-Data is transmitted in static segment. This is not recommended -→ Possibility 2 or 6 shall be used instead.
6. NM-Vote and NM-Data are transmitted in dynamic segment in separate PDUs.

		FlexRay Segment	
		Static	Dynamic
PDU Type	NM-Vote	3 and 4	5 and 6
	NM-Data	3 and 5	4 and 6
	NM-Vote and NM-Data	1	2

Table 7-5 Summary of FlexRay PDU combination alternatives

Although every node can be configured independently to one of the above six options it is beneficial to choose a “common” transmission alternative.

In addition to the above PDU transmission alternatives FlexRay offers two physical channels where data can be transmitted. Frames can be transmitted on Channel A, Channel B or on both Channels A and B. For the dynamic segment a transmission on both channels requires two transmission- and receive-buffers as the FlexRay Protocol does not support shared transmission on Channel A and B in dynamic slots. In this special case the FlexRay NM can be configured to support a double transmit and receive (see 7.8.7).

7.9.1 FlexRay NM Cycle requirements

This section defines the schedule specific requirements that are required for a reliable transmission of FlexRay NM-messages.

FRNM193: The FlexRay NM Voting Cycle (**FRNM_VOTING_CYCLE** configuration parameter) shall be defined as the number of cycles needed to transmit the NM-Vote of every node at least once.

Note: The value of the NM-Voting Cycle is typically determined by the number of cycles needed to transmit the votes of all “dynamic segment voter” nodes. For example, if only one slot in the dynamic segment is used, 3 nodes transmitting in the dynamic segment would require that the Voting Cycle is set to 4 – see also FRNM195, FRNM196 and
Figure 10-3 (on page 81).

FRNM194: The FlexRay NM Data Cycle (**FRNM_DATA_CYCLE** configuration parameter) shall be defined as the number of cycles needed to transmit the NM-Data of every node at least once.

Note: The value of the NM-Data Cycle is typically determined by the number of cycles needed to transmit the NM-Data of all nodes using the dynamic segment. For example, if only one slot in the dynamic segment is used and 5 nodes transmit their NM-Data in the dynamic segment, the NM-Data Cycle is set to 8 – see also FRNM195 and Figure 10-4 (on page 82).

FRNM195: The FlexRay NM schedule specific cycle configurations **FRNM_VOTING_CYCLE, FRNM_DATA_CYCLE** and **FRNM_REPETION_CYCLE** shall have a value of 1, 2, 4, 8, 16, 32 or 64.

Rationale: The limitation to the mentioned values is because FlexRay Cycle Multiplexing is used, which is only defined for these values.

FRNM196: The FlexRay NM Repetition Cycle (**FRNM_REPETION_CYCLE** configuration parameter) shall be an integer multiple (including 1) of the NM Voting Cycle (**FRNM_VOTING_CYCLE**)

Rationale: To improve the reliability of the FlexRay NM, a number of repetitions of the NM Voting Cycle can be used. This will increase the chance that a “keep-awake” vote is not missed (e.g. due to a transmission error).

See

Figure 10-3 **(on page 81).**

FRNM197: The cycle offset within the NM Voting Cycle (**FRNM_VOTING_CYCLE**) shall be uniquely defined for each node in the NM Cluster by **FRNM_VOTING_CYCLE_OFFSET** (configuration parameter).

Note: This is used for transmission of NM-Vote PDU in the dynamic segment. The offset will be used to define the cycle where the FlexRay controller will transmit the NM-Vote PDU in the defined dynamic slot (**FRNM_VOTE_SLOT** – FRNM002).

FRNM199: The cycle offset within the NM-Data Cycle (**FRNM_DATA_CYCLE**) shall be uniquely defined for each node in the NM Cluster by **FRNM_DATA_CYCLE_OFFSET** (configuration parameter).

Note: This is used for transmission of NM-Data PDU in the dynamic segment. The offset will be used to define the cycle where the FlexRay controller will transmit the NM-Data PDU in the defined dynamic slot (**FRNM_DATA_SLOT** – FRNM201).

7.9.2 NM-Message scheduled requirements

For NM-Messages sent in the dynamic segment of the FlexRay Schedule, it is highly recommended to choose the first slots in the dynamic segment for the following reasons:

- Bandwidth (if no NM-message is sent less bandwidth is consumed)
- Flexibility (different nodes can send in the same slot but in different cycles)
- Predictability (it is guaranteed that the first slot is transmitted in each cycle)
- Determinism (transmission in the first slot always occurs at the same point in time in reference to the communication cycle start).

FRNM002: If at least one node transmits its NM-Vote in the dynamic segment, the slot it occupies shall be reserved exclusively for NM Votes in all cycles of the schedule. (**FRNM_VOTE_SLOT, FRNM_VOTE_SLOT_NUMBER** configuration parameters).

FRNM158: The payload of each NM-Frame in the dynamic segment shall only contain NM PDUs.

Rationale: As the presence and absence of FlexRay Frame in the dynamic segment containing NM-Messages can be interpreted as a vote, no application data is allowed to be transferred within the same frame – as this would influence the vote or data transmission.

FRNM200: Any NM-Vote that is transmitted in the dynamic segment shall be scheduled to use the NM-Vote-Slot (see FRNM002)

FRNM201: If at least one node transmits its NM-Data in the dynamic segment, the slot it occupies shall be reserved exclusively for NM Data in all cycles of the schedule. (**FRNM_DATA_SLOT, FRNM_DATA_SLOT_NUMBER** configuration parameters).

FRNM202: NM-Data that is transmitted separately from the NM-Vote(not combined with a NM-Vote) in the dynamic segment shall be scheduled to use the NM-Data-Slot (see FRNM201)

FRNM203: NM-Data that is transmitted combined with the NM-Vote in the dynamic segment shall be scheduled to use the NM-Vote-Slot (see FRNM002).

FRNM003: For each NM-PDU of each node in the NM cluster a unique FlexRay communication cycle, slot and cycle repetition factor combination shall be reserved within the FlexRay Interface.

7.10 Transmission Error Handling

FRNM035: If periodic NM-message transmission is running and if no NM-message is successfully transmitted within the time interval of **`FRNM_MSG_TIMEOUT_TIME`** (configuration parameter), the FlexRay NM shall notify the Generic NM Interface that transmission failure has occurred by calling **`Nm_TransmissionTimeoutException`**.

FRNM223: If the FlexRay bus communication of a FlexRay NM channel has been shut down the corresponding FlexRay NM shall be shut down.

Rationale: FlexRay NM depends on the availability of the FlexRay communication (via. FrIf) – if the bus has been shut down FlexRay NM cannot react properly.

Note: The phrase “NM channel” must not be confused with the meaning of “FlexRay channel”. The former is a logical unit, where the second is a physical bus interfaces line. FlexRay offers two channels per Communication controller, which are NOT independent, and can therefore not be seen as independent “NM channels”.

FRNM224: If the FlexRay NM is shut down the initialization status shall be set to **`NM_UNINIT`**.

Since FlexRay communication is independent of FlexRay NM and cannot be directly “stopped” by FlexRay NM, FlexRay NM does not support command of this type:

FRNM060: The optional service call **`FrNm_StartTransmission`** shall not be supported.

FRNM061: The optional service call **`FrNm_StopTransmission`** shall not be supported.

FRNM146: The optional service call **`FrNm_TriggerTransmission`** shall not be supported

7.11 Error classification

FRNM049: Reporting of development errors shall be statically configurable with use of the **`FRNM_DEV_ERROR_DETECT`** switch (configuration parameter).

FRNM056: Development errors shall not be returned by API functions. In case of a development error the corresponding API function shall return **`NM_E_NOT_OK`**, if applicable.

FRNM022: Development errors shall be reported to the Development Error Tracer.

FRNM057: Production errors shall not be returned by API functions. In case of a production error the corresponding API function shall return **NM_E_NOT_OK**, if applicable.

FRNM023: Production errors shall be reported to the Diagnostic Event Manager.

FRNM021: The following errors shall be detectable by the FlexRay NM depending on its build version (development/production mode).

Type or error	*Relevance*	*Related error code*	*Value*
API service used without module initialization	Development	FRNM_E_NO_INIT	01h
API service called with invalid channel handle	Development	FRNM_E_INVALID_CHANNEL	02h

FRNM050: If not initialized, the FlexRay NM shall reject every API service other than **FrNm_Init**. The called function shall not be executed. Instead **FRNM_E_NO_INIT** shall be reported to the Development Error Tracer and it shall return **NM_E_NOT_INIT** to the calling function.

FRNM051: When the NM API service with an invalid channel handle is called, the corresponding function shall report **FRNM_E_INVALID_CHANNEL** error to the Development Error Tracer (the value of invalid channel handle shall be passed to DET as instance ID) and it shall return **NM_E_NOT_OK** to the calling function.

Note: The NM-channel handle is invalid if it is different from the allowed configured values.

Note: Currently no production errors are defined.

8 API specification

FRNM034: FlexRay NM API consists of services that are bus independent and can be called as required. Each service other than **`Nm_Init`** refers to one NM channel only.

Note: The phrase “NM channel” must not be confused with the meaning of “FlexRay channel”. The former is a logical unit, where the second is a physical bus interfaces line. FlexRay offers two channels per Communication controller, which are NOT independent, and can therefore not be seen as independent “NM channels”.

The following figure gives an overview of the available API services and interfaces.

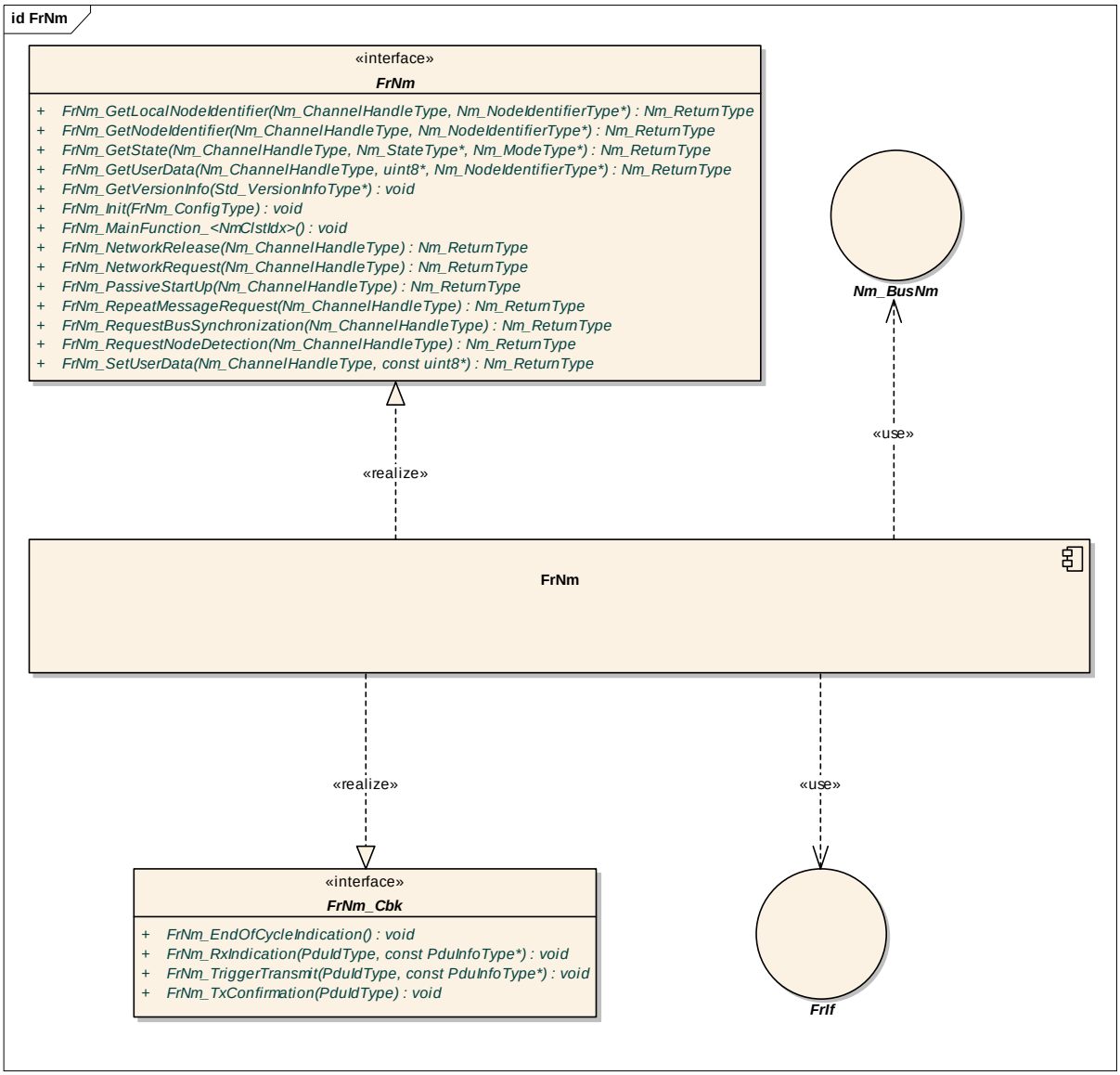

Figure 8-1 API Specification (Overview)

