Document Title	Specification of FlexRay Network Management
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	028
Document Classification	Standard

Document Version	3.1.0
Document Status	Final
Part of Release	3.1
Revision	5

Document Change History

Date	Version	Changed by	Change Description
20.09.2010	3.1.0	AUTOSAR Administration	• Updated FRNM021, FRNM305, FRNM316, FRNM317, FRNM256, FRNM257 • Added FRNM340, FRNM376, FRNM393, FRNM338, FRNM342, FRNM378, FRNM379, FRNM380, FRNM383, FRNM384, FRNM385, FRNM386, FRNM318, FRNM315 • Deleted FRNM318, FRNM306 • Legal disclaimer revised
02.02.2009	3.0.3	AUTOSAR Administration	Layout adaptations
30.01.2009	3.0.2	AUTOSAR Administration	• Incorporation of core partner change requests for R3.0 • Legal disclaimer revised
23.01.2008	3.0.1	AUTOSAR Administration	Updated chapter 8/10 after changes in BSW UML model

Document Change History

Date	Version	Changed by	Change Description
04.12.2007	3.0.0	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 • Document meta information extended • Small layout adaptations made • Added **FRNM311**
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

Disclaimer

Advice for users

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-Network	Instance of the FlexRay NM to handle one physical FlexRay Bus. Caution: The FlexRay Bus contains two channels which cannot be handled independent! Therefore the **NM-Network** covers both FlexRay bus channels. Equivalent to one NM-Cluster
NM Data Cycle	Number of FlexRay 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 FlexRay 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.
NM-Cluster	Obsolete, equivalent to NM-Network
ClusterAwake Vote	At least one Node other than itself votes for keeping the cluster awake.

3 Related documentation

1.1 Input documents

[1] AUTOSAR Layered Architecture,
AUTOSAR_LayeredSoftwareArchitecture.pdf

[2] AUTOSAR General Requirements on Basic Software Modules
AUTOSAR_SRS_General.pdf

[3] AUTOSAR Requirements on Basic Software Module NM
AUTOSAR_SRS_NM.pdf

1.2 Related standards and norms

[4] FlexRay Communications System Specifications, V2.1

1.3 Related AUTOSAR documents

[5] AUTOSAR Specification of Communication Manager
AUTOSAR_SWS_ComManager.pdf

[6] Specification of Generic Network Management Interface
AUTOSAR_SWS_NMInterface. pdf

[7] Specification of FlexRay Interface
SWS_FlexRayInterface. pdf

[8] Specification of ECU State Manager
AUTOSAR_SWS_EcuStateManager.pdf

[9] AUTOSAR Specification of Module Diagnostic Event Manager
AUTOSAR_SWS_DEM.pdf

[10] Specification of Development Error Tracer
AUTOSAR_APIspec_DevelopmentErrorTracer.pdf

[11] AUTOSAR Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[12] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes.pdf

[13] Specification of Compiler Abstraction
AUTOSAR_SWS_CompilerAbstraction.pdf

[14] Specification of Operation System

AUTOSAR_SWS_OS.pdf

[15] Specification of FlexRay State Manager
AUTOSAR_SWS_FlexRay_StateManager.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.

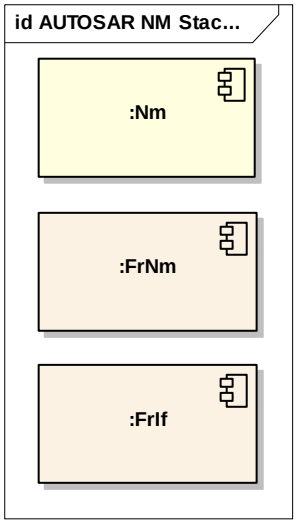

Figure 4-1 AUTOSAR NM Stack on FlexRay

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

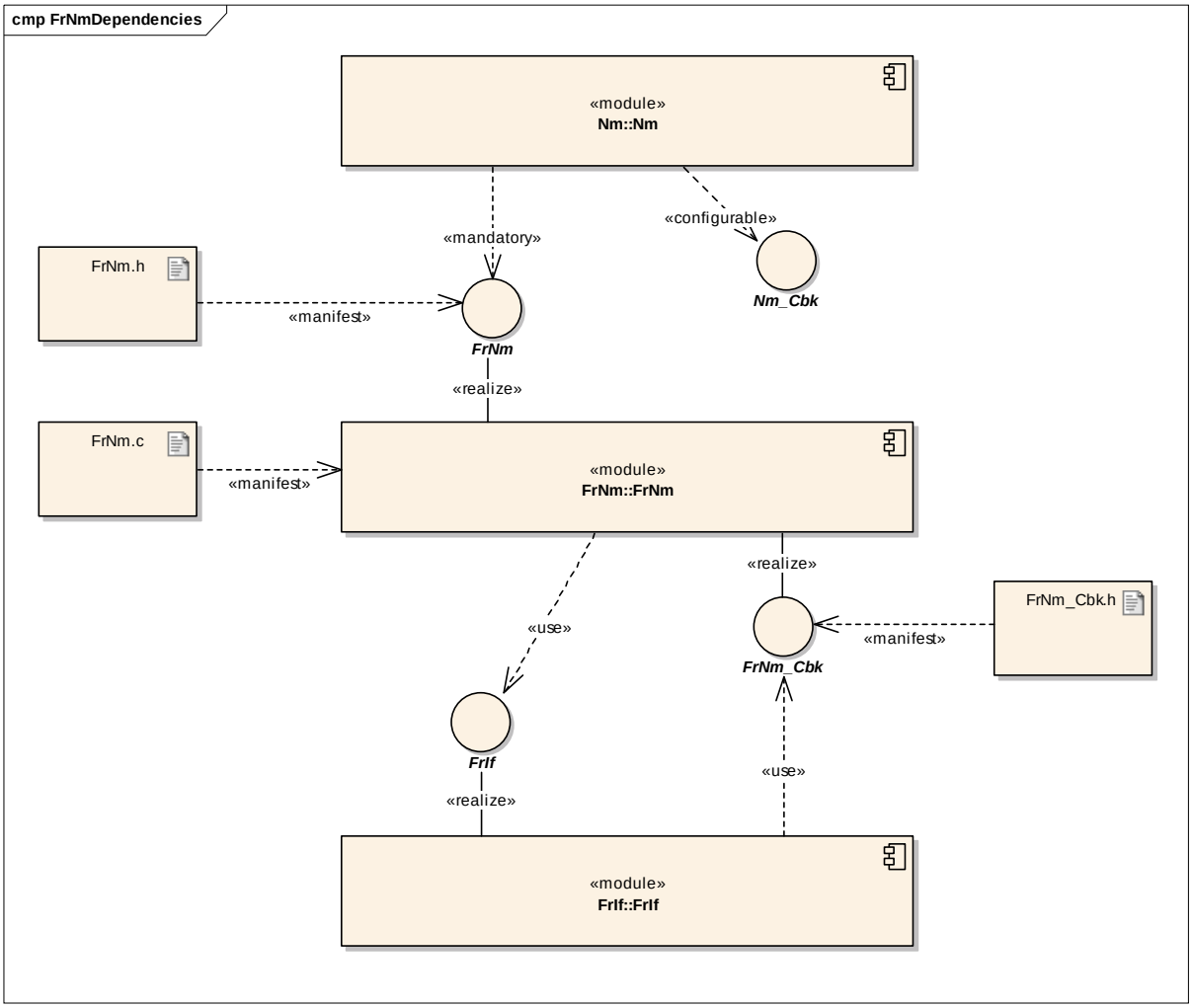

Figure 5-1 NM Overview

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 [14] [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

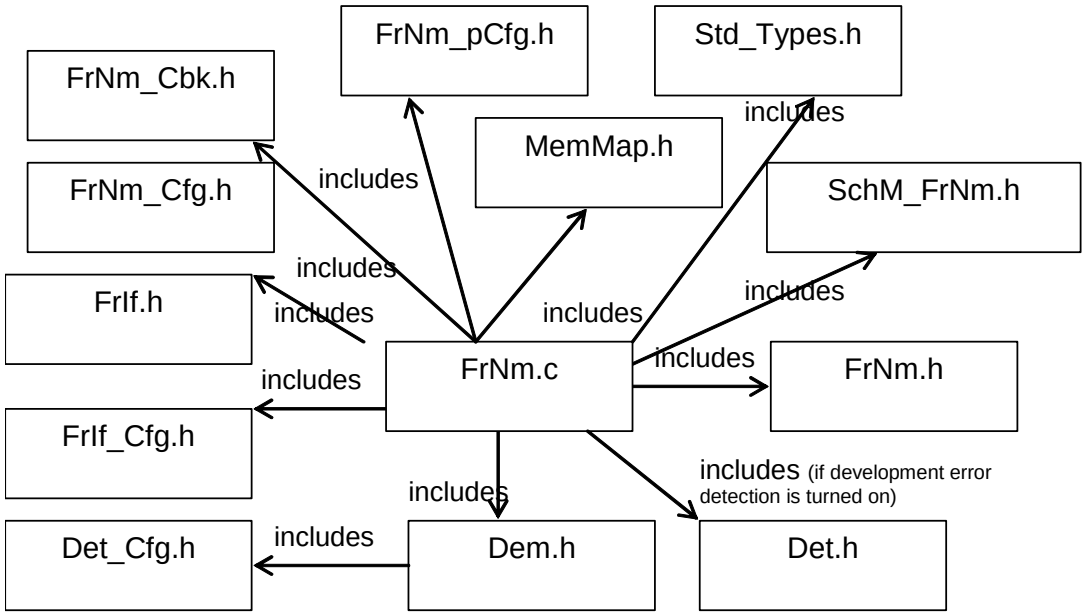

Figure 5-2 Header File Structure

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 interface to Schedule Manager)
- `MemMap.h`

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 8.3.14
[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
[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 7.2.3.1
[BSW02505] Sending node identifier	FRNM222
[BSW02506] Reading node identifier	FRNM047
[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
[BSW02508] Unambiguous node identification per bus	FRNM037

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 [8], [15]).

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 7.8.5 on page 42).

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 NM-Vote messages as an indication that the node requires bus-communication.

[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.

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

Note: The FrNmReadySleepCnt 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: FrNmReadySleepCnt = 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: Internal timers of the FlexRay NM will be described in chapter 7.

FRNM168: The FlexRay NM module shall synchronize state changes in the FlexRay NM state machine to the FlexRay periodic Schedule.

Rationale: The FlexRay NM algorithm is based on the fact that all ECUs, which participate in the FlexRay NM, are synchronized on a global time (based on a periodic repetition of its communication scheme, the so called Cycle – see [4]). To prevent asymmetric behavior of the ECUs (e.g. only a part of the ECUs changes to sleep mode, while the other part stays awake) the FlexRay NM aligns the state changes to a NM Repetition Cycle (which is aligned to a basic FlexRay communication cycle) to guarantee a synchronous behavior of the NM state machines of all ECUs in the NM cluster.