8.1 Imported types

In this chapter all types included from the following files are listed:

- Std_Types.h:
  - Std_ReturnType
  - Std_VersionInfoType
- Platform_Types.h:
  - uint8, uint16, uint32
  - suint8, sint16, sint32
  - uint8_least, uint16_least, uint32_least
  - sint8_least, sint16_least, sint32_least
  - float32, float64

8.2 Type Definitions

8.2.1 Generic NM Type Definitions

The FlexRay NM will use the type definitions as specified in Specification of Generic Network Management Interface [5] in chapter 8.2 Type definitions.

8.2.2 FlexRay NM specific Type Definitions

For **`FrNm_ConfigType`** see chapter 10.4 on page 70.

8.3 Function definitions: FrNm Services provided to upper layers

8.3.1 FrNm_Init

Service name:	FrNm_Init	
Syntax:	For all configuration Variants `void FrNm_Init ( FrNm_ConfigType * const nmConfigPtr );`	
Service ID:	`0x00`	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	`nmConfigPtr`	Pointer to the selected configuration set
	-	-
	-	-
Parameters (out):	None	N/A
Return value:	None	N/A
	-	-
Description:	Initializes the FlexRay NM and its internal state machine. (see FRNM028)	
Caveats:	This service function has to be called after the initialization of the corresponding bus interface.	

Configuration:	Mandatory

8.3.2 FrNm_PassiveStartUp

Service name:	FrNm_PassiveStartUp	
Syntax:	`Nm_ReturnType FrNm_PassiveStartUp ( const Nm_ChannelHandleType nmChannelHandle );`	
Service ID:	`0x01`	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	`None`	N/A
Return value:	`NM_E_OK`	No error
	`NM_E_NOT_OK`	Start of network management has failed
	`NM_E_NOT_EXECUTED`	Start of network management not allowed
Description:	This function initiates the Passive Startup of the FlexRay NM. It triggers the transition from Bus-Sleep Mode to the Network Mode in Repeat Message State (via the Synchronize State). This service has no effect if the current state is not Bus-Sleep Mode. In that case `NM_E_NOT_EXECUTED` is returned. (see FRNM138)	
Caveats:	FrNm must be initialized correctly.	
Configuration:	Mandatory	

8.3.3 FrNm_NetworkRequest

Service name:	FrNm_NetworkRequest	
Syntax:	`Nm_ReturnType FrNm_NetworkRequest ( const Nm_ChannelHandleType nmChannelHandle );`	
Service ID:	`0x02`	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	`None`	N/A
Return value:	`NM_E_OK`	No error
	`NM_E_NOT_OK`	Requesting of bus communication has failed
Description:	This function requests the network because the ECU needs to communicate on the bus. Network state shall be changed to 'requested'. (see FRNM144)	
Caveats:	FrNm must be initialized correctly.	
Configuration:	Optional – only available if **`FRNM_PASSIVE_NODE_ENABLED`** is set to OFF	

8.3.4 FrNm_NetworkRelease

Service name:	FrNm_NetworkRelease	
Syntax:	`Nm_ReturnType FrNm_NetworkRelease ( const Nm_ChannelHandleType nmChannelHandle );`	
Service ID:	`0x03`	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	`None`	N/A
Return value:	`NM_E_OK`	No error
	`NM_E_NOT_OK`	Releasing of bus communication has failed
Description:	This function releases the network because the ECU doesn't have to communicate on the bus. Network state shall be changed to 'released'. (see FRNM145)	
Caveats:	FrNm must be initialized correctly.	
Configuration:	Optional – only available if **`FRNM_PASSIVE_NODE_ENABLED`** is set to OFF	

8.3.5 Nm_DisableCommunication

Not supported by FlexRay NM.

8.3.6 Nm_EnableCommunication

Not supported by FlexRay NM.

8.3.7 FrNm_SetUserData

Service name:	FrNm_SetUserData	
Syntax:	`Nm_ReturnType FrNm_SetUserData ( const Nm_ChannelHandleType nmChannelHandle, const uint8 * const nmUserDataPtr );`	
Service ID:	`0x06`	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
	`nmUserData`	User data for the next transmitted NM message
	-	-
Parameters (out):	`None`	N/A
Return value:	`NM_E_OK`	No error
	`NM_E_NOT_OK`	Setting of user data has failed

Description:	This function sets user data for NM-Data transmitted next on the bus. (see FRNM043)
Caveats:	FrNm must be initialized correctly.
Configuration:	Optional (Only available if FRNM_USER_DATA_ENABLED is set to ON) (see FRNM077)

8.3.8 FrNm_GetUserData

Service name:	FrNm_GetUserData	
Syntax:	`Nm_ReturnType FrNm_GetUserData ( const Nm_ChannelHandleType nmChannelHandle, uint8 * const nmUserDataPtr );`	
Service ID:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	nmUserDataPtr	Pointer to the location where the user data from the last successfully received NM message shall be copied.
	-	-
Return value:	NM_E_OK	No error
	NM_E_NOT_OK	Getting of user data has failed
Description:	This function gets user data from the last successfully received NM message. (see FRNM044)	
Caveats:	FrNm must be initialized correctly.	
Configuration:	Optional (Only available if FRNM_USER_DATA_ENABLED is set to ON) (see FRNM077)	

8.3.9 Nm_GetPduData

Service name:	Nm_GetPduData	
Syntax:	`Nm_ReturnType FrNm_GetPduData ( const Nm_ChannelHandleType nmChannelHandle, uint8 * const nmPduData );`	
Service ID:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
	-	-
Parameters (out):	nmPduData	Pointer where NM PDU shall be copied to.
	-	-
	-	-
Return value:	NM_E_OK	No error
	NM_E_NOT_OK	Getting of NM PDU data has failed

Description:	Get the whole PDU data out of the most recently received NM message.
Caveats:	FrNm must be initialized correctly
Configuration:	Optional (Only available if `FRNM_NODE_DETECTION_ENABLED` is set to ON).

8.3.10 FrNm_RepeatMessageRequest

Service name:	FrNm_RepeatMessageRequest	
Syntax:	`Nm_ReturnType FrNm_RepeatMessageRequest ( const Nm_ChannelHandleType nmChannelHandle );`	
Service ID:	`0x09`	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	`None`	N/A
Return value:	`NM_E_OK`	No error
	`NM_E_NOT_OK`	Repeat Message Request has failed
Description:	This function causes a Repeat Message Request to be transmitted next on the bus. (see FRNM172)	
Caveats:	FrNm must be initialized correctly.	
Configuration:	Optional (Only available if `FRNM_NODE_DETECTION_ENABLED` is set to ON) (see FRNM170)	

8.3.11 FrNm_GetNodeIdentifier

Service name:	FrNm_GetNodeIdentifier	
Syntax:	`Nm_ReturnType FrNm_GetNodeIdentifier ( const Nm_ChannelHandleType nmChannelHandle, Nm_NodeIdentifierType * const nmNodeIdPtr );`	
Service ID:	`0x0A`	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	`nmNodeIdPtr`	Pointer to the location where the node identifier from the last successfully received NM-message shall be copied.
Return value:	`NM_E_OK`	No error
	`NM_E_NOT_OK`	Getting of the node identifier out of the last received NM-message has failed
Description:	This function gets the node identifier from the last successfully received NM-message. (see FRNM047)	
Caveats:	FrNm must be initialized correctly.	
Configuration:	Optional (Only available if `FRNM_NODE_DETECTION_ENABLED` is set to ON)	

	(see FRNM170)

8.3.12 FrNm_GetLocalNodeIdentifier

Service name:	FrNm_GetLocalNodeIdentifier	
Syntax:	`Nm_ReturnType FrNm_GetLocalNodeIdentifier ( const Nm_ChannelHandleType nmChannelHandle, Nm_NodeIdentifierType* const nmNodeIdPtr );`	
Service ID:	`0x0B`	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	`nmNodeIdPtr`	Pointer the location where the node identifier of the local node shall be copied.
Return value:	`NM_E_OK`	No error
	`NM_E_NOT_OK`	Getting of the node identifier of the local node has failed
Description:	This function gets the node identifier configured for the local node. (see FRNM046)	
Caveats:	FrNm must be initialized correctly.	
Configuration:	Optional (Only available if `FRNM_NODE_DETECTION_ENABLED` is set to ON) (see FRNM170)	

8.3.13 FrNm_RequestBusSynchronization

Service name:	FrNm_RequestBusSynchronization	
Syntax:	`Nm_ReturnType FrNm_RequestBusSynchronization ( const Nm_ChannelHandleType mmChannelHandle );`	
Service ID:	`0xC0`	
Sync/Async:	Synchronous	
Reentrancy:	Non-Reentrant	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	`None`	N/A
Return value:	`NM_E_OK`	No error
	-	
	-	
Description:	This function has no functionality – the service is provided only to be compatible to future extensions and to be compatible to the CAN-NM interface.	
Caveats:	FrNm must be initialized correctly.	
Configuration:	Optional (Only available if `FRNM_BUS_SYNCHRONIZATION_ENABLED` is defined)	

8.3.14 FrNm_CheckRemoteSleepIndication

Service name:	Nm_CheckRemoteSleepIndication	
Syntax:	`Nm_ReturnType Nm_CheckRemoteSleepIndication ( const Nm_ChannelHandleType nmChannelHandle, boolean * const nmRemoteSleepIndPtr );`	
Service ID:	`0x0D`	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (but not for the same NM-Channel)	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
Parameters (out):	`NmRemoteSleepInd Ptr`	Pointer to the location where the check result of remote sleep indication shall be copied.
Return value:	`NM_E_OK`	No error
	`NM_E_NOT_OK`	Checking of remote sleep indication bits has failed
	`NM_E_NOT_EXECUTED`	Checking of remote sleep indication has not executed
Description:	This function checks if remote sleep indication has taken place or not.	
Caveats:	The FrNm and the Nm itself are initialized correctly.	
Configuration:	Optional (Only available if `FRNM_REMOTE_SLEEP_INDICATION_ENABLED` is set to ON)	