Note: FRNM048 describes on how the implementation will fulfill FRNM168.

7.2 Operational modes

In the following chapter operational modes of the AUTOSAR FlexRay NM coordination algorithm are described in detail. Figure 7-1 (in chapter 7.4 on page 32) shows the detailed UML state chart of the FlexRay NM.

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 by the FlexRay NM module to the upper layer of this module by calling **`Nm_StateChangeNotification`** callback function.

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.

FRNM311: It shall be configurable with the configuration parameter FrNm_MainAcrossFrCycle if the execution of FrNm_Main function crosses the FlexRay cycle boundary (set to TRUE) or not (set to FALSE). Note: FlexRay NM Vector is available at the end of a FlexRay cycle thus evaluation takes place in the following FlexRay cycle.

Evaluation of the condition RepetitionCycleCompleted shall be evaluated according to the following rules:
If FrNm_MainAcrossFrCycle is FALSE RepetitionCycleCompleted := (CycleNumber % RepetitionCycleLength) = 0
If FrNm_MainAcrossFrCycle is TRUE RepetitionCycleCompleted := (FlexRay "CycleEnd" Event) AND (((CycleNumber – 1) % RepetitionCycleLength) = 0)

Hint: Repetition cycle is completed after we receive all the Votes in the last voting cycle.

7.2.1 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 of this module by calling **`Nm_StateChangeNotification`**, 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).

FRNM315 When the FrNm module enters the Bus-Sleep Mode, the module shall indicate **`Nm_BusSleepMode`** to the upper layer, except when Bus-Sleep Mode is entered by default at initialization.

FRNM135: When the FlexRay NM module has entered the Bus-Sleep Mode the module shall set the flag **`FrNm_RepeatMessage`** to FALSE.

FRNM137: When the FlexRay NM module has entered the Bus-Sleep Mode the module shall deactivate the transmission of NM-Data and NM-Vote.

FRNM175: When the FlexRay NM module receives a NM-message successfully and is in the Bus-Sleep Mode it shall notify the upper layer of this module by calling `NM_NetworkStartIndication`.

Rationale: FRNM175 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).

FRNM316: BusSleep Mode shall be left and the Synchronize Mode shall be entered if Generic NM Interface calls FrNm_NetworkRequest.

FRNM317: BusSleep Mode shall be left and the Synchronize Mode shall be entered if Generic NM Interface calls FrNm_PassiveStartUp.

7.2.2 Synchronize Mode

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

FRNM140: When the FlexRay NM module has entered the Synchronize Mode, it shall notify the upper layer of this module by calling **`Nm_StateChangeNotification.`**

FRNM143: The FlexRay NM module shall leave the Synchronize Mode and enter the Network Mode at the first boundary between two NM Repetition Cycles.

FRNM308: When the FlexRay NM module enters the Synchronize mode, it shall deactivate the transmission of NM-Data and the Node shall not vote for keeping the cluster awake (send “negative” NM-Votes).

FRNM340:
If FlexRay NM is in Synchronize state and FlexRay NM receives the indication `FrNm_StartupError` and if **`FrNm_NetworkRequested`** is set to TRUE, then FlexRay NM would remain in Synchronize state.

FRNM376: If FlexRay NM is in Synchronize state and FlexRay NM receives the indication `FrNm_StartupError` and if **`FrNm_NetworkRequested`** is set to FALSE, then FlexRay NM will transit to Bus Sleep State.

7.2.3 Network Mode

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

FRNM115: The FlexRay NM module shall synchronize all state changes into and within the Network Mode 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 44.

FRNM108: When the FlexRay NM module has entered the Network Mode, it shall enter the Network Mode internal state Repeat Message.

FRNM109: When the FlexRay NM module has entered the Network Mode, it shall set the flag **`FrNm_RepeatMessage`** to TRUE.

FRNM110: When the FlexRay NM module has entered the Network Mode, it shall notify the upper layer of this module by calling **`Nm_StateChangeNotification`**.

FRNM133: FlexRay NM module shall left the Network Mode and enters the Bus-Sleep Mode if the Repetition Cycle is completed and FrNm_ReadySleepCnt < 1

FRNM111: When the FlexRay NM module receives a Repeat Message Request successfully and is in the Network Mode it shall set the flag **`FrNm_RepeatMessage`** to TRUE.

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

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

FRNM119: When the FlexRay NM module has entered the Network Mode, it shall initialize the NM Repeat Message Timer (**`FrNm_RepeatMessageTimer`**) for a configurable amount of time determined by **`FrNmRepeatMessageTime`** (configuration parameter).

FRNM307: The FlexRay NM module shall leave the Network mode immediately if the function **`FrNm_Init`** is called.

FRNM309: The type of **`FrNm_ReadySleepCnt`** shall be set to a type which is capable to support the maximum value of **`FrNmReadySleepCnt`** (configuration parameter)

7.2.3.1 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 (see explanation in chapter 7.8.3).

FRNM116: When the FlexRay NM module has entered the Repeat Message State it shall activate the transmission of NM-Data 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 FlexRay NM module has entered the Repeat Message State it shall start the NM Repeat Message Timer (**`FrNm_RepeatMessageTimer`**).

FRNM120: The FlexRay NM module shall leave the Repeat Message State when a Repetition Cycle is completed and if the flag **`FrNm_RepeatMessage`** is set to FALSE, and enter either the Normal Operation state (if **`FrNm_NetworkRequested`** is set to TRUE) or the Ready Sleep State (if **`FrNm_NetworkRequested`** is set to FALSE).

FRNM121: When the FlexRay NM module lefts the Repeat Message State (see FRNM120) it shall enter the Normal Operation State if the flag **`FrNm_NetworkRequested`** is set to TRUE (see FRNM113).

FRNM122: When the FlexRay NM module lefts the Repeat Message State (see FRNM120) it shall enter the Ready Sleep State if the flag **`FrNm_NetworkRequested`** is set to FALSE (see FRNM114).

FRNM383: If FlexRay NM is in Repeat Message state and FlexRay NM receives the indication `FrNm_StartupError`, then FlexRay NM transits to Synchronize state. This transition shall only be executed if **`FRNM_CYCLE_COUNTER_EMULATION`** is set to FALSE.

FRNM384: If global time could not be retrieved, then FlexRay NM transits to Synchronize state. This transition shall only be executed if **`FRNM_CYCLE_COUNTER_EMULATION`** is set to FALSE.

FRNM385: If FlexRay NM is in Repeat Message state and FlexRay NM receives the indication `FrNm_StartupError`, then FlexRay NM transits to Bus Sleep state. This transition shall only be executed if **`FRNM_CYCLE_COUNTER_EMULATION`** is set to TRUE.

FRNM386: If global time could not be retrieved, then FlexRay NM transits to Bus Sleep state. This transition shall only be executed if **`FRNM_CYCLE_COUNTER_EMULATION`** is set to TRUE.

7.2.3.2 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 FlexRay NM module enters the Normal Operation State, it shall activate the transmission of NM-Data and the Node shall vote for keeping the cluster awake (send “positive” NM-Votes).

FRNM124: The FlexRay NM module shall leave the Normal Operation State and enter the Repeat Message State if a Repeat Message Request has been detected (the flag **`FrNm_RepeatMessage`** is set to TRUE – see FRNM111)and if a Repetition Cycle is completed.

FRNM125: The FlexRay NM module shall leave the Normal Operation State and enter the Ready Sleep State if no Repeat Message Request is active (the flag **`FrNm_RepeatMessage`** is set to FALSE) and the network has been released (the flag **`FrNm_NetworkRequested`** is set to FALSE – see FRNM114) and if a Repetition Cycle is completed.

FRNM342: If FlexRay NM is in Normal Operation state and FlexRay NM receives the indication `FrNm_StartupError` or when global time could not be retrieved, then FlexRay NM would transition to Synchronize state.

7.2.3.3 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 FlexRay NM module enters the Ready Sleep State, it shall deactivate the transmission of NM-Data and the Node shall not vote for keeping the cluster awake (send “negative” NM-Votes).

FRNM127: When the FlexRay NM module enters the Ready Sleep State from another state than the Ready Sleep State, it shall initialize the Ready Sleep Counter (**`FrNm_ReadySleepCnt`**) with the start value **`FrNmReadySleepCnt`** (configurable parameter) – see FRNM101.

FRNM128: When the FlexRay NM module detects a NM-Vote to keep the cluster awake it shall initialize the Ready Sleep Counter (**`FrNm_ReadySleepCnt`**) with the start value **`FrNmReadySleepCnt`** (configurable parameter).

FRNM129: The FlexRay NM module shall leave the Ready Sleep State (and the Network Mode) and enter the Bus-Sleep Mode if the Ready Sleep Counter has expired (**FrNm_ReadySleepCnt** < 1) at the end of the NM Repetition Cycle.

Note: As all transitions regarding the **FrNm_ReadySleepCnt** are guarded the order of evaluation is implicitly defined by the guards. E.g. the decrement of the **FrNm_ReadySleepCnt** is only done 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).

FRNM130: The FlexRay NM module shall leave the Ready Sleep State and enter the Repeat Message State if a Repetition Cycle is completed, the Ready Sleep Counter has not expired (**FrNm_ReadySleepCnt** >0) and a Repeat Message Request is active (the flag **FrNm_RepeatMessage** is set to TRUE – see FRNM111).

FRNM131: The FlexRay NM module shall leave the Ready Sleep State and enter the Normal Operation State if a Repetition Cycle is completed, the Ready Sleep Counter has not expired (**FrNm_ReadySleepCnt** >0) and no Repeat Message Request is active (the flag **FrNm_RepeatMessage** is set to FALSE) and the network has been requested (the flag **FrNm_NetworkRequested** is set to TRUE – see FRNM113).

FRNM132: The FlexRay NM module shall decrement the Ready Sleep Counter (**FrNm_ReadySleepCnt**) at the end of an NM Repetition Cycle and before the evaluation of state change requests if no Repeat Message Request is active (the flag **FrNm_RepeatMessage** is set to FALSE) and the network has been released (the flag **FrNm_NetworkRequested** is set to FALSE) and the Ready Sleep Counter has not expired (**FrNm_ReadySleepCnt** >0).

FRNM314: When both event “ClusterAwakeVote detected” and “Repetition Cycle completed” in state “Ready Sleep” occurs at the same time the transition where “ClusterAwakeVote detected” is executed first.

Note: A local request to keep the network awake (i.e., Network Request) will be Ignored in the Ready Sleep State when the FrNmReadySleepCnt has already reached 0.

FRNM338: If the configuration parameter **FRNM_CYCLE_COUNTER_EMULATION** is set to FALSE, then on reception of **FrNm_StartupError** the FlexRay NM will transition to Synchronize state.

FRNM378: If **FRNM_CYCLE_COUNTER_EMULATION** is set to TRUE, the timer **FrNm_SyncLossTimer** shall be set to the initial value on every reception of the NM vote. The timer is decremented based on an OS timer.

FRNM379: If **FRNM_CYCLE_COUNTER_EMULATION** is set to TRUE and the FlexRay Global Time could not be retrieved, every time the timer **FrNm_SyncLossTimer** expires, the FrNm_ReadySleepCnt is decremented by 1.

FRNM380: If **`FRNM_CYCLE_COUNTER_EMULATION`** is set to TRUE and the FlexRay Global Time could not be retrieved, every time the timer **`FrNm_SyncLossTimer`** expires, the timer shall be reset to the initial vale.

7.3 Network states

Network states (i.e. ‘requested’ and ‘released’) are two additional “states” (which are stored in the **`FrNm_NetworkRequested`** flag) of the AUTOSAR FlexRay NM state machine that exist in parallel to the state machine described in chapter 7.4. Network states distinguish whether the software components need to communicate on the bus (the network state is then ‘requested’); or not (the network state is then ‘released’). Note that if the network is released an ECU may still communicate because at least one other ECU still requests the network.
This network states reflect the demand of an upper layer (e.g. ComM) on keeping the bus awake (network requested); or not (network released).

7.4 UML State Chart Diagram

The Figure 7-1 shows an UML state diagram with respect to the API specification.

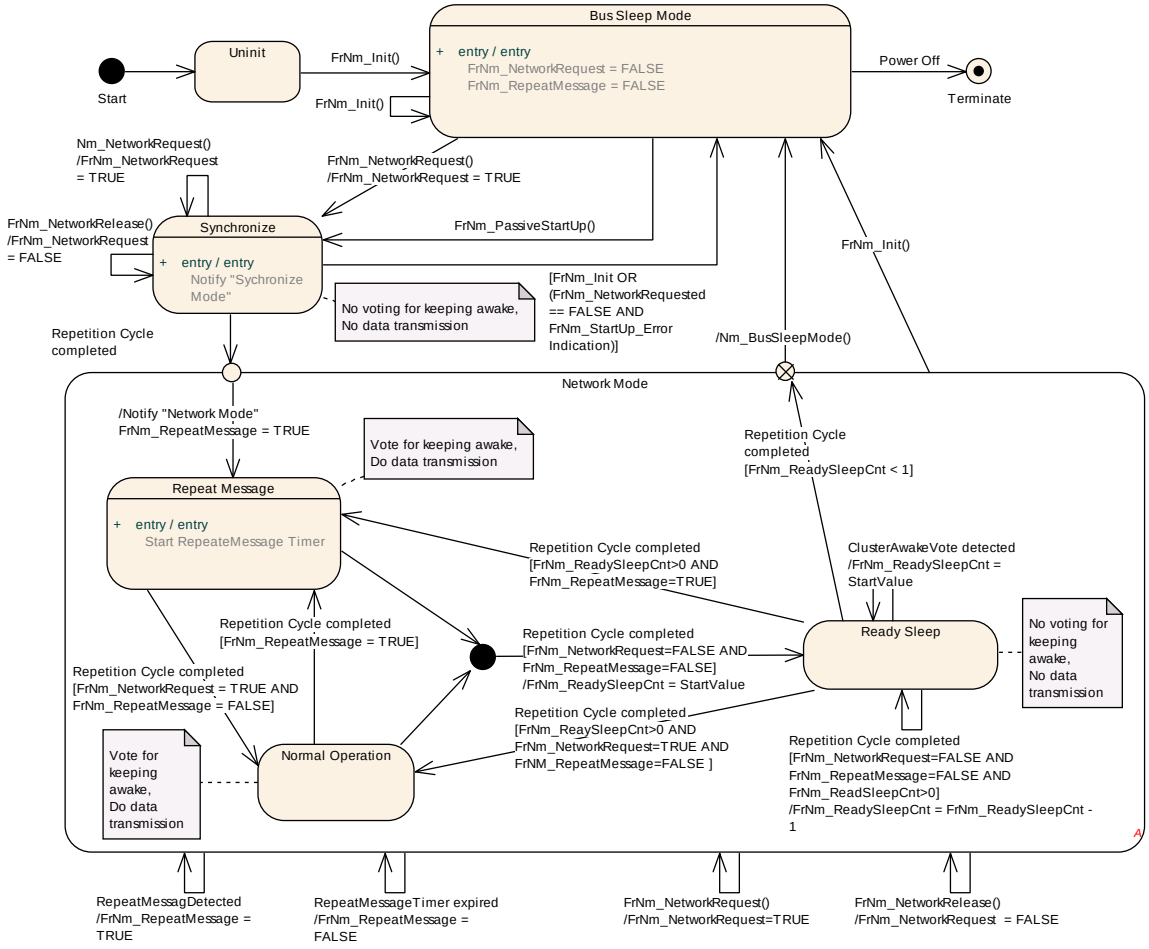

Figure 7-1 UML State Chart

7.5 Initialization and Startup

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

FRNM072: After a reset, the FlexRay NM module shall set the initialization status to **`NM_UNINIT`**.

FRNM029: The FlexRay NM module's environment shall initialize the FlexRay NM module after the corresponding FlexRay Interface is initialized and before any other FlexRay NM service is called.

FRNM032: If the FlexRay NM module is not initialized: the FlexRay NM module shall reject a call of any FlexRay NM function, except **`FrNm_Init`**, and return with a respective error code.

FRNM136: FlexRay NM shall set the flag FrNm_NetworkRequest to FALSE after initialization

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

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

Note: FlexRay supports an automatic aggregation of Network Management data – called Network Management Vector (see chapter 9.3.3.4 Network management service, page 209, of the FlexRay protocol specification [4]).
This NM-Vector is calculated by the FlexRay Controller by exchanging the NM-Vector in selected network management enabled frames within the static segment of the communication cycle. Every node may be configured to send one NM-Vector in one of its send slots.
Throughout each communication cycle the FlexRay communication controller will maintain an aggregated network management vector by applying a bit-wise OR between each received Network Management Vector (regardless of whether the frame is subscribed to a receive buffer).
This method is a powerful way to receive the NM-Vector of nodes on the network without involving the CPU, but requires at least one send-slot for every node (participiating in the Network Management) in the static segment.

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 only contain NM-Vote, NM-Data or both.

FRNM147: The FlexRay NM module 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 [4]). 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 by the configuration parameter **`FRNM_PDU_SCHEDULE_VARIANT`** which NM-message transmission formats (NM-Data, NM-Vote and the combined NM-Data/Vote format) are recognized by the FlexRay NM module.

Rationale: The FlexRay NM module must be capable to receive and process the NM-messages which are on the FlexRay bus (FRNM058). To optimize the resource need of the NM it must be configurable which formats are supported by the NM, in order to avoid the overhead of “unused” formats.

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

Note: The FrIf configuration is responsible for the actual FlexRay schedule configuration. This requirement is to require that the FrNm will support such a configuration.

FRNM169: Every FlexRay NM node shall be independently configurable the configuration parameter **`FrNmHwVoteEnable`** _to use the FlexRay NM HW support for receptions of NM-Votes which are transmitted in the static segment.

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

Note: The FrIf configuration is responsible for the actual FlexRay schedule configuration. This requirement is to require that the FrNm will support such a configuration.
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: The support of the Control Bit Vector for the FlexRay NM module shall be configurable at pre-compile time by the configuration parameter **`FrNmControlBitVectorEnabled`** (see chapter 10).

FRNM222: The support of the Source Node Identifier for the FlexRay NM module shall be configurable at pre-compile time by the configuration parameter **`FRNM_SOURCE_NODE_IDENTIFIER_ENABLED`** (see chapter 10).

FRNM154: The length of the NM Data PDU for the FlexRay NM module shall be configurable at pre-compile time to any integer value from 0 to 8 by the configuration parameter **`FrNmPduLength`** (see chapter 10).

FRNM313: The behaviour of the FlexRay NM module to send NM-Data PDU shall be configurable at pre-compile by the configuration parameter **`FRNM_NM_DATA_ENABLED`** (see chapter 10).

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

FRNM156: The NM-Data PDU Control Bit Vector of the FlexRay NM module 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 of the FlexRay NM module shall contain a Repeat Message Request Bit (RptMsgRequest) with the following meaning:
0: Repeat Message State not requested
1: Repeat Message State requested

FRNM213: The support of the Repeat Message Bit for the FlexRay NM module shall be configurable at pre-compile time by the configuration parameter **`FrNmRepeatMessageBitEnabled`** (see chapter 10).

FRNM157: The FlexRay NM module shall not use Bit 7 of the Control Bit Vector.

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: The FlexRay NM module shall set Bit 7 of the Control Bit Vector to 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: The FlexRay NM module shall set Bit 1 to 6 of the Control Bit Vector to 0_b which are reserved for future extension.

7.6.2.2 FlexRay NM-Vote PDU format

FRNM215: The NM-Vote PDU format of the FlexRay NM module 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 of the FlexRay NM module shall contain a Voting Bit (Vote) with the following meaning:
0: vote against keeping awake
1: vote for keeping awake

FRNM218: The FlexRay NM module shall not use Bits 0-6 of the NM-Vote PDU.

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: The FlexRay NM module shall set Bits 0-6 of the NM-Vote PDU to 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.

FRNM163: The FlexRay NM module shall place the PDU containing the NM-Vote 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 payload of a FlexRay frame (see also [4]). 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 44) 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 FlexRay NM module shall combine the NM-Vote PDU Format with the Control Bit Vector Format of the NM-Data PDU in case the FlexRay NM module shall transmit the NM-Vote in the same PDU as the NM-Data.

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 [7].

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 [7].

7.6.5 Functional requirements on FrNm API

The following requirements define the available FlexRay NM functions.

FRNM037: The set of the Source Node Identifier shall be configurable at pre-compile time by the configuration parameter **`FrNmNodeId`** (see chapter 10).

7.7 Execution

The FlexRay NM State machine and hence the NM-task execution has to be synchronized with the FlexRay bus schedule (FRNM168). FlexRay NM decisions and state changes have to be aligned to the FlexRay Bus Cycle. To guarantee synchronized state changes and decisions of the FrNm, the FrNM Mainfunction (FrNm_MainFunction_<NmClstIdx>) has to be executed within a specific time window as shown in Figure 7-2 (below). The borders of this window are defined on one side by the availability of all NM-Votes of the actual cycle and on the other side by the last point in time where the own NM-Vote (of the next cycle) has to be sent.
As the relative time for a FlexRay 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 a AUTOSAR OS Schedule Table which is synchronized to the global (FlexRay) time.