8.3.15 FrNm_GetState

Service name:	FrNm_GetState	
Syntax:	`Nm_ReturnType FrNm_GetState ( const Nm_ChannelHandleType nmChannelHandle, Nm_StateType * const nmStatePtr, Nm_ModeType * const nmModePtr );`	
Service ID:	`0x0E`	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`nmChannelHandle`	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	`nmStatePtr`	Pointer to the location where the state of the network management shall be copied.
	`nmModePtr`	Pointer to the location where the mode of the network management shall be copied.
Return value:	`NM_E_OK`	No error
	`NM_E_NOT_OK`	Getting of NM state has failed
Description:	This function returns the state and the mode of the network management.	
Caveats:	FrNm must be initialized correctly.	
Configuration:	Mandatory	

8.3.16 FrNm_GetVersionInfo

Service name:	FrNm_GetVersionInfo	
Syntax:	`void FrNm_GetVersionInfo ( Std_VersionInfoType * NmVerInfoPtr );`	
Service ID:	0x0F	
Sync/Async:	Synchronous	
Reentrancy:	Yes	
Parameters (in):	None	
Parameters (out):	NmVerInfoPtr	Pointer to the location where the version information of this module shall be copied.
Return value:	None	
Description:	This service returns the version information of this module. The version information includes: - Module Id - Vendor Id - Vendor specific version numbers (BSW00407). Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.	
Caveats:	--	
Configuration:	Optional (only available if FRNM_VERSION_INFO_API is set to ON)	

8.3.17 FrNm_RequestNodeDetection

Service name:	FrNm_RequestNodeDetection	
Syntax:	`Nm_ReturnType FrNm_RequestNodeDetection ( const Nm_ChannelHandleType nmChannelHandle );`	
Service ID:	0x10	
Sync/Async:	Synchronous	
Reentrancy:	Non-Reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
	-	-
	-	-
Parameters (out):	None	N/A
Return value:	NM_E_OK	No error
	NM_E_NOT_OK	Requesting of node detection has failed or it has been rejected
	NM_E_NOT_EXECUTED	Requesting of node detection is currently not allowed
Description:	This function requests the node detection of the NM. (see FRNM171)	
Caveats:	FrNm must be initialized correctly.	
Configuration:	Optional (Only available if FRNM_NODE_DETECTION_ENABLED is defined) (see FRNM170)	

8.4 Call-back notifications: NM callbacks provided to lower layers

8.4.1 FrNm_TxConfirmation

Service name:	FrNm_TxConfirmation	
Syntax:	`void FrNm_TxConfirmation ( PduIdType FrNmTxPduId )`	
Service ID:	0xE0	
Sync/Async:	Sync	
Reentrancy:	Yes	
Parameters (in):	-	-
	FrNnTxPduId	ID of FlexRay NM L-PDU that has been transmitted
	-	-
Parameters (out):	None	N/A
Return value:	None	N/A
	-	-
Description:	This function is called by the FlexRay Interface after a FlexRay NM L-PDU has been transmitted.	
Caveats:	The FrIf and FrNm are initialized correctly. This function might be called in an interrupt context.	
Configuration:	Mandatory	

8.4.2 FrNm_RxIndication

Service name:	FrNm_RxIndication	
Syntax:	`void FrNm_RxIndication ( PduIdType FrNmRxPduId, const unit8 * FrNmRxSduPtr )`	
Service ID:	0xE1	
Sync/Async:	Sync	
Reentrancy:	Yes	
Parameters (in):	-	-
	FrNmRxPduId	ID of FlexRay NM L-PDU that has been received
	FrNmRxSduPtr	Pointer to received FlexRay NM L-SDU data
Parameters (out):	None	N/A
Return value:	None	N/A
	-	-
Description:	This function is called by the FlexRay Interface after a FlexRay NM L-PDU has been received. It shall copy the received FlexRay NM L-PDU and store it locally with respect to the received FlexRay NM L-PDU ID.	
Caveats:	The FrIf and FrNm are initialized correctly. This function might be called in interrupt context.	
Configuration:	Mandatory	

8.4.3 FrNm_TriggerTransmit

Service name:	FrNm_TriggerTransmit	
Syntax:	`void FrNm_TriggerTransmit ( PduIdType FrNmTxPduId, uint8 * FrNmRxSduPtr )`	
Service ID:	0xE4	
Sync/Async:	Sync	
Reentrancy:	Yes	
Parameters (in):	-	-
	FrNmTxPduId	ID of FlexRay NM L-PDU that has been triggered
	FrNmRxSduPtr	Pointer to triggered FlexRay NM L-SDU data
Parameters (out):	None	N/A
Return value:	None	N/A
	-	-
Description:	This function is called by the FlexRay Interface when FlexRay NM L-PDU has to be transmitted. It shall copy the triggered FlexRay NM L-PDU with respect to the triggered FlexRay NM L-PDU ID.	
Caveats:	The FrIf and FrNm are initialized correctly. This function might be called in interrupt context.	
Configuration:	Mandatory	

8.4.4 FrNm_CycleStartIndication

Service name:	FrNm_CycleStartIndication	
Syntax:	`void FrNm_CycleStartIndication ( void )`	
Service ID:	0xE5	
Sync/Async:	Sync	
Reentrancy:	No	
Parameters (in):	-	-
	-	
	-	
Parameters (out):	None	N/A
Return value:	None	N/A
	-	-
Description:	This function is called by the FlexRay Interface when FlexRay Interface detects the Cycle Start event of the FlexRay communication controller. It is used to trigger the processing of NM-messages in case of an NM-Repetition Cycle and/or NM-Data Cycle. This call shall also signal that the FrIf is ready to provide access to the hardware aggregated vote of this cycle.	
Caveats:	The FrIf and FrNm are initialized correctly. This function might be called in interrupt context.	
Configuration:	Mandatory	

8.5 Scheduled functions: FrNm Services provided for operating system

8.5.1 FrNm_MainFunction_<NmClstIdx>

Service name:	FrNm_MainFunction_<NmClstIdx>	
Syntax:	`void FrNm_MainFunction_<NmClstIdx> ( void )`	
Service ID:	0xF0	
Sync/Async:	Synchronous	
Reentrancy:	Non-Reentrant	
Parameters (in):	None	N/A
	-	-
	-	-
Parameters (out):	None	N/A
Return value:	None	N/A
	-	-
Description:	Main function of FlexRay NM. There is one dedicated FlexRay NM Main Function for each NM cluster. The API names are therefore: • FrNm_MainFunction_0() for FlexRay NM cluster associated with FlexRay NM channel 0 • FrNm_MainFunction_1() for FlexRay NM cluster associated with FlexRay NM channel 1 • Etc.	
Caveats:	None	
Configuration:	Mandatory	

8.6 Expected interfaces: Services called by the FrNm

8.6.1 Mandatory Interfaces

This chapter presents interfaces required to fulfil mandatory NM functionality.

8.6.1.1 FrIf API

API function	*Module*	*Description*
`Std_ReturnType FrIf_Transmit ( PduIdType, PduInfoType * );`	FrIf	Transmit a PDU via FlexRay

8.6.1.2 Nm CBK

CBK function	Module	Description

`void` **`Nm_NetworkStartIndication`**
`(`
`  const Nm_ChannelHandleType`
`);`
	NM Interface	Notification that a NM-message has been received in the Bus-Sleep Mode. The callback function shall start the network management state machine.

`void` **`Nm_NetworkMode`**
`(`
`  const Nm_ChannelHandleType`
`);`
	NM Interface	Notification that the FlexRay network management has entered Network Mode. The callback function shall enable transmission of application messages.

`void` **`Nm_SynchronizeMode`**
`(`
`  const Nm_ChannelHandleType`
`);`
	NM Interface	Notification that the FlexRay network management has entered Synchronize Mode.

`void` **`Nm_BusSleepMode`**
`(`
`  const Nm_ChannelHandleType`
`);`
	NM Interface	Notification that the network management has entered Bus-Sleep Mode. This callback function should perform a transition of the hardware and transceiver to bus-sleep mode.

`void` **`Nm_RemoteSleepIndication`**
`(`
`  const Nm_ChannelHandleType`
`);`
	NM Interface	Notification that the network management has detected that all other nodes are ready to sleep. *Configuration parameter:* **`FRNM_REMOTE_SLEEP_INDICATION_ENABLED`**

`void`
`Nm_RemoteSleepCancellation`
`(`
`  const Nm_ChannelHandleType`
`);`
	NM Interface	Notification that the network management has detected that no more all other nodes are ready to sleep.
		Configuration parameter: **`FRNM_REMOTE_SLEEP_INDICATION_ENABLED`**

Note: The Generic NM Interface is currently seen as thin adaptation layer (e.g. implemented as c-macros) which will be used to interface to the ComM. See [4].

8.6.1.3 DEM CBK

CBK function	Module	Description

`Std_ReturnType` **`Dem_ReportErrorStatus`**
`(`
`  Dem_EventIdType  EventId,`
`  Dem_EventStatusType  EventStatus`
`)`
	Dem	Service for reporting Errors during start up and normal operation to the DEM.

8.6.2 Optional Interfaces

This chapter presents interfaces required to fulfil optional NM functionality.

8.6.2.1 DET API

CBK function	Module	Description
`void` **`Det_ReportError`** `(`	Det	Service for reporting of development errors in the development mode.

uint16 ModuleId, uint8 InstanceId, uint8 ApiId, uint8 ErrorId)		*Configuration parameter:* **NM_DEV_ERROR_DETECT**

8.6.2.2 Nm CBK

CBK function	*Module*	*Description*
void **Nm_ Nm_PduRxIndication** (const Nm_ChannelHandleType boolean Nm_VoteIndication boolean Nm_DataIndication);	NM Interface	Notification that a NM message (NM-Vote, NM-Data or both) has been received by the FlexRay NM.

8.7 Parameter check

FRNM024: If detection of development errors is enabled by **FRNM_DEV_ERROR_DETECT** (configuration parameter), then validity check of input parameters shall be made for FlexRay NM API services.

FRNM025: Parameter type check shall be made at compile time. If types do not fit, the compilation process shall be stopped and respective compilation warnings or errors shall be reported to the extent supported by the compiler.

FRNM026: Parameter value check (for parameters of the constant value) shall be made at configuration time. If the value is invalid, the configuration process shall be stopped and respective configuration error shall be reported.

FRNM027: Parameter value check (for parameters of the variable value) shall be made at execution time. If the value is invalid, execution of a service shall be rejected and respective execution error shall be reported.

8.8 Version check

FRNM074: FlexRay NM shall check at pre-compile time if the if version numbers of C- and H-files are identical.

8.9 UML State Chart Diagram

The Figure 8-2 shows an UML state diagram with respect to the API specification.

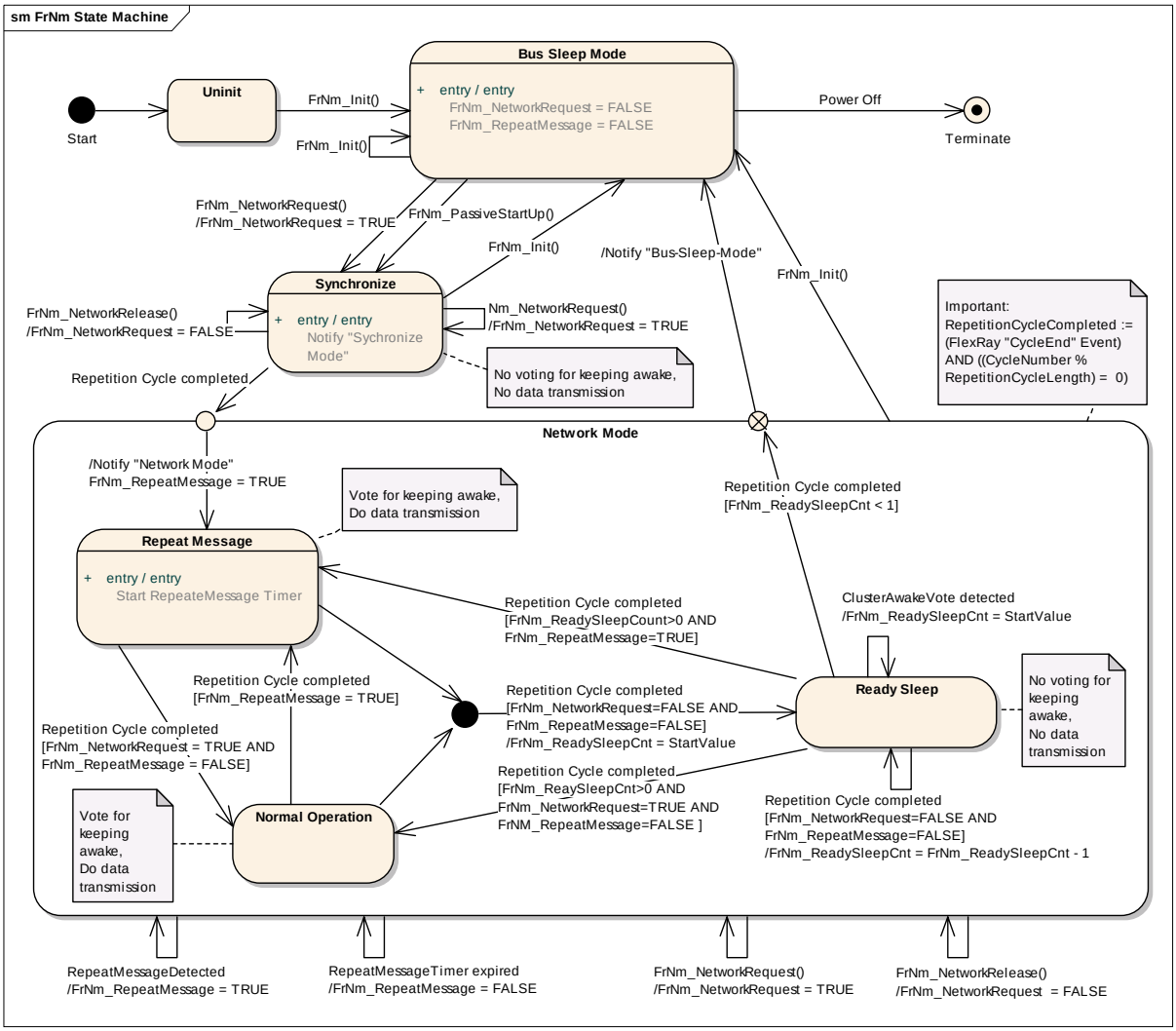

Figure 8-2 UML State Chart

9 Sequence diagrams

9.1 Use Case 01 – Initialization

Sequence in Figure 9-1 shows how to initialize the FlexRay Network Management.

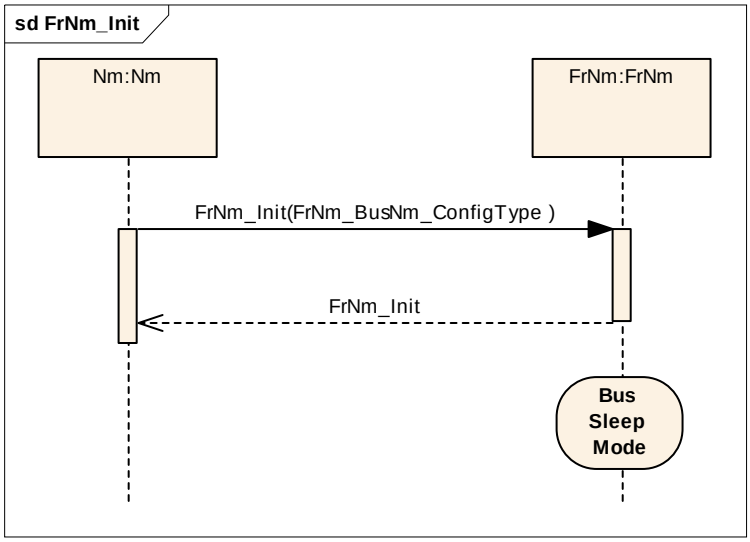

Figure 9-1 FrNm Init Sequence

9.2 Use Case 02 .- Passive Startup

Sequence in Figure 9-2 shows the normal passive startup.

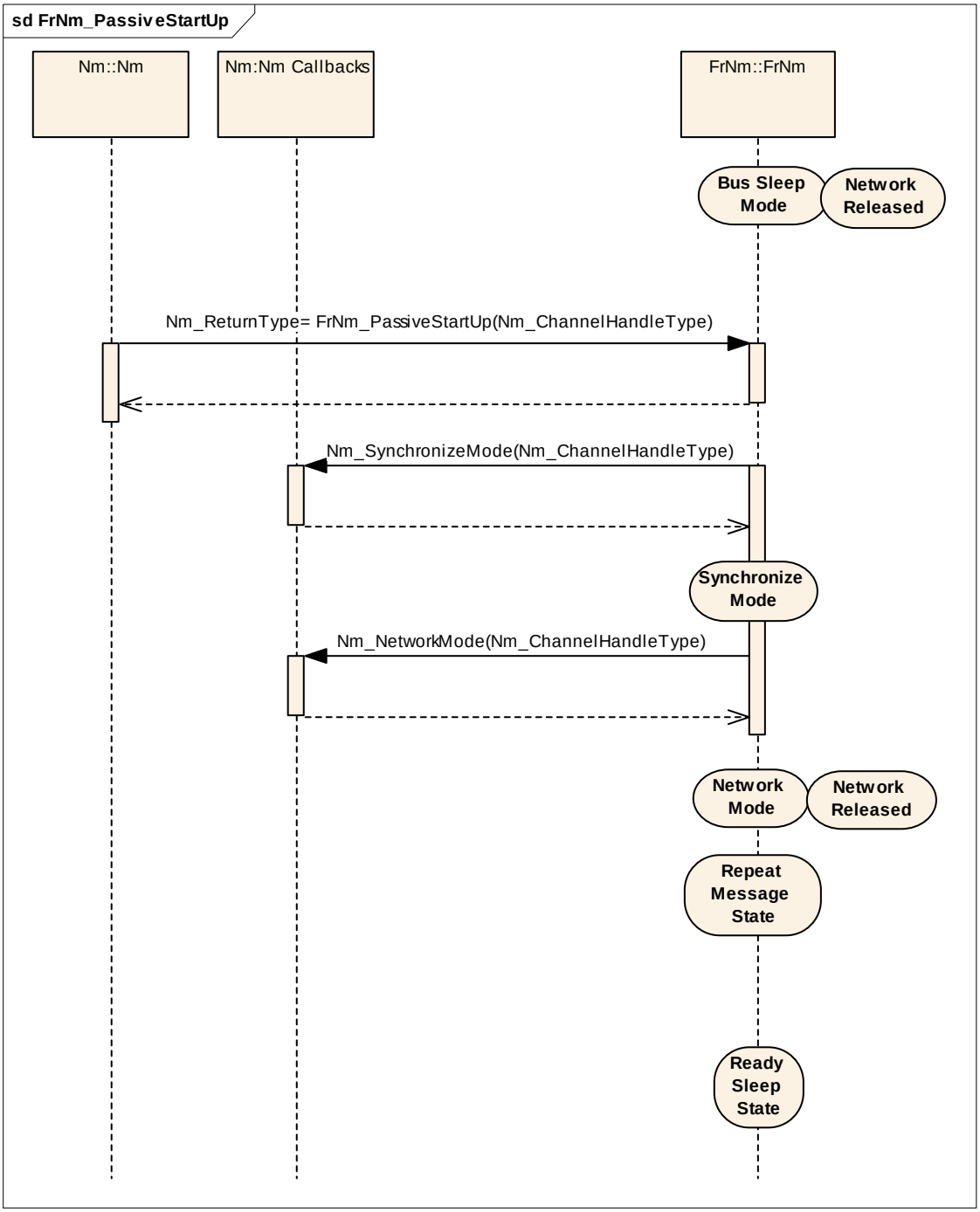

Figure 9-2 FrNm passive startup sequence

9.3 Use Case 03 – Passive Startup with a Network Request

Sequence in Figure 9-3 shows a passive startup where a network is requested before Network Mode has been reached.