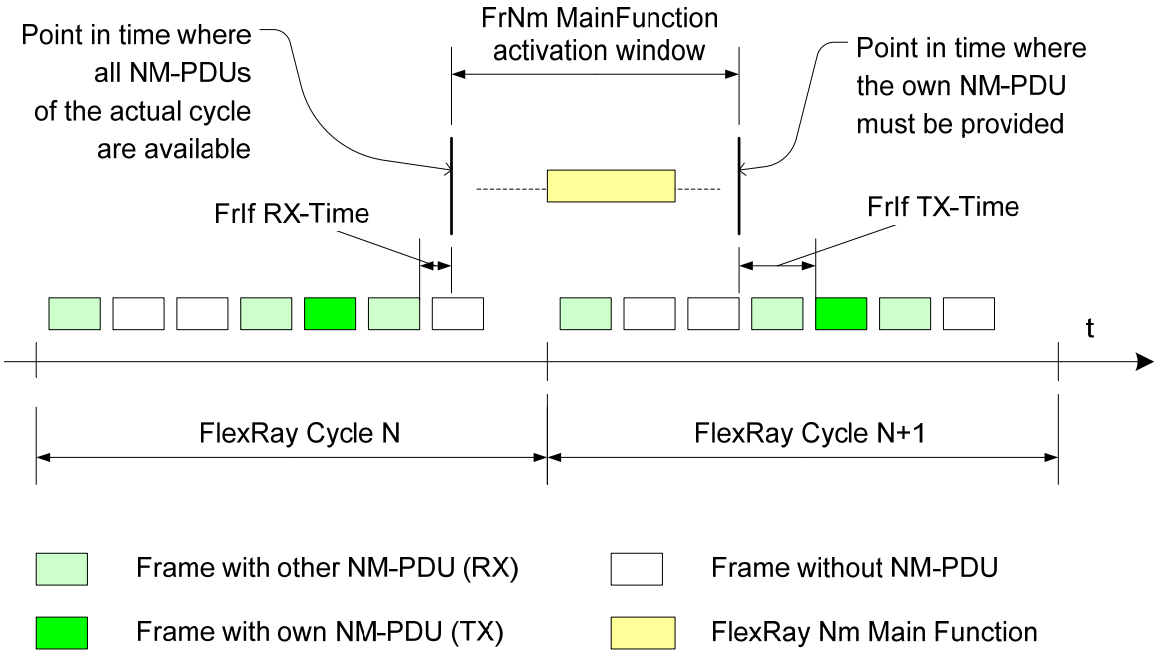

Figure 7-2 FrNm Mainfunction Execution

7.7.1 General requirements

FRNM225: The FlexRay NM module's implementer shall realize the FlexRay NM coordination algorithm processor independent, which means the FlexRay NM coordination algorithm 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 and for Transmit of NM Messages.

Note: It is could be up to the implementation if there is only one NM-Task (which is activated by the **`FrNm_<Rx|Tx>Confirmation`** callbacks) or several NM-Tasks with specific functionality, but [3][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 module's environment should execute the FlexRay NM module (**`FrNm_MainFunction_<NmClstIdx>`**) at least once a FlexRay communication cycle.

Note: This requirement is a recommendation that need not to be fulfilled in every implementation.

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: The FlexRay NM module shall call the FlexRay Interface function **`FrIf_Transmit`** to transmit NM-Vote and NM-Data if the transmission of cyclic NM-messages is started.

The FlexRay Interface module shall call the FlexRay NM module function **`FrNm_TxConfirmation`** if a NM-message is successfully transmitted.

The FlexRay Interface module shall call the FlexRay NM module function **`FrNm_RxIndication`** if a NM-message is received.

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 module's integrator shall define a schedule to activate the FlexRay NM Task 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 and any of the votes are overwritten in the new communication cycle.

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 AUTOSAR OS [14] provides the method of Schedule Tables which can be synchronized with a Global Time. These could be used to fulfill the FlexRay NM execution requirements.

7.7.3.2 Asynchronous FlexRay NM-Task execution

FRNM177: The FlexRay NM module 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: The detection of remote sleep indication by the FlexRay NM module shall be configurable at pre-compile time by the configuration parameter **`FrNmRemoteSleepIndicationEnabled`** (see chapter 10).

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 **`FrNmRemoteSleepIndTime`** (configuration parameter), the NM shall notify the upper layer of this module that all other nodes in the cluster are ready to sleep (the ‘Remote Sleep Indication’) by calling **`<UL>_RemoteSleepIndication`** .

FRNM186: The FlexRay NM module shall reject a check of Remote Sleep Indication (**`FrNm_CheckRemoteSleepIndication`**) when not in Network Mode. The function **`FrNm_CheckRemoteSleepIndication`** shall immediately return the value **`NM_E_NOT_OK`** and shall not execute any functionality.

FRNM229: If FlexRay NM module 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 the upper layer of this module that some nodes in the cluster are not ready to sleep anymore (the ‘Remote Sleep Cancellation’) by calling **`<UL_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 the upper layer of this module that some nodes in the cluster are not ready to sleep anymore (the 'Remote Sleep Cancellation') by calling **`<UL_RemoteSleepCancellation`**

7.8.3 Detection of Nodes (optional)

Nodes which have the Node Detection Feature enabled (FRNM170) will send Identification Data (NM-Data PDU) on the bus (see chapter 7.6.2) if in the *Repeat Message State* (see chapter 7.2.3.1) or the *Normal Operation State*. This data can be received by other nodes using the **`FrNm_GetNodeIdentifier`** function.
A node can identify (detect) nodes on the FlexRay Bus by repeated calling of the previous mentioned function.
To ensure that all nodes (which are configured to do so) will send their Identification Data the **`FrNm_RepeatMessageRequest`** can be used to bring all nodes to the *Repeat Message State*.

FRNM170: The support of node detection for the FlexRay NM module shall be configurable at pre-compile time by the configuration parameter **`FrNmNodeDetectionEnabled`** (see chapter 10).
FrNmSourceNodeIdentifierEnabled have to be also on.

7.8.4 User data (optional)

FRNM077: The support of user data for the FlexRay NM module shall be configurable at pre-compile time by the configuration parameter **`FrNmUserDataEnabled`** (see chapter 10).

7.8.5 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: The Passive Node Configuration shall be configurable at pre-compile time by the configuration parameter **`FRNM_PASSIVE_NODE_ENABLED`** (see chapter 10).

FRNM188: If Passive Node Configuration is enabled (FRNM187), the FlexRay NM module shall not use the Remote Sleep Indication options (FRNM180).

Note: Configuration parameter **`FrNmRemoteSleepIndicationEnabled`** shall be set to false.

7.8.6 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: The NM PDU Reception indication shall be configurable at pre-compile time by the configuration parameter **`FrNmPduRxIndicationEnabled`** (see chapter 10).

FRNM190: If NM PDU Reception indication is enabled (FRNM189), the FlexRay NM module shall call the function `NM_PduRxIndication` at the successful reception of an NM-PDU.

7.8.7 State change notification (optional)

FRNM191: The optional state change notification service shall be configurable at pre-compile time by the configuration parameter **`FrNmStateChangeIndicationEnabled`** (see chapter 10).

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

7.8.8 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, by sending (and receiving) two PDUs (one for channel A and one for channel B). The following requirements describe the required functionality.

FRNM231: The dual channel PDU support of the FlexRay NM shall be statically configurable at pre-compile time by the configuration parameter **`FrNmDualChannelPduEnable`** (see chapter 10).

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 PDU corresponds to the NM-Vote
3. NM-Vote and NM-Data are transmitted in the static segment in separate PDUs. This alternative is not recommended -→ Alternative 1 should be used instead.
4. NM-Vote transmitted in static and NM-Data transmitted in dynamic segment.
5. NM-Vote is transmitted in dynamic and NM-Data is transmitted in static segment. This alternative is not recommended -→ Possibility 2 or 6 should be used instead.
6. NM-Vote and NM-Data are transmitted in dynamic segment in separate PDUs.
7. NM-Vote and a copy of the CBV are transmitted in the static segment (using the FlexRay NM Vector support) and NM-Data is transmitted in the dynamic segment (see chapter 7.6.2.3).

		FlexRay Segment	
		Static	Dynamic
PDU Type	NM-Vote	3 and 4	5 and 6
	NM-Data	3 and 5	4,6 and 7
	NM-Vote and NM-Data	1	2
	Combined NM-Vote and CBV	7	-

Table 7-5 Summary of FlexRay PDU schedule alternatives

Note: If the NM-Vote is transmitted in the static segment of the FlexRay schedule (alternative 1,3,4 and 7), the usage of the HW NM-Vector support is possible.

Although every node can be configured independently to one of the above seven 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. 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 7.8.8).

FRNM233: The schedule variant for the FlexRay NM module shall be statically configurable at pre-compile time by the configuration parameter **`FRNM_PDU_SCHEDULE_VARIANT`** (see chapter 10).

FRNM234: In case that a combined NM-Vote and CBV is transmitted in static and dynamic segment (option 7 in Table 7-5) the FlexRay NM module shall use the combined NM-Vote and CBV in the static part for evaluation of the NM-Vote.

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 module's integrator shall define the FlexRay NM Voting Cycle (**`FrNmVotingCycle`** configuration parameter) 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 96).

FRNM194: The FlexRay NM module's integrator shall define the FlexRay NM Data Cycle (**`FrNmDataCycle`** configuration parameter) 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 97).

FRNM195: The FlexRay NM schedule specific cycle configuration parameters **`FrNmVotingCycle`**, **`FrNmDataCycle`** and **`FrNmRepetitionCycle`** 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 (**`FrNmRepetitionCycle`** configuration parameter) shall be an integer multiple (including 1) of the NM Voting Cycle (**`FrNmVotingCycle`**)

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 96).

7.9.2 NM-Message scheduled requirements

There are no schedule requirements for the FlexRay NM messages. Anyhow it is highly recommended for frames, containing NM information, which are sent in the dynamic segment of the FlexRay Schedule, to choose the first slots in the dynamic segment and to contain only NM information (NM-Vote and/or NM-Data) 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).

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 **`FrNmMsgTimeoutTime`** (configuration parameter), the FlexRay NM shall notify the the upper layer of this module that transmission failure has occurred by calling **`<UL>_TransmissionTimeoutException`**.

FRNM223: If the FlexRay bus communication of a FlexRay NM cluster has been shut down the corresponding FlexRay NM shall be shut down, by the BSW upper layer which is in control of FrNm.

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 cluster” 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 clusters”.

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

7.11 Error classification

FRNM294: Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file Dem_IntErrId.h and included via Dem.h.

FRNM293: Development error values are of type uint8.

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`
API service called with Invalid pointer	Development	`FRNM_E_INVALID_POINTER`	`03h`

7.12 Error detection

FRNM049: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch **`FrNmDevErrorDetect`** (see chapter 10) shall activate or deactivate the detection of all development errors.

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.

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.

FRNM050: If not initialized, the FlexRay NM module shall reject every API function other than **`FrNm_Init`**. The called function shall not be executed and shall return **`NM_E_NOT_OK`** (if possible) and the FlexRay NM module shall report the error code **`FRNM_E_NO_INIT`** to the Development Error Tracer (if DET is enabled).

FRNM051: When the FlexRay NM API service with an invalid handle is called, the corresponding function shall report **`FRNM_E_INVALID_CHANNEL`** error to the Development Error Tracer (if DET is enabled) and it shall return **`NM_E_NOT_OK`** to the calling function

FRNM295: The detection of production code errors cannot be switched off.

7.13 Error notification

FRNM022: Detected development errors shall be reported to the *Det_ReportError* service of the Development Error Tracer (DET) if the pre-processor switch **FrNmDevErrorDetect** is set (see chapter 10).

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

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

Note: Currently no production errors are defined.

7.14 Parameter check

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

FRNM284: The FlexRay NM module's implementer shall realize a parameter type check at compile time.

FRNM285: If parameter type checks 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.

FRNM286: The FlexRay NM module's implementer shall realize a parameter value check (for parameters of the constant value) at configuration time.

FRNM287: If parameter value check fails, the configuration process shall be stopped and respective configuration error shall be reported.

7.15 Version check

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

8 API specification

FlexRay NM API consists of services that are FlexRay specific and can be called as required.

FRNM034: Each service other than **`Nm_Init`** refers to one NM cluster only.

Note: The phrase “NM cluster” 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 cluster”.

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

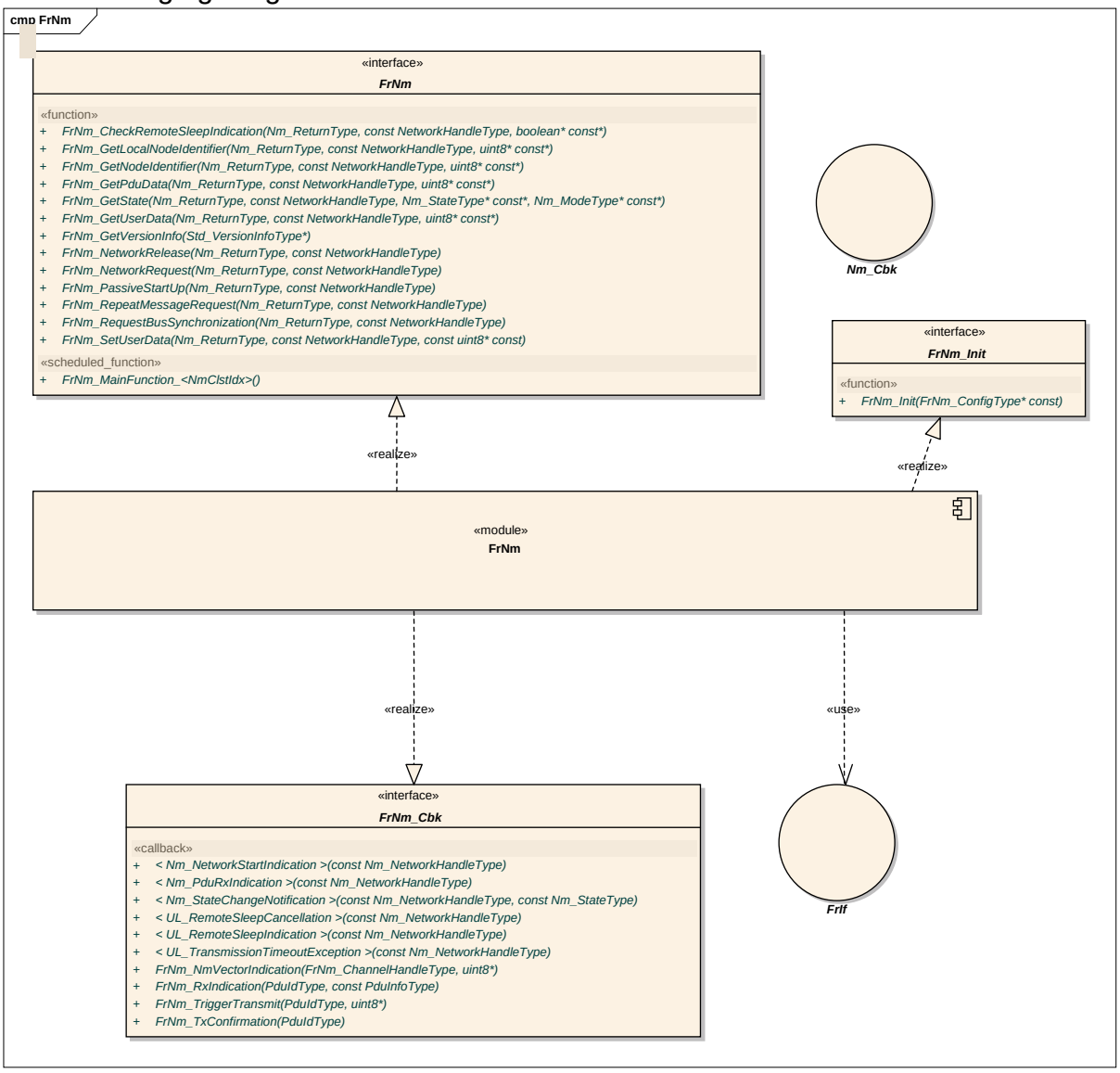

Figure 8-1 API Specification (Overview)

8.1 Imported types

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

FRNM235:

Header file	*Imported Type*
Nm_Types.h	Nm_ModeType
	Nm_StateType
	Nm_ReturnType
	Nm_NetworkHandleType
ComStack_Types.h	PduIdType
Dem_Types.h	Dem_EventIdType
FrNm_Types.h	NetworkHandleType
PrimitiveTypes.h	PduInfoType
Std_Types.h	Std_VersionInfoType
	Std_ReturnType

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 ([6]) in chapter 8.2 Type definitions.

8.2.2 FlexRay NM specific Type Definitions

8.2.2.1 FrNm_ConfigType

Name:	`FrNm_ConfigType`
Type:	`Structure`
Range:	Implementation specific.
Description:	Contains configuration parameters.

For **`FrNm_ConfigType`** see chapter 10.5 on page 87.

8.3 Function definitions: FrNm Services provided to upper layers

8.3.1 FrNm_Init

FRNM236:

Service name:	FrNm_Init	
Syntax:	`void FrNm_Init( FrNm_ConfigType* const nmConfigPtr )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	nmConfigPtr	Pointer to the selected configuration set.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the FlexRay NM and its internal state machine.	

FRNM028: The function **`FrNm_Init`** shall initialize the FlexRay NM module.

FRNM073: After a successful initialization of the FlexRay NM module, the function **`FrNm_Init`** shall set the initialization status to **`NM_INIT`**, otherwise to **`NM_UNINIT`**.

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

FRNM030: The function **`FrNm_Init`** shall deactivate the periodic transmission of NM-Vote and NM-Data after the initialization of the FlexRay NM module (see FRNM100, FRNM137).

FRNM042: After the initialization of the FlexRay NM module the function **`FrNm_Init`** shall set the Reserved Bytes in the NM-Data and NM-Vote PDU to 0.

FRNM045: After the initialization of the FlexRay NM module the function **`FrNm_Init`** shall set the User Data bytes to FFh.

FRNM033: The function **`FrNm_Init`** shall initialize the FlexRay NM module.

8.3.2 FrNm_PassiveStartUp

FRNM237:

Service name:	FrNm_PassiveStartUp	
Syntax:	`Nm_ReturnType FrNm_PassiveStartUp( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	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:	Initiates the Passive Startup of the FlexRay NM.	

FRNM258: The function FrNm_PassiveStartUp shall initiate the Passive Startup of the FlexRay NM.

FRNM138: The function FrNm_PassiveStartUp shall trigger the transition from Bus-Sleep Mode to the Network Mode in Repeat Message State (via the Synchronize State).

FRNM260: The function FrNm_PassiveStartUp shall have no effect on the operation mode of the FlexRay NM module if the current state is not Bus-Sleep Mode. In that case the function FrNm_PassiveStartUp shall return `NM_E_NOT_EXECUTED` (see FRNM138).

8.3.3 FrNm_NetworkRequest

FRNM238:

Service name:	FrNm_NetworkRequest	
Syntax:	`Nm_ReturnType FrNm_NetworkRequest( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	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'.	

FRNM139: If the FlexRay NM module's environment is calling the function **`FrNm_NetworkRequest`** in the Bus-Sleep Mode, the function **`FrNm_NetworkRequest`** shall leave the Bus-Sleep Mode, set the Network Request Flag **`FrNm_NetworkRequested`** to TRUE and enter the Synchronize Mode.

FRNM141: If the FlexRay NM module's environment is calling the function **`FrNm_NetworkRequest`** in the Synchronize Mode, the function **`FrNm_NetworkRequest`** shall set the Network Request Flag **`FrNm_NetworkRequested`** to TRUE.

FRNM113: If the FlexRay NM module's environment is calling the function **`FrNm_NetworkRequest`** in the Network Mode, the function **`FrNm_NetworkRequest`** shall set the Network Request Flag **`FrNm_NetworkRequested`** to TRUE.

FRNM261: The function FrNm_NetworkRequest shall be only available if the configuration parameter FRNM_PASSIVE_NODE_ENABLED is set to OFF.

8.3.4 FrNm_NetworkRelease

FRNM239:

Service name:	FrNm_NetworkRelease	
Syntax:	`Nm_ReturnType FrNm_NetworkRelease( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	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'.	

FRNM142: If the FlexRay NM module's environment is calling the function **`FrNm_NetworkRelease`** in the Synchronize Mode, the function **`FrNm_NetworkRelease`** shall set the Network Request Flag **`FrNm_NetworkRequested`** to FALSE.

FRNM114: If the FlexRay NM module's environment is calling the function **`FrNm_NetworkRelease`** in the Network Mode, the function **`FrNm_NetworkRelease`** shall set the Network Request Flag **`FrNm_NetworkRequested`** to FALSE.

FRNM262: The function FrNm_NetworkRelease shall be only available if the configuration parameter FRNM_PASSIVE_NODE_ENABLED is set to OFF.

8.3.5 FrNm_SetUserData

FRNM240:

Service name:	FrNm_SetUserData	
Syntax:	`Nm_ReturnType FrNm_SetUserData( const NetworkHandleType NetworkHandle, const uint8* const nmUserDataPtr )`	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
	nmUserDataPtr	User data for the next transmitted NM message
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	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.	

FRNM043: If user data handling is enabled for the FlexRay NM module, the function **`FrNm_SetUserData`** shall set the user data.

FRNM263: The function FrNm_SetUserData shall be only available if the configuration parameter FrNmUserDataEnabled is set to ON. (see FRNM077)

8.3.6 FrNm_GetUserData

FRNM241:

Service name:	FrNm_GetUserData	
Syntax:	`Nm_ReturnType FrNm_GetUserData( const NetworkHandleType NetworkHandle, uint8* const nmUserDataPtr )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmUserDataPtr	Pointer to the location where the user data from the last successfully received NM message shall be copied.
Return value:	Nm_ReturnType	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.	

FRNM044: If user data handling is enabled for the FlexRay NM module, the function **`FrNm_GetUserData`** shall provide the user data from the last received NM-Data PDU.