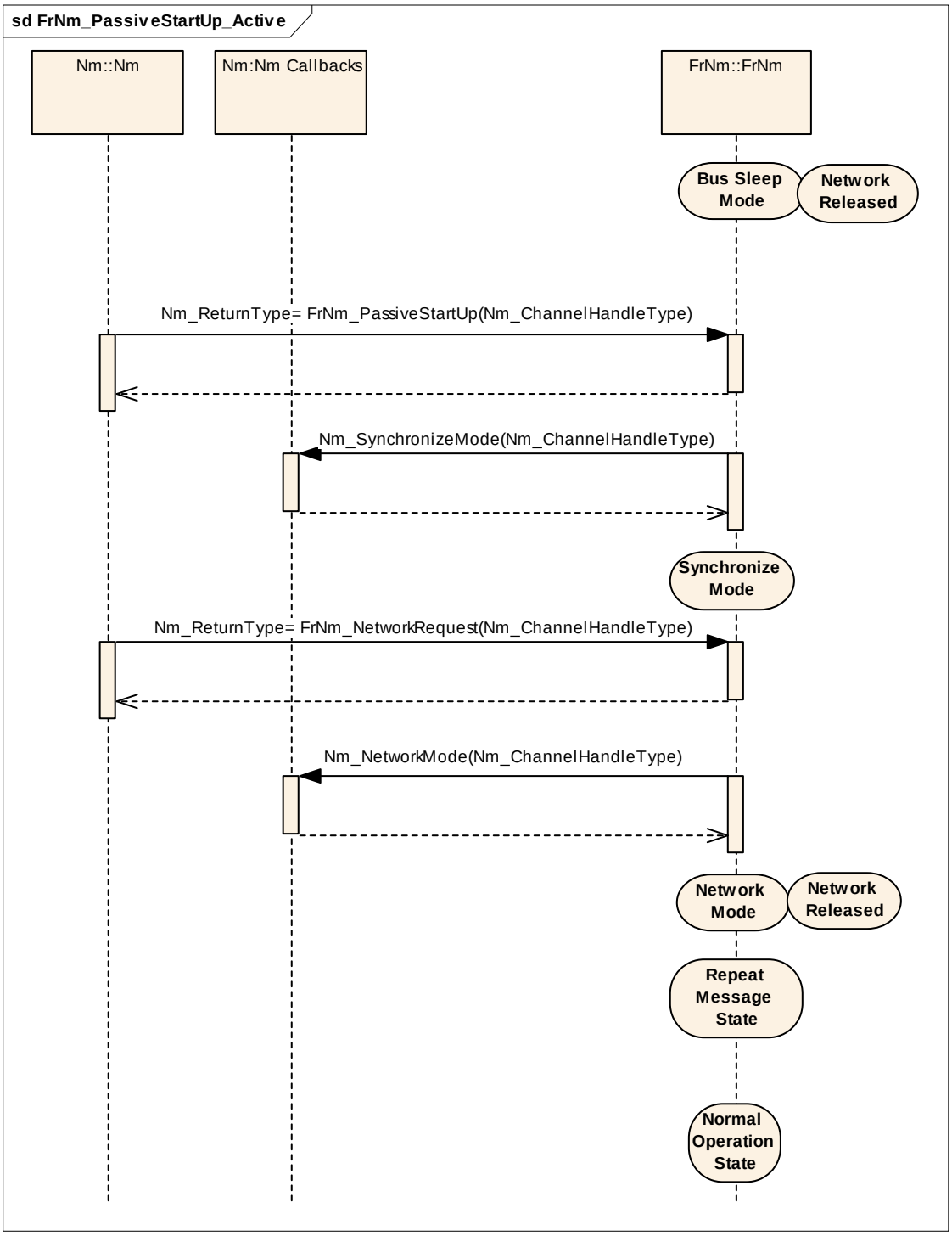

Figure 9-3 FrNm passive startup with a "active" network request sequence

9.4 Use Case 04 – Normal Operation

Sequence in Figure 9-4 shows how to request and release the network

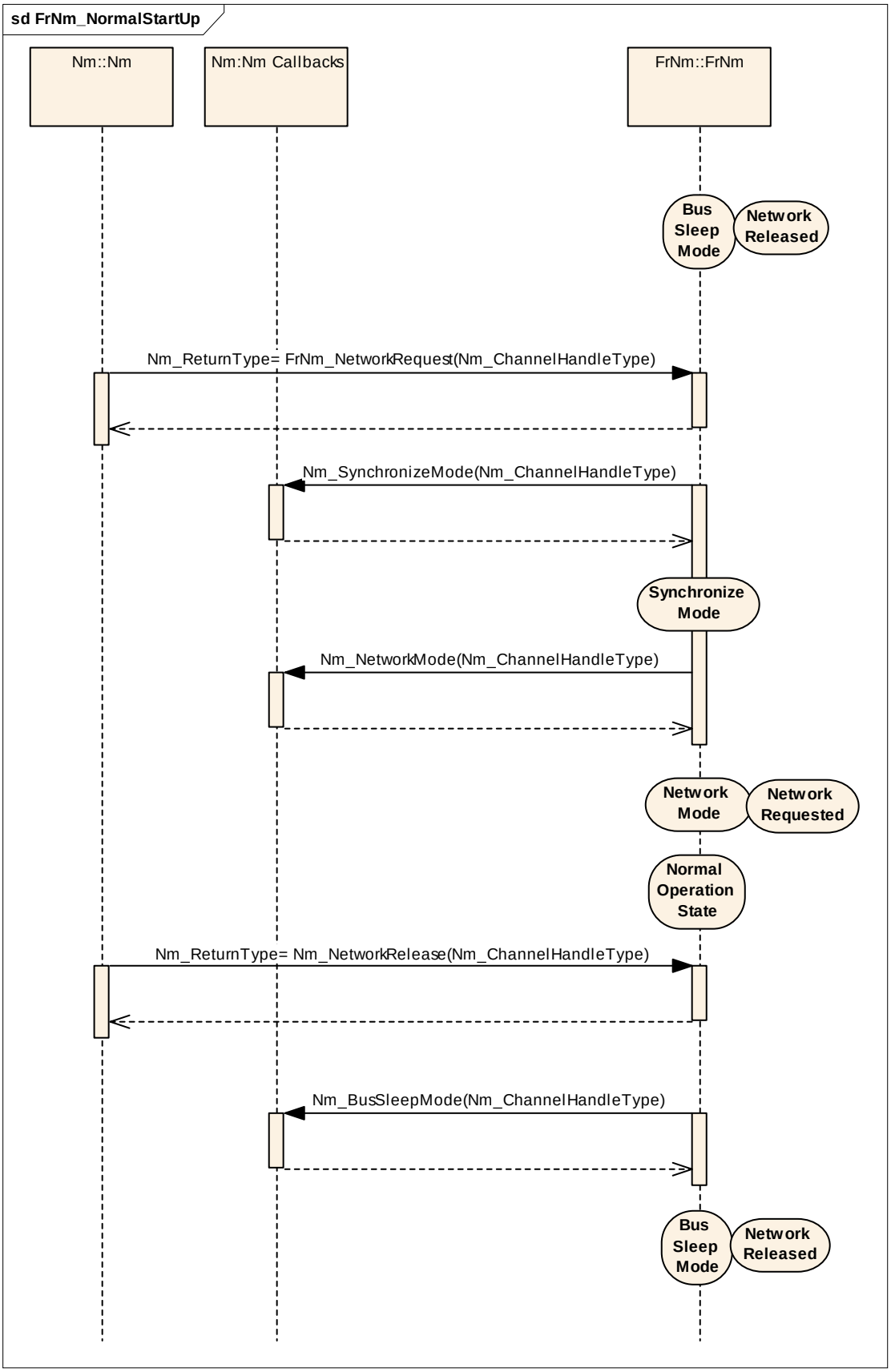

Figure 9-4 FrNm normal operation sequence

9.5 Use Case 05 – Shutdown

Sequence in Figure 9-5 shows a normal shutdown sequence.

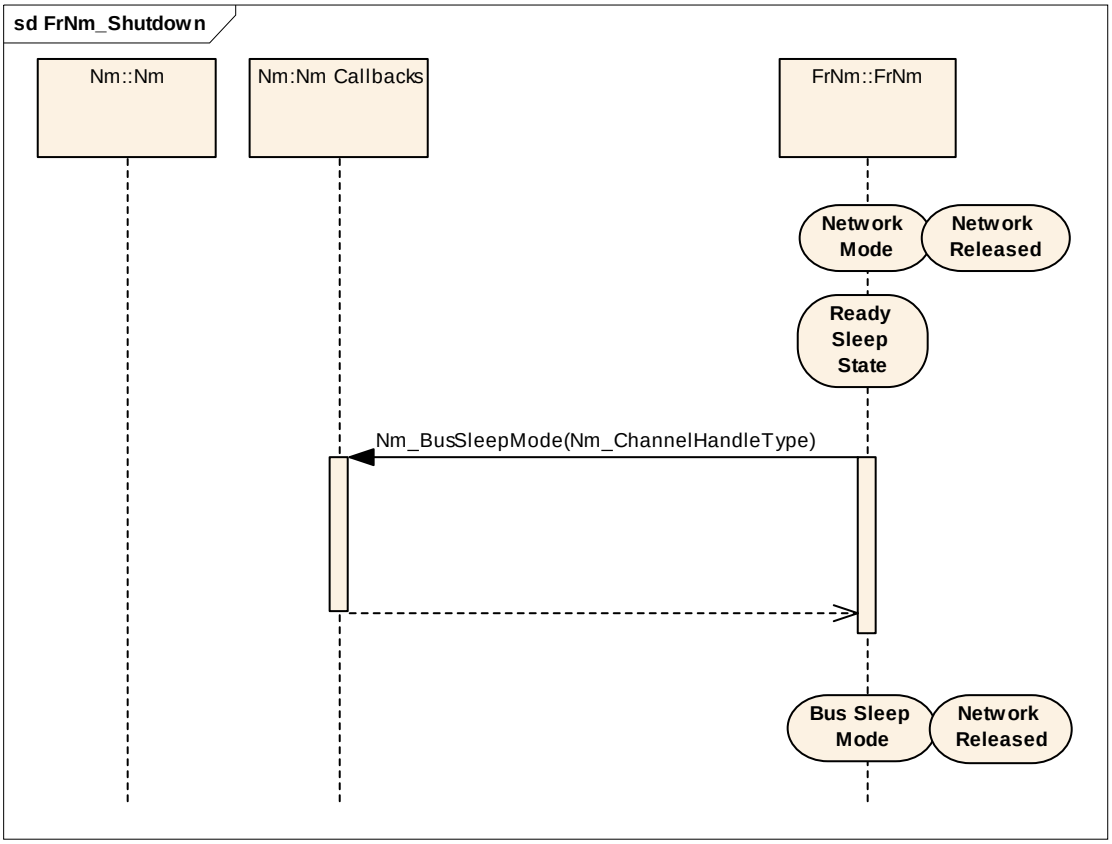

Figure 9-5 FrNm Shutdown

10 Configuration specification

The following chapter contains tables of all configuration parameters and switches used to determine the functional units of the AUTOSAR Generic Network Management. The default values of configuration parameters are denoted as bold.

FRNM020: Both static and runtime configuration parameters shall be located outside the source code of the module.

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) for the parameter specification. Chapter 10.2 specifies the structure (containers) and the parameters of the module FrNm. Chapter 10.3 specifies published information of the module FrNm.

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture
- AUTOSAR ECU Configuration Specification
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Variants

Variants describe sets of configuration parameters. E.g., variant 1: only pre-compile time configuration parameters; variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant a parameter can only be of one configuration class.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.1.4 Specification template for configuration parameters

The following tables consist of three sections:

- the general section
- the configuration parameter section
- the section of included/referenced containers

Pre-compile time - specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* - the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* - the configuration parameter shall be of configuration class *Post Build* and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

10.2 Variants

10.2.1 Variant 1: Pre-compile time

All configuration parameters are configurable at pre-compile time.

Use case: Source code optimizations

10.2.2 Variant 2: Pre-compile time of FrNm_GlobalConfig

All configuration parameters of the container **`FrNm_GlobalConfig`** related to enable or disable an optional feature shall be configurable at pre-compile time. The remaining configuration parameters shall be configurable at link time.

Use case: Object code libraries

10.2.3 Variant 3: Pre-compile time of FrNm_GlobalConfig; PDU configure on post build

The parameters contained in **`FrNm_PduConfig`** are configurable at post-build time. The parameters contained in **`FrNm_GlobalConfig`** are configurable at pre-compile time

Use case: ECU configuration can be flashed (L)

10.3 Global configurable parameters

FRNM017: The Global Scope specifies configuration parameters that shall be defined in the module's configuration header file **`FrNm_Cfg.h`**.

10.3.1 Global Scope

SWS Item	
Container Name	Global Scope - FrNm_GlobalConfig
Description	This container contains all configuration parameters of FlexRay NM configured from the NM Module perspective.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
Global Properties - FrNm_GlobalProperties	1	Module / None
Global Features - FrNm_GlobalFeatures	1	Module / None
Global Constants - FrNm_GlobalConstants	1	Module / None

10.3.1.1 Global Properties

SWS Item	
Container Name	Global Properties - FrNm_GlobalProperties
Description	This container contains module properties related to the FlexRay NM functionality.
Configuration Parameters	

Name	**`FRNM_DEV_ERROR_DETECT`**

Description	Pre-processor switch for enabling development error detection.(FRNM049)		
Type	`#define`		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**`FRNM_VERSION_INFO_API`**		
Description	Pre-processor switch for enabling version info API support.		
Type	`#define`		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None		

10.3.1.2 Global Features

SWS Item	
Container Name	Global Features - FrNm_GlobalFeatures
Description	This container contains module features related to the FlexRay NM functionality.
Configuration Parameters	

Name	**`FRNM_USER_DATA_ENABLED`**		
Description	Pre-processor switch for enabling user data support. (FRNM077)		
Type	`#define`		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**`FRNM_NODE_DETECTION_ENABLED`**		
Description	Pre-processor switch for enabling node detection support. (FRNM170)		
Type	`#define`		
Unit	--		
Range	ON	Enabled	

	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	This configuration is only available if the **FRNM_PASSIVE_NODE_ENABLED** configuration switch is set to OFF.		