FRNM264: The function FrNm_GetUserData shall be only available if the configuration parameter FrNmUserDataEnabled is set to ON. (see FRNM077)

8.3.7 FrNm_GetPduData

FRNM242:

Service name:	FrNm_GetPduData	
Syntax:	`Nm_ReturnType FrNm_GetPduData( const NetworkHandleType NetworkHandle, uint8* const nmPduData )`	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmPduData	Pointer where NM PDU shall be copied to.
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Getting of NM PDU data has failed
Description:	Gets PDU data.	

FRNM265: The function **`FrNm_GetPduData`** shall get the whole PDU data out of the most recently received NM message.

FRNM266: The function **`FrNm_GetPduData`** shall be only available if the configuration parameter **`FrNmUserDataEnabled`** is set to ON.

8.3.8 FrNm_RepeatMessageRequest

FRNM243:

Service name:	FrNm_RepeatMessageRequest	
Syntax:	`Nm_ReturnType FrNm_RepeatMessageRequest( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	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.	

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

FRNM226: If the FlexRay NM module's environment is calling the function **`FrNm_RepeatMessageRequest`** in the Network Mode and if the Node detection service is activated (**`FrNmNodeDetectionEnabled`**), the function **`FrNm_RepeatMessageRequest`** shall set the flag **`FrNm_RepeatMessage`** to TRUE.

FRNM228: The function FrNm_RepeatMessageRequest shall be only available if the configuration parameter FrNmNodeDetectionEnabled is set to ON. (see FRNM170)

8.3.9 FrNm_GetNodeIdentifier

FRNM244:

Service name:	FrNm_GetNodeIdentifier	
Syntax:	`Nm_ReturnType FrNm_GetNodeIdentifier( const NetworkHandleType NetworkHandle, uint8* const nmNodeIdPtr )`	
Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmNodeIdPtr	Pointer to the location where the node identifier from the last successfully received NM-message shall be copied.
Return value:	Nm_ReturnType	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.	

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

FRNM267: The function FrNm_GetNodeIdentifier shall be only available if the configuration parameter FrNmNodeDetectionEnabled is set to ON. (see FRNM170)

8.3.10 FrNm_GetLocalNodeIdentifier

FRNM245:

Service name:	FrNm_GetLocalNodeIdentifier	
Syntax:	`Nm_ReturnType FrNm_GetLocalNodeIdentifier( const NetworkHandleType NetworkHandle, uint8* const nmNodeIdPtr )`	
Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmNodeIdPtr	Pointer the location where the node identifier of the local node shall be copied.
Return value:	Nm_ReturnType	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.	

FRNM046: If node detection is enabled, the function **`FrNm_GetLocalNodeIdentifier`** shall provide the node identifier configured for the local host node (**`FrNmNodeId`**).

FRNM268: The function FrNm_GetLocalNodeIdentifier shall be only available if the configuration parameter FrNmNodeDetectionEnabled is set to ON. (see FRNM170)

8.3.11 FrNm_RequestBusSynchronization

FRNM246:

Service name:	FrNm_RequestBusSynchronization	
Syntax:	`Nm_ReturnType FrNm_RequestBusSynchronization( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0xc0	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-Cluster
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Nm_ReturnType	NM_E_OK: No error NM_E_NOT_OK: Function failed
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.	

FRNM174: The FlexRay NM module shall support the function **`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 function is used by future extensions (e.g. Gateway) the function shall at least be available.

FRNM269: The function FrNm_RequestBusSynchronization shall be only available if the configuration parameter FrNmBusSynchronizationEnabled is set to ON.

8.3.12 FrNm_CheckRemoteSleepIndication

FRNM247:

Service name:	FrNm_CheckRemoteSleepIndication	
Syntax:	`Nm_ReturnType FrNm_CheckRemoteSleepIndication( const NetworkHandleType NetworkHandle, boolean* const nmRemoteSleepIndPtr )`	
Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (but not for the same NM-Channel)	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmRemoteSleepIndPtr	Pointer to the location where the check result of remote sleep indication shall be copied.
Return value:	Nm_ReturnType	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.	

FRNM270: The FlexRay NM module and the Nm module shall be initialized correctly before the FlexRay NM module's environment is calling the function FrNm_CheckRemoteSleepIndication.

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

FRNM271: The function FrNm_CheckRemoteSleepIndication shall be only available if the configuration parameter FrNmRemoteSleepIndicationEnabled is set to ON.

8.3.13 FrNm_GetState

FRNM248:

Service name:	FrNm_GetState	
Syntax:	`Nm_ReturnType FrNm_GetState( const NetworkHandleType NetworkHandle, Nm_StateType* const nmStatePtr, Nm_ModeType* const nmModePtr )`	
Service ID[hex]:	0x0e	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
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_ReturnType	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.	

FRNM104: The function **`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.

8.3.14 FrNm_GetVersionInfo

FRNM249:

Service name:	FrNm_GetVersionInfo	
Syntax:	`void FrNm_GetVersionInfo( Std_VersionInfoType* NmVerInfoPtr )`	
Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	NmVerInfoPtr	Pointer to the location where the version information of this module shall be copied.
Return value:	None	
Description:	Returns the version information.	

FRNM272: The function FrNm_GetVersionInfo shall return the version information of this module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers (BSW00407).

FRNM273: If source code for caller and callee of FrNm_GetVersionInfo is available, the FlexRay NM module should realize FrNm_GetVersionInfo as a macro, defined in the module's header file.

FRNM274: The function FrNm_GetVersionInfo shall be only available if the configuration parameter FrNmVersionInfoApi is set to ON.

8.3.15 FrNm_StartupError

FRNM393

Service name:	FrNm_StartupError	
Syntax:	`void FrNm_StartupError( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x10	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called by the FrSM when synchronization of the FlexRay cluster could not be achieved.	

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

8.4.1 FrNm_RxIndication

FRNM251:

Service name:	FrNm_RxIndication	
Syntax:	`void FrNm_RxIndication( PduIdType FrNmRxPduId, const PduInfoType* PduInfoPtr )`	
Service ID[hex]:	0xe1	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	FrNmRxPduId	ID of FlexRay NM PDU that has been received
	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called by the FlexRay Interface after a FlexRay NM PDU has been received.	

FRNM013: The function **`FrNm_RxIndication`** shall handle the data from the NM-message that means the function shall copy the received FlexRay NM PDU and store it locally with respect to the received FlexRay NM PDU ID.

FRNM276: The FrIf module and the FrNm module shall be initialized correctly before the FlexRay NM module's environment is calling the function FrNm_RxIndication.

The function FrNm_RxIndication might be called by the FlexRay NM module's environment in an interrupt context.

8.4.2 FrNm_TriggerTransmit

FRNM252:

Service name:	FrNm_TriggerTransmit	
Syntax:	`void FrNm_TriggerTransmit( PduIdType FrNmTxPduId, uint8* FrNmRxSduPtr )`	
Service ID[hex]:	0xe4	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	FrNmTxPduId	ID of FlexRay NM PDU that has been triggered
	FrNmRxSduPtr	Pointer to triggered FlexRay NM L-SDU data
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called by the FlexRay Interface when FlexRay NM PDU has to be transmitted.	

FRNM280: The function FrNm_TriggerTransmit shall copy the triggered FlexRay NM PDU with respect to the triggered FlexRay NM PDU ID.

FRNM277: The FrIf module and the FrNm module shall be initialized correctly before the FlexRay NM module's environment is calling the function FrNm_TriggerTransmit.

The function FrNm_TriggerTransmit might be called by the FlexRay NM module's environment in an interrupt context.

8.4.3 FrNm_ TxConfirmation

Service name:	<UL_TxConfirmation>	
Syntax:	`void <UL_TxConfirmation>( PduIdType FrIf_TxPduId )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different FrIf_TxPduId, Non reentrant for the same FrIf_TxPduId	
Parameters (in):	FrIf_TxPduId	PDU-ID of FlexRay PDU whose transmission is being confirmed
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This API service of an upper layer BSW module <UL> (e.g. PduR, FrTp, FrNm) is called by the FlexRay Interface to confirm to this upper layer BSW module that the PDU with index FrIf_TxPduId has been transmitted via the FlexRay Communication System.	

Caveats: This API service is called during the execution of the FrNm_MainFunction_<NmClstIdx>

8.5 Scheduled functions: FrNm Services provided to BSW Scheduler

8.5.1 FrNm_MainFunction_<NmClstIdx>

FRNM255:

Service name:	FrNm_MainFunction_<NmClstIdx>
Syntax:	`void FrNm_MainFunction_<NmClstIdx>( )`
Service ID[hex]:	0xf0
Timing:	FIXED_CYCLIC
Description:	Main function of FlexRay NM.

This cyclically executed API service of the FlexRay Network Management serves the purposes of maintenance of the FrNm State Machine (see 7.4). Please refer to chapter 7.2

This API service of the FlexRay Network Management is cyclically called from a task body provided by the BSW Scheduler.

Since the duration of a FlexRay Cycle may be different for two Clusters of an ECU, the calling period of this API service shall be configurable independently for each Cluster at system configuration time.

FRNM283: There shall be 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 cluster 0
- FrNm_MainFunction_1() for FlexRay NM cluster associated with FlexRay NM cluster 1
- Etc.

Note: The maximum number of independent FrNm Clusters is defined by the FrNm global definition **`FrNmNumberOfClusters`** (see chapter FrNmGlobalConstants).

8.6 Expected interfaces: Services called by the FrNm

8.6.1 Mandatory Interfaces to FrIf, Dem and FrSm

This chapter presents interfaces required to fulfill mandatory NM functionality.

FRNM256:

API function	*Description*
FrIf_Transmit	Requests the sending of a PDU.
FrIf_GetNmVector	Derives the FlexRay NM Vector.
Nm_BusSleepMode	Notification that the network management has entered Bus-Sleep Mode.
Nm_NetworkStartIndication	Notification that a NM-message has been received in the Bus-Sleep Mode, what indicates that some nodes in the network have already entered the Network Mode. The callback function shall start the network management state machine.
Dem_ReportErrorStatus	Reports errors to the DEM.
Nm_PduRxIndication	Notification that a NM message has been received.

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 [6].

8.6.2 NM_NetworkStartIndication

FRNM304:

Service name:	< Nm_NetworkStartIndication >	
Syntax:	`void < Nm_NetworkStartIndication >( const Nm_NetworkHandleType nmNetworkHandle )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different FrNm Clusters. Non reentrant for the same FrNm Cluster	
Parameters (in):	nmNetworkHandle	Identification of the NM-Cluster
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This API service of an upper layer Nm is called by the FlexRay Network Management to indicate to this upper layer BSW module that a NM-message was successfully received and the FrNm is in the Bus-Sleep Mode.	

Caveats: This API service is called during the execution of the FlexRay Network Management module (e.g. by the **`FrNm_MainFunction_<NmClstIdx>`**).

8.6.3 NM_PduRxIndication

FRNM305:

Service name:	< Nm_PduRxIndication >
Syntax:	`void < Nm_PduRxIndication >( const Nm_NetworkHandleType nmNetworkHandle )`
Service ID[hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Reentrant for different FrNm Clusters. Non reentrant for the same FrNm Cluster
Parameters (in):	nmNetworkHandle | Identification of the NM-Cluster
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This API service of an upper NM is called by the FlexRay Network Management to indicate to this upper layer BSW module Nm that a NM-message was successfully received (see FRNM190) This function is only available on the definition of _PDU_RX_INDICATION_ENABLED

Caveats: This API service is called during the execution of the FlexRay Network Management module (e.g. by the **`FrNm_MainFunction_<NmClstIdx>`**).

8.6.4 Optional Interfaces

This chapter presents interfaces required to fulfil optional NM functionality.

FRNM257:

API function	***Description***
Nm_StateChangeNotification	Notification that the state of the lower layer <BusNm> has changed.
Det_ReportError	Service to report development errors.

8.7 Configurable Interfaces

This chapter lists all interfaces where the target API service of any upper layer to be called has to be set up by static configuration of the FlexRay Network Management. These call-out services are specified and implemented in the upper layer BSW modules, which use the FlexRay Network Management according to [1]. The specific call-out notification is specified in the corresponding AUTOSAR SWS document (see chapter 0).

In addition to upper layer AUTOSAR BSW modules, the FrNm shall, with the functionality described within the specification in hand, also support other non-AUTOSAR upper layer software modules, provided that these modules interact with the FrNm in the same manner as the upper layer AUTOSAR BSW modules. In particular, those non-AUTOSAR modules need to provide APIs as described in this chapter.

8.7.1 <UL>_RemoteSleepCancellation

FRNM299:

Service name:	< UL_RemoteSleepCancellation >	
Syntax:	`void <UL_RemoteSleepCancellation>( const Nm_NetworkHandleType nmNetworkHandle )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different FrNm Clusters. Non reentrant for the same FrNm Cluster	
Parameters (in):	nmNetworkHandle	Identification of the NM-Cluster
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This API service of an upper layer BSW module <UL> (e.g. ComM) is called by the FlexRay Network Management to indicate to this upper layer BSW module that the “Remote Sleep” has been cancelled as an NM-message with an indication to keep the bus awake has been received and the Remote Sleep indication has been previously detected .	

Caveats: This API service is called during the execution of the FlexRay Network Management module (e.g. by the **`FrNm_MainFunction_<NmClstIdx>`**).

8.7.2 <UL>_RemoteSleepIndication

FRNM300:

Service name:	< UL_RemoteSleepIndication >
Syntax:	`void <UL_RemoteSleepIndication>( const Nm_NetworkHandleType nmNetworkHandle )`
Service ID[hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Reentrant for different FrNm Clusters. Non reentrant for the same FrNm Cluster
Parameters (in):	nmNetworkHandle: Identification of the NM-Cluster
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This API service of an upper layer BSW module <UL> (e.g. ComM) is called by the FlexRay Network Management to indicate to this upper layer BSW module that all other nodes in the cluster are ready to sleep (the 'Remote Sleep Indication'), as 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).

Caveats: This API service is called during the execution of the FlexRay Network Management module (e.g. by the **`FrNm_MainFunction_<NmClstIdx>`**).

8.7.3 <UL>_TransmissionTimeoutException

FRNM302:

Service name:	< UL_TransmissionTimeoutException >
Syntax:	`void <UL_TransmissionTimeoutException>( const Nm_NetworkHandleType nmNetworkHandle )`
Service ID[hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Reentrant for different FrNm Clusters. Non reentrant for the same FrNm Cluster
Parameters (in):	nmNetworkHandle: Identification of the NM-Cluster
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This API service of an upper layer BSW module <UL> is called by the FlexRay Network Management to indicate to this upper layer BSW module that no NM-message could successfully be transmitted within the time interval of FRNM_MSG_TIMEOUT_TIME, and the periodic NM-message transmission is running.

Caveats: This API service is called during the execution of the FlexRay Network Management module (e.g. by the **`FrNm_MainFunction_<NmClstIdx>`**).

Callback is available if **`FrNmMsgTimeoutTime`** >0

9 Sequence diagrams

9.1 Use Case 01 – Initialization

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

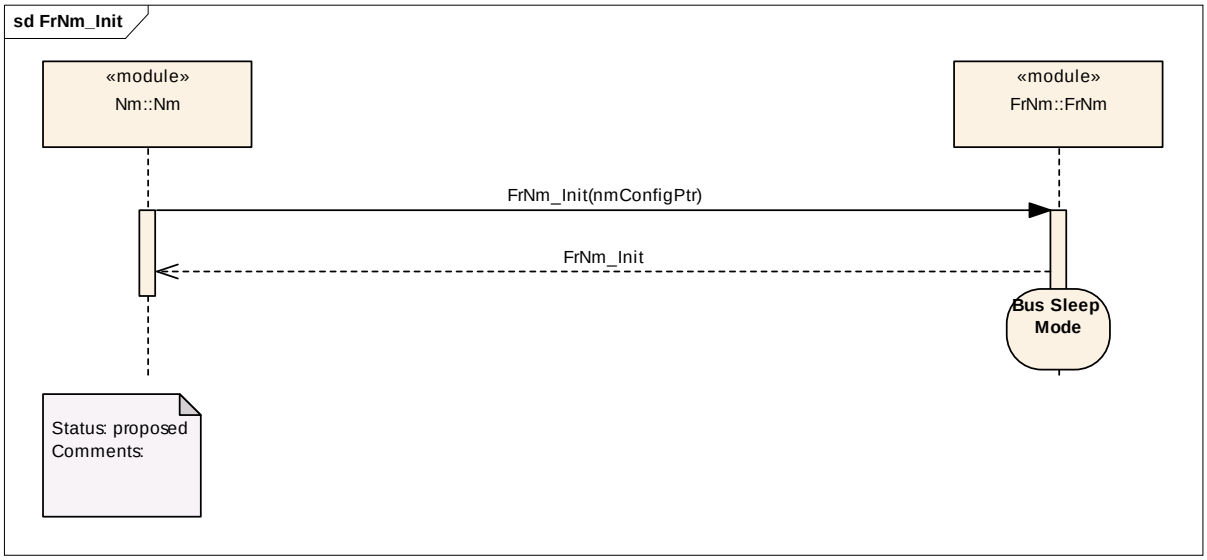

Figure 9-1 FrNm Init Sequence

9.2 Use Case 02 .- Passive Startup

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

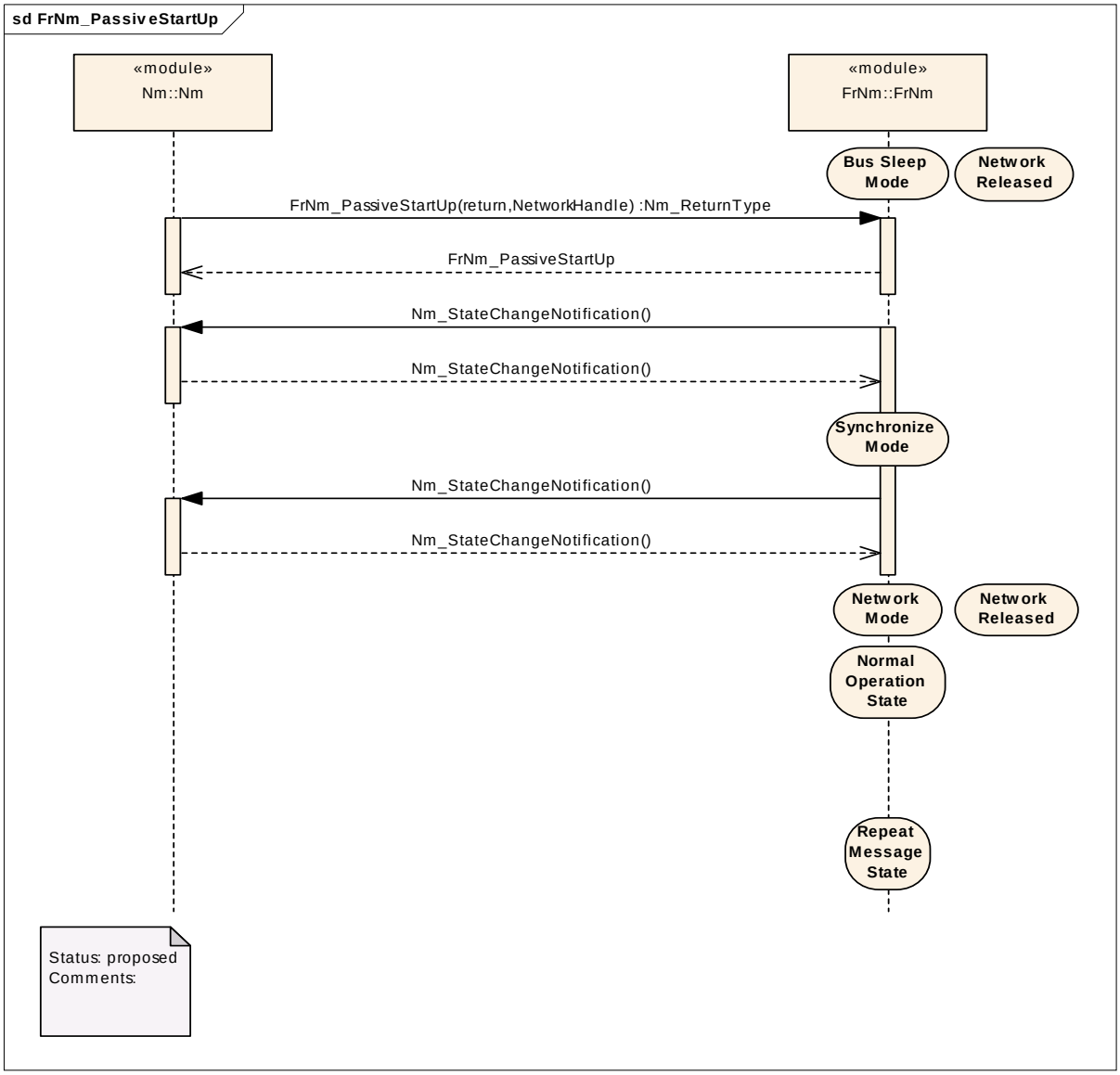

Figure 9-2 FrNm passive startup sequence

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.