Name	**FRNM_CONTROL_BIT_VECTOR_ENABLED**		
Description	Pre-processor switch for enabling control bit vector support.(FRNM076)		
Type	`#define`		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	This configuration is only available if the **FRNM_NODE_DETECTION_ENABLED** configuration switch is set to ON.		

Name	**FRNM_SOURCE_NODE_IDENTIFIER_ENABLED**		
Description	Pre-processor switch for enabling SourceNodeIdentifier support.(FRNM222)		
Type	`#define`		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**FRNM_PASSIVE_NODE_ENABLED**		
Description	Pre-processor switch for enabling Passive Node Configuration support.(FRNM187)		
Type	`#define`		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**FRNM_REMOTE_SLEEP_INDICATION_ENABLED**		
Description	Pre-processor switch for enabling remote sleep indication.(FRNM180)		
Type	`#define`		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	This configuration is only available if the **FRNM_PASSIVE_NODE_ENABLED** configuration switch is set to OFF.		

Name	**FRNM_PDU_RX_INDICATION_ENABLED**		
Description	Pre-processor switch for enabling PDU reception indication.(FRNM189)		
Type	#define		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**FRNM_REPEAT_MESSAGE_BIT_ENABLED**		
Description	Pre-processor switch for enabling the repeat message bit support (FRNM213)		
Type	#define		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	This configuration is only available if the **FRNM_CONTROL_BIT_VECTOR_ENABLED** configuration switch is set to ON.		

Name	**FRNM_STATE_CHANGE_INDICATION_ENABLED**		
Description	Pre-processor switch for enabling state change indication. (FRNM191)		
Type	#define		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**FRNM_VOTE_DYNAMIC_PDU_ENABLED**		
Description	Pre-processor switch for voting with dynamic PDU occurrence. (FRNM166)		
Type	#define		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**FRNM_BUS_SYNCHRONIZATION_ENABLED**		
Description	Pre-processor switch for enabling the bus synchronisation stub. (FRNM174)		
Type	#define		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	

	Post Build	--	
Scope	Module		
Dependency	None		

Name	**FRNM_NM_DATA_DISABLED**		
Description	Pre-processor switch for enabling the transmission of NM-Data. (FRNM227)		
Type	`#define`		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**FRNM_DUAL_CHANNEL_PDU_ENABLE**		
Description	Pre-processor switch for enabling the support of dual channel transmission and reception of NM-Messages. (FRNM231)		
Type	`#define`		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	x	Variant 1, 2, 3
	Link time	--	
	Post Build	--	
Scope	Module		
Dependency	None		

Included Containers			
Container Name	***Multiplicity***	***Scope***	***Dependency***
none			

10.3.1.3 Global Constants

SWS Item	
Container Name	Global Constants - FrNm_GlobalConstants
Description	This container contains module constants related to the Generic NM functionality.
Configuration Parameters	

Name	**FRNM_NUMBER_OF_CHANNELS**		
Description	Number of AUTOSAR NM channels allowed within one ECU.		
Type	`#define / const uint8`		
Unit	`--`		
Range	`1..255`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	--	
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None		

10.4 Channel configurable parameters

FRNM018: The Channel Scope specifies configuration parameters that shall be located in an external data structure of type **`FrNm_ChannelConfigType`**.

FRNM036: The following runtime configurable parameters shall be configurable for each channel separately.

10.4.1 Channel Scope

SWS Item	
Container Name	Channel Scope - FrNm_ChannelConfigType
Description	This container contains all configuration parameters of FlexRay NM configured from the channel perspective.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
Channel Identifiers	1	Instance / None
Channel Switches	1	Instance / None
Channel Timing	1	Instance / None

10.4.1.1 Channel Identifiers

SWS Item	
Container Name	Channel Identifiers
Description	This container contains instance specific identifiers related to the respective FlexRay Channel.
Configuration Parameters	

Name	**`FRNM_CHANNEL_ID`**		
Description	NM Channel identifier configured for the respective FlexRay Channel.		
Type	`#define / const uint8` Variant 1 `const uint8` Variant 2, 3		
Unit	`--`		
Range	`0..255`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2
	Post Build	L/M	Variant 3
Scope	Instance		
Dependency	It is used for referring to the respective NM channel handle. It must be		

	unique for each NM instance within one ECU.

Name	**FRNM_INSTANCE_ID**		
Description	FlexRay NM instance identifier configured for the respective FlexRay Channel.		
Type	`#define / const uint16 Variant 1` `const uint16 Variant 2, 3`		
Unit	`--`		
Range	`2000h..20FEh`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2
	Post Build	L/M	Variant 3
Scope	Instance		
Dependency	It is used for reporting of development errors to DET. It must be unique for each NM instance within one ECU.		

Name	**FRNM_TX_PDU_ID**		
Description	L-PDU identifier configured for the respective FlexRay Channel.		
Type	`#define / const PduIdType Variant 1` `const PduIdType Variant 2, 3`		
Unit	`--`		
Range	`0..255 / 0..65536`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2
	Post Build	L/M	Variant 3
Scope	Instance		
Dependency	It is used for referring to the FlexRay Interface transmit function. It must be consistent with the value configured in the FlexRay Interface. Remark: This ID is used for the combined transmission of NM-Vote and NM-Data or for the transmission of NM-Vote if NM-Data is sent in a separate PDU. See 7.9.		

Name	**FRNM_TX_PDU2_ID**		
Description	2nd L-PDU identifier configured for the respective FlexRay Channel.		
Type	`#define / const PduIdType Variant 1` `const PduIdType Variant 2, 3`		
Unit	`--`		
Range	`0..255 / 0..65536`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2
	Post Build	L/M	Variant 3
Scope	Instance		
Dependency	It is used for the optional second PDU for referring to the FlexRay Interface Transmit function (see **FRNM_TX_PDU_ID** and 7.9) and must be consistent with the value configured in the FlexRay Interface. Available only if **FRNM_DUAL_CHANNEL_PDU_ENABLE** is set to ON.		

Name	**FRNM_TX_DATA_PDU_ID**		
Description	L-PDU identifier configured for the respective FlexRay Channel.		
Type	`#define / const PduIdType Variant 1` `const PduIdType Variant 2, 3`		
Unit	`--`		
Range	`0..255 / 0..65536`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2
	Post Build	L/M	Variant 3
Scope	Instance		
Dependency	It is used if NM-Data is sent in a separate PDU, for referring to the FlexRay Interface transmit function for the transmission of NM-Data messages and		

	must be consistent with the value configured in the FlexRay Interface.

Name	**FRNM_TX_PDU2_ID**		
Description	2nd L-PDU identifier configured for the respective FlexRay Channel.		
Type	`#define / const PduIdType  Variant 1` `const PduIdType  Variant 2, 3`		
Unit	`--`		
Range	`0..255 / 0..65536`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2
	Post Build	L/M	Variant 3
Scope	Instance		
Dependency	It is used for the optional second PDU for referring to the FlexRay Interface Transmit function – see **FRNM_TX_PDU_ID** and 7.9. It must be consistent with the value configured in the FlexRay Interface. Available only if **FRNM_DUAL_CHANNEL_PDU_ENABLE** is set to ON.		

Name	**FRNM_NODE_ID**		
Description	NM node identifier configured for the respective FlexRay Channel.		
Type	`#define / const uint8  Variant 1` `const uint8  Variant 2, 3`		
Unit	`--`		
Range	`0..255`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2
	Post Build	L/M	Variant 3
Scope	Instance		
Dependency	It is used for identifying the respective NM node in the NM-cluster. It must be unique for each NM node within one NM cluster.		

Name	**FRNM_VOTE_SLOT**		
Description	Slot of FlexRay Schedule in the dynamic segment used for transmission of the NM-Vote. (FRNM002)		
Type	`#define / const uint16`		
Unit	`--`		
Range	`1..2047`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**FRNM_VOTE_SLOT_NUMBER**		
Description	Number of slots of FlexRay Schedule in the dynamic segment used for transmission of the NM-Vote. (FRNM002)		
Type	`#define / const uint8`		
Unit	`--`		
Range	`1..16`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**FRNM_DATA_SLOT**
Description	Slot of FlexRay Schedule in the dynamic segment used for transmission of the NM-Data. (FRNM201)
Type	`#define / const uint16`

Unit	`--`		
Range	`1..2047`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**`FRNM_DATA_SLOT_NUMBER`**		
Description	Number of slots of FlexRay Schedule in the dynamic segment used for transmission of the NM-Data. (FRNM201)		
Type	`#define / const uint8`		
Unit	`--`		
Range	`1..16`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**`FRNM_VOTING_CYCLE_OFFSET`**		
Description	Cycle offset within the NM Voting Cycle to transmit the ECUs NM-Vote PDU in the dynamic segment. (FRNM197)		
Type	`#define / const uint8`		
Unit	`--`		
Range	`0..63`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	--	
Scope	Module		
Dependency	**`FRNM_VOTING_CYCLE_OFFSET < FRNM_VOTING_CYCLE`**.		

Name	**`FRNM_DATA_CYCLE_OFFSET`**		
Description	Cycle offset within the NM Data Cycle to transmit the NM-Data PDU in the dynamic segment. (FRNM199)		
Type	`#define / const uint8`		
Unit	`--`		
Range	`0..63`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	--	
Scope	Module		
Dependency	**`FRNM_DATA_CYCLE_OFFSET < FRNM_DATA_CYCLE`**. Although it is advisable to choose either “0” as value.		

Name	**FRNM_CHANNEL_HANDLE**		
Description	Channel identifier configured for the respective instance of the NM.		
Type	`const uint8`		
Unit	--		
Range	0..*	--	
	--	--	
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2
	Post Build	M, L	Variant 3
Scope	instance		
Dependency	The `FRNNM_CHANNEL_HANDLE` shall be encoded in the `FrNmRxPduId` parameter which is passed to `FrNm_RxIndication()` function called by the FrIf. The `FRNM_CHANNEL_HANDLE` shall be encoded in the `FrNmTxPduId` which is passed to `FrNm_TxConfirmation()` function called by the FrIf.		

Name	**FRNM_PDU_LENGTH**		
Description	Length of the NM-Data PDU. (FRNM154)		
Type	`#define / const uint8`		
Unit	`--`		
Range	0…8		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	--	
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None		

10.4.1.2 Channel Switches

SWS Item	
Container Name	Channel Switches
Description	This container contains instance-specific switches related to the respective FlexRay NM PDU.
Configuration Parameters	

Name	**FRNM_CHANNEL_ACTIVE**		
Description	It determines if the respective NM channel is active or not.		
Type	`#define / const Nm_BooleanType Variant 1` `const Nm_BooleanType Variant 2, 3`		
Unit	--		
Range	NM_TRUE	Activated	
	NM_FALSE	Deactivated	
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2
	Post Build	L/M	Variant 3
Scope	Instance		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None		

10.4.1.3 Channel Timing

SWS Item	
Container Name	Channel Timing
Description	This container contains instance-specific timing related to the respective FlexRay Channel.
Configuration Parameters	

Name	**FRNM_DATA_CYCLE**		
Description	Number of FlexRay Schedule Cycles needed to transmit the NM-Data of all ECUs on the FlexRay Bus. (FRNM194)		
Type	`#define / const enum`		
Unit	`--`		
Range	`1, 2, 4, 8 ,16, 32, 64`	Note: The values are derived from the FlexRay requirements for cycle multiplexing ([1])	
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**FRNM_REPETITION_CYCLE**		
Description	Number of FlexRay Schedule Cycles used to repeat the transmission of the NM-Vote of all ECUs on the FlexRay Bus multiple times. (FRNM196)		
Type	`#define / const enum`		
Unit	`--`		
Range	`1, 2, 4, 8 ,16, 32, 64`	Note: The values are derived from the FlexRay requirements for cycle multiplexing ([1])	
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	--	
Scope	Module		
Dependency	**FRNM_REPETITION_CYCLE = n * FRNM_VOTING_CYCLE** where **n** = [1, 2, 4, 8, 16, 32, 64]		

Name	**FRNM_VOTING_CYCLE**		
Description	Number of FlexRay Schedule Cycles needed to transmit the NM-Vote of all ECUs on the FlexRay Bus. (FRNM193)		
Type	`#define / const enum`		
Unit	`--`		
Range	`1, 2, 4, 8 ,16, 32, 64`	Note: The values are derived from the FlexRay requirements for cycle multiplexing ([1])	
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	--	
Scope	Module		
Dependency	None		

Name	**FRNM_MSG_TIMEOUT_TIME**		
Description	Timeout of a NM-message [number of communication cycles]. It determines how long the NM shall wait with notification of transmission failure while communication errors occur on the bus. (FRNM035)		
Type	`#define / const uint16` Variant 1 `const uint16` Variant 2, 3		
Unit	`[number of communication cycles]`		
Range	`0..65535`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2
	Post Build	L/M	Variant 3
Scope	Instance		
Dependency	None		

Name	**FRNM_READY_SLEEP_CNT**		
Description	Numbers of repetitions in the ready sleep state before NM switches to bus sleep mode. (FRNM101)		
Type	`#define / const uint16`		
Unit	`--`		
Range	`1..65535`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	--	
Scope	Module		
Dependency	**FRNM_READY_SLEEP_CNT >= 1** On a value of “1”, the NM-State Machine will leave the Ready Sleep State after one NM Repetition Cycle with no “keep awake” votes.		

Name	**FRNM_REMOTE_SLEEP_IND_TIME**		
Description	Timeout for Remote Sleep Indication. It defines the time [number of communication cycles] how long it shall take to recognize that all other nodes are ready to sleep. (FRNM181)		
Type	`#define / const uint16` Variant 1 `const uint16` Variant 2, 3		
Unit	`[number of communication cycles]`		
Range	`0..65535`		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build	-	
Scope	Instance		
Dependency	**FRNM_REMOTE_SLEEP_IND_TIME** >= `FRNM_REPETITION_CYCLE`. The value 0 denotes that no Remote Sleep Indication functionality is configured.		

Name	**FRNM_REPEAT_MESSAGE_TIME**		
Description	Timeout for Repeat Message State. Defines the time [ms] how long the NM shall stay in the Repeat Message State.		
Type	`#define / const uint16 Variant 1` `const uint16 Variant 2, 3`		
Unit	[ms]		
Range	0..65535		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	x	Variant 2, 3
	Post Build		
Scope	Instance		
Dependency	The value 0 denotes that no Repeat Message State is configured, which means that Repeat Message State is transient and implies that it is left immediately after entry and consequently no startup stability is guaranteed and no node detection procedure is possible.		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None		

Note: Configuration of the enabled functionality for FlexRay NM shall be derived from the respective configuration of the Generic NM.

10.5 Published parameters

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

FRNM019: The following table specifies configuration parameters that shall be published in the module's description file.

SWS Item	FRNM019	
Information elements		
Information element name	***Type / Range***	***Information element description***
FRNM_VENDOR_ID	#define, uint16 / --	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
FRNM_MODULE_ID	#define, uint16 / --	Module ID of this module from Module List
FRNM_AR_MAJOR_VERSION	#define, uint16 / --	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
FRNM_AR_MINOR_VERSION	#define, uint16 / --	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
FRNM_AR_PATCH_VERSION	#define, uint16 / --	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
FRNM_SW_MAJOR_VERSION	#define, uint16 / --	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
FRNM_SW_MINOR_VERSION	#define, uint16 / --	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
FRNM_SW_PATCH_VERSION	#define, uint16 / --	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

10.6 Configuration constraints

FRNM068: The following configuration constraints are required for a proper work of FlexRay NM:

- NM_REPEAT_MESSAGE_TIME > FR_COMMUNICATION_CYCLE_TIME
- NM_READY_SLEEP_COUNT = m, m $\in$ N
- NM_TIMEOUT_TIME > FRNM_MSG_CYCLE_TIME
- FRNM_MSG_CYCLE_TIME = k * FR_COMMUNICATION_CYCLE_TIME , k $\in$ N

FRNM069: The following configuration constraints are conditionally recommended for FlexRay NM:

- NM_REPEAT_MESSAGE_TIME = 0 (conditions: (1) startup of all applications is completed as soon as the FlexRay communication is started and (2) node detection is not required in the FlexRay NM-cluster)
- NM_READY_SLEEP_COUNT = 1 (condition: bus communication is always shut down at the end of the NM Repetition Cycle in all nodes within the same FlexRay NM-cluster, even in presence of race conditions)

10.7 Examples

The following examples require FlexRay knowledge that is not described in the examples (e.g. the definition of a minislot). The FlexRay Communications System Specifications, V2.1 ([1]) contain the necessary information.

10.7.1 Example of Bus-Schedule with NM-Vote PDUs

Assume an example network of five nodes with the respective IDs of 1, 2, 3, 4 and 5 as shown in

Figure 10-1 (below).

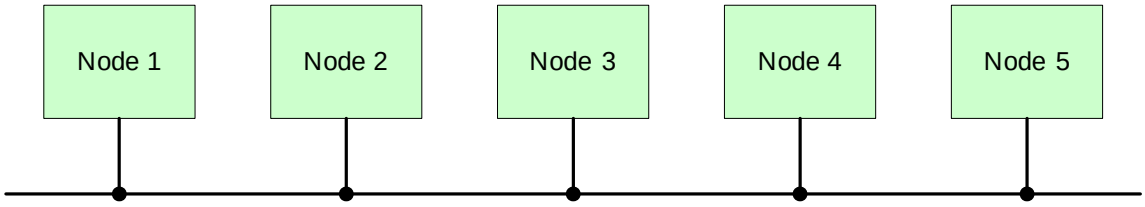

Figure 10-1 Example of five Node Network

The FlexRay Schedule allots 5 slots in the static segment and 6 slots in the dynamic segment as shown in

Figure 10-2 (on page 80). To keep the example simple the nodes are assigned static numerically equivalent to the node numbers, e.g., Node 1 is sending in static Slot 1, Node2 is sending in static slot 2 and so on.

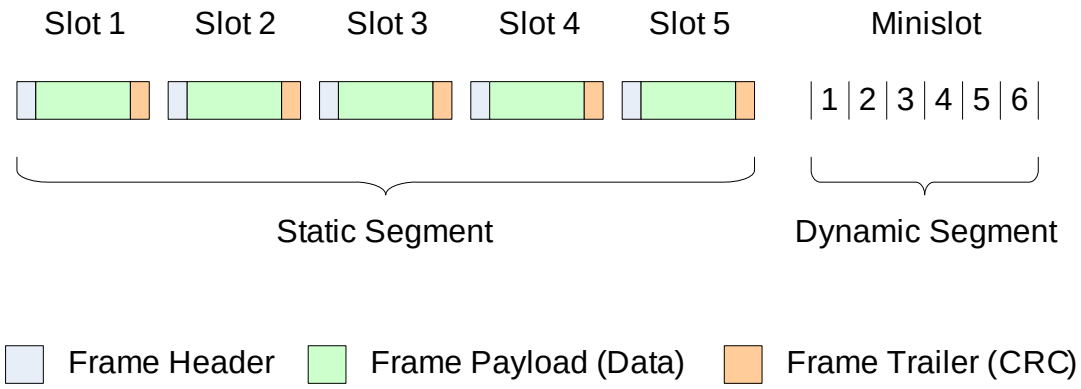

Figure 10-2 Example of Bus Schedule

Node 2 and 5 transmit their NM-Vote in the static segment, while the three remaining nodes 1, 3 and 4 transmit their NM-Vote in the dynamic segment as shown in

Figure 10-3 (on page 81).

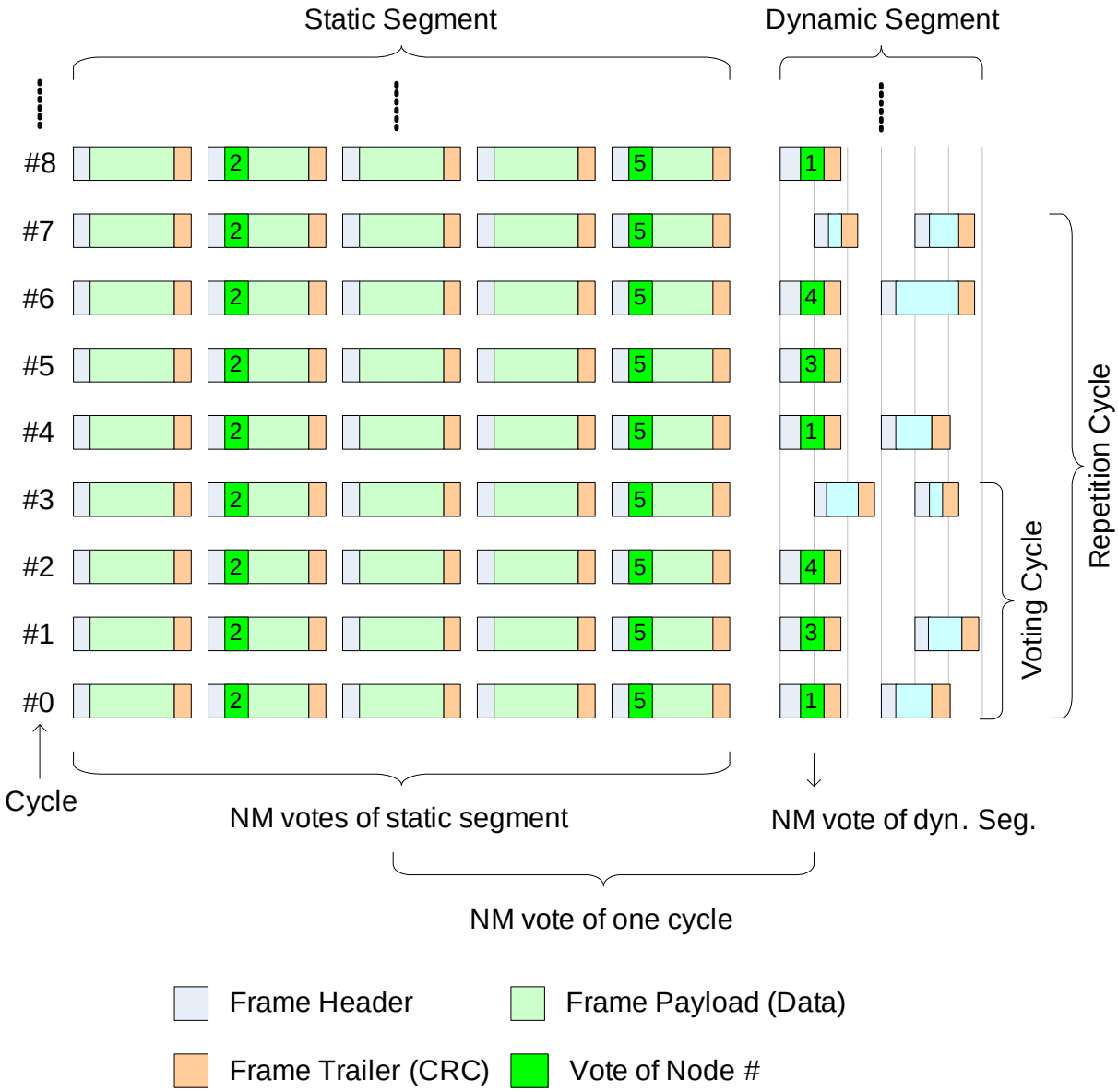

Figure 10-3 Example of NM-Vote in dynamic and static segment

As three dynamic voters exist, the Voting Cycle will be 4 and is repeated, therefore, every four cycles (cycle 0-1-2-3 and 4-5-6-7 and so on). Notice that no NM-Vote will be transmitted in last cycle of the Voting Cycle as all nodes have already voted.
In this example the Repetition Cycle is set to 2, which is twice the Voting cycle. Therefore every node will sent his vote twice.

10.7.2 Example of Bus Schedule with NM-Data PDUs

This example uses the same setup as in the previous example (10.7.1) - five nodes (Figure 10-1 on page 79), 5 slot in the static segment and 6 slots in the dynamic segment (Figure 10-2 on page 80).

Five Node need to send their NM-Data, which leads to a Data Cycle of “8”, as only 1,2,4,8,16,32 and 64 are allowed due the restriction of the FlexRay Cycle multiplexing (see FRNM195).

Figure 10-4 (below) shows that Node 1 will send its NM-Data in cycle 2, 10 and so on. Node 2, 3, 4 and 5 will behave in a similar way.
As the Data Cycle is “8”, all nodes will send their NM-Data only every 8 cycles.

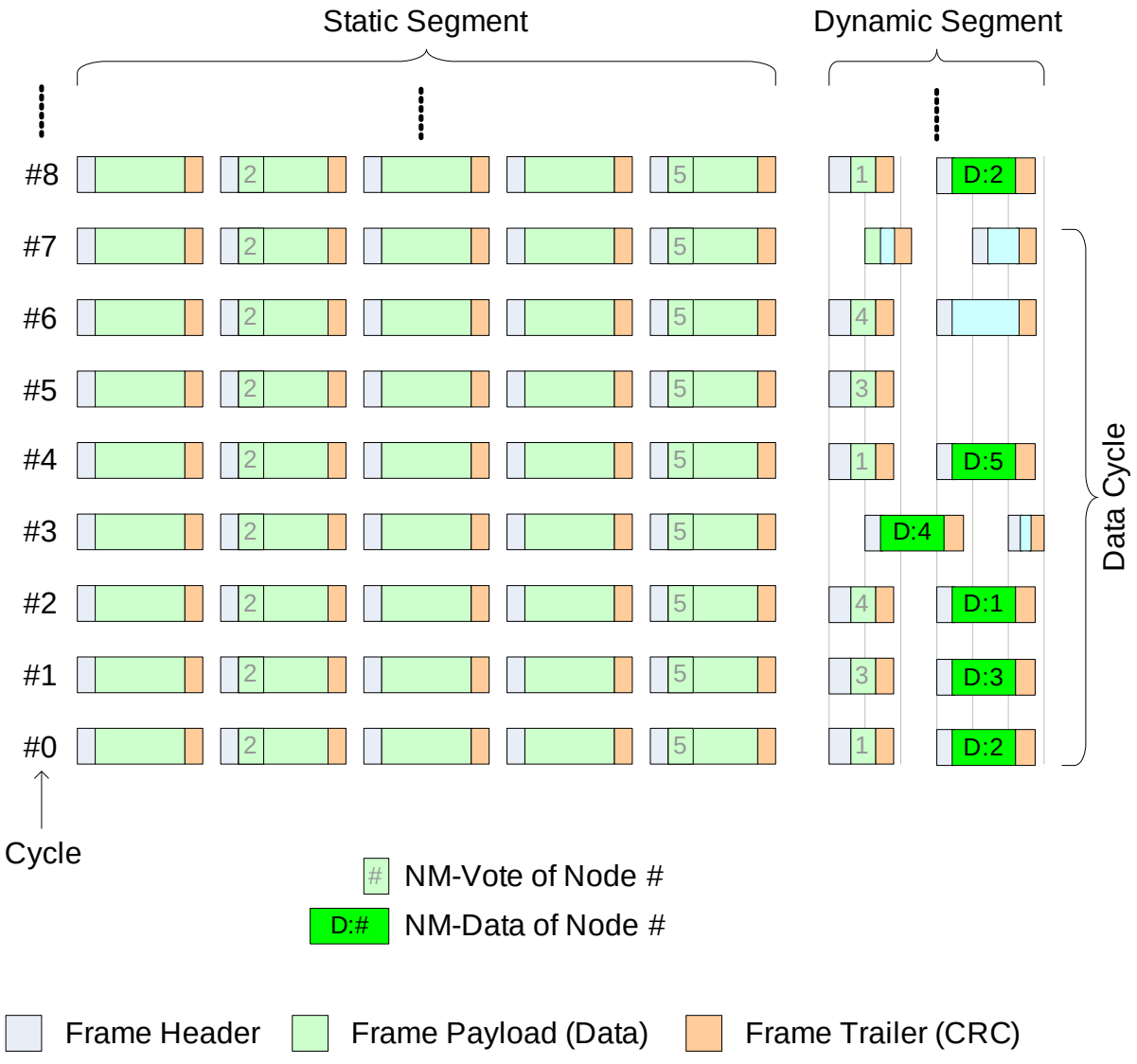

Figure 10-4 Example of Bus Schedule with NM-Data

11 Changes to Release 1

11.1 Deleted SWS Items

11.1.1 Changes till Release 1.5

SWS Item	*Rationale*
FRNM031	Requirement superfluous since bus load reduction functionality is not provided by the FlexRay NM.

11.1.2 Changes of Release 1.6

SWS Item	*Rationale*
FRNM004	Not longer supported by the Module Generic Interface [5]
FRNM005	Not longer supported by the Module Generic Interface [5])
FRNM014	due to passive node functionality
FRNM016	Not longer supported by the Module Generic Interface [5])
FRNM038	
FRNM039	
FRNM040	
FRNM041	
FRNM054	Not longer supported by the Module Generic Interface [5])
FRNM067	Not needed anymore as the FlexRay NM state machine is synchronous to the FlexRay schedule.
FRNM076	
FRNM078	
FRNM079	
FRNM080	
FRNM062	Not longer supported

11.1.3 Changes of Release 1.7

SWS Item	*Rationale*
FRNM159	Due to#15463: over-specification
FRNM149	Due to #15466: superfluous and in conflict with FRNM002

11.2 Replaced SWS Items

11.2.1 Changes of Release 1.6

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
FRNM003	FRNM002, FRNM200, FRNM201, FRNM202, FRNM203	Changed transmission scheme
FRNM008 FRNM009	FRNM007, FRNM048, FRNM176	Added support of static segment leads to changed task requirements

11.3 Changed SWS Items

11.3.1 Changes till Release 1.5

SWS Item	Rationale
FRNM002	Editorial change
FRNM003	Editorial change
FRNM004	Editorial change
FRNM005	Editorial change
FRNM010	Editorial change
FRNM011	Editorial change
FRNM012	Editorial change
FRNM013	Editorial change
FRNM015	Requirement rephrased regarding respective configuration of FrIf
FRNM016	Editorial change
FRNM017	SWS template adaptation – configuration specification
FRNM018	SWS template adaptation – configuration specification
FRNM021	Editorial change (“wrong” → “invalid”)
FRNM035	Called API function name adaptation
FRNM036	SWS template adaptation – configuration specification
FRNM038	Editorial change
FRNM039	Editorial change
FRNM042	Initialization refined
FRNM047	Configuration condition removed
FRNM045	Initialization refined
FRNM050	Editorial change
FRNM051	NM_E_NOT_OK as expected return value, invalid channel handle for instance identifier determination
FRNM052	Editorial change

11.3.2 Changes of Release 1.6

SWS Item	Rationale
FRNM006	
FRNM010	
FRNM013	Redefined to gain freedom for the implementation
FRNM014	
FRNM030	
FRNM042	
FRNM046	
FRNM047	
FRNM055	
FRNM058	
FRNM060	
FRNM061	
FRNM071	

11.3.3 Changes of Release 1.7

SWS Item	Rationale
FRNM116	Due to #15458: change to fit to configurable behaviour of passive nodes
FRNM015	Due to #15460: change to fit to hardware support for NM vector processing

FRNM058	Due to #15464: change to fit to hardware support for NM vector processing
FRNM147	Due to #15465: rephrased – exchanged “independently” with “separately”
FRNM154	Due to (#15467: typo

11.4 Added SWS Items

11.4.1 Changes till Release 1.5

SWS Item	*Rationale*
FRNM054	Requirement for Reception Indication of a Control Bit Vector added
FRNM055	Control Bit Vector format and coding extended
FRNM056	Do not return development errors by API functions
FRNM057	Do not return production errors by API functions
FRNM058	Requirement for processing of every NM message added.
FRNM059	Selectable post-built time configuration defined
FRNM060	Transmission timeout monitoring refined
FRNM061	Transmission timeout monitoring refined
FRNM062	Transmission timeout monitoring refined
FRNM064	File structure added
FRNM065	File structure added
FRNM066	File structure added
FRNM067	NM-Task and NM-Message inter-relation
FRNM068	Configuration constraints
FRNM069	Configuration constraints
FRNM070	SWS template adaptation – File structure added
FRNM071	Initialization refined
FRNM072	Initialization refined
FRNM073	Initialization refined
FRNM074	Versions check added
FRNM075	Platform independency added

11.4.2 Changes of Release 1.6

SWS Item	*Rationale*

Due the large number of new requirements, only a summary is listed in this chapter.

FRNM100, FRNM101, FRNM102, FRNM103, FRNM104, FRNM105, FRNM106, FRNM107, FRNM108, FRNM109, FRNM110, FRNM111, FRNM112, FRNM113, FRNM114, FRNM115, FRNM116, FRNM117, FRNM118, FRNM119, FRNM120, FRNM121, FRNM122, FRNM123, FRNM124, FRNM125, FRNM126, FRNM127, FRNM128, FRNM129, FRNM130, FRNM131, FRNM133, FRNM134, FRNM135, FRNM136, FRNM137, FRNM138, FRNM139, FRNM140, FRNM141, FRNM142, FRNM143, FRNM144, FRNM144, FRNM145, FRNM146, FRNM147, FRNM148, FRNM149, FRNM150, FRNM151, FRNM154, FRNM155, FRNM156, FRNM157, FRNM158, FRNM159, FRNM160, FRNM161, FRNM162, FRNM163, FRNM165, FRNM166, FRNM167, FRNM168, FRNM169, FRNM170, FRNM171, FRNM172, FRNM174, FRNM175, FRNM176, FRNM177, FRNM179, FRNM180, FRNM181, FRNM185, FRNM186, FRNM187, FRNM188, FRNM189, FRNM190, FRNM191, FRNM192, FRNM193, FRNM194, FRNM195, FRNM196, FRNM197, FRNM199, FRNM200, FRNM201, FRNM202, FRNM203, FRNM205, FRNM213, FRNM214, FRNM215, FRNM216, FRNM218, FRNM219, FRNM220, FRNM221, FRNM222, FRNM223, FRNM224, FRNM225

11.4.3 Changes of Release 1.7

SWS Item	*Rationale*
FRNM226	Due to #14949: Added setting of FrNm_RepeatMessage on service call
FRNM227	Due to #15467: Do not send NM-Data if no Data is configurated (len = 0)