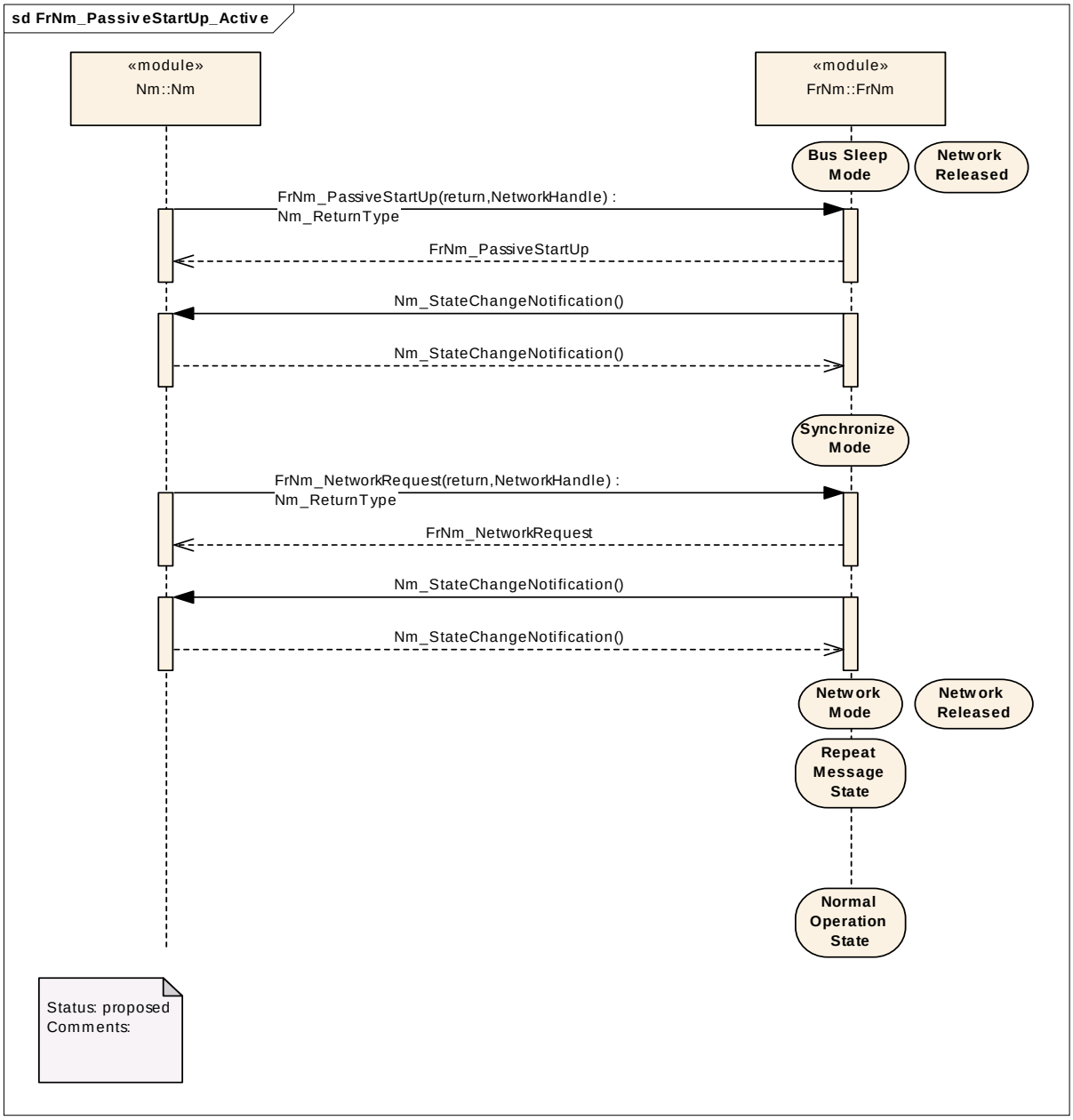

Figure 9-3 FrNm passive startup with a “active” network request sequence

9.4 Use Case 04 – Normal Operation

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

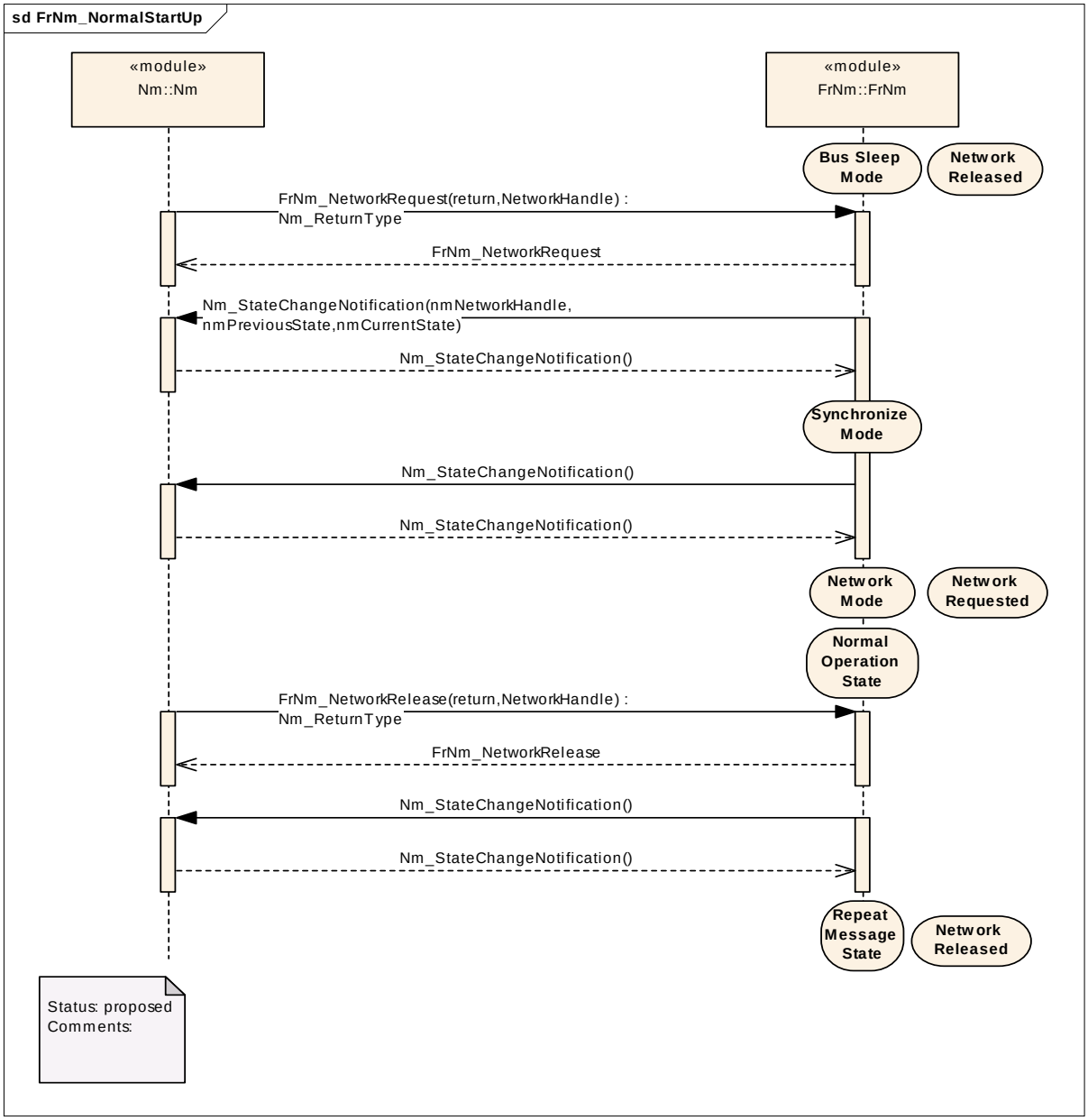

Figure 9-4 FrNm normal operation sequence

9.5 Use Case 05 – Shutdown

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

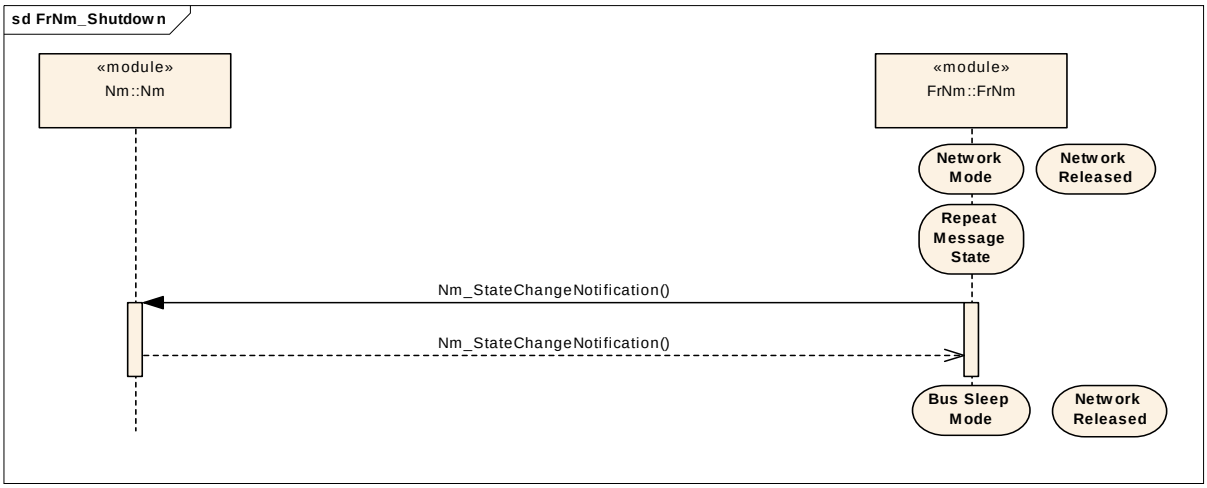

Figure 9-5 FrNm Shutdown

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 [1]
- 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

FRNM290: All configuration parameters are configurable at pre-compile time.

Use case: Source code optimizations

10.2.2 Variant 2: Pre-compile time of FrNmGlobalConfig

FRNM291: All configuration parameters of the container **`FrNmGlobalConfig`** 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 FrNmGlobalConfig; PDU configure on post build

FRNM292: The parameters contained in **`FrNm_PduConfig`** are configurable at post-build time. The parameters contained in **`FrNmGlobalConfig`** are configurable at pre-compile time

Use case: ECU configuration can be flashed (L)

10.3 Configurable parameters

10.3.1 FrNm

Module Name	*FrNm*
Module Description	The Flexray Nm module

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrNmChannelConfig	1	This container contains all configuration parameters of FlexRay NM configured from the channel perspective.
FrNmGlobalConfig	1	This container contains all global configuration parameters for the FrNm module.

10.4 Global configurable parameters

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

10.4.1 FrNmGlobalConfig

SWS Item	--
Container Name	FrNmGlobalConfig{FrNm_GlobalConfig}
Description	This container contains all global configuration parameters for the FrNm module.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrNmGlobalConstants	1	This container contains module constants related to the Flexray NM functionality.
FrNmGlobalFeatures	1	This container contains module features related to the FlexRay NM functionality.
FrNmGlobalProperties	1	This container contains module properties related to the FlexRay NM functionality.

10.4.2 FrNmGlobalConstants

SWS Item	--
Container Name	FrNmGlobalConstants{FrNm_GlobalConstants}
Description	This container contains module constants related to the Flexray NM functionality.
Configuration Parameters	

SWS Item	--	
Name	FrNmNumberOfClusters {FRNM_NUMBER_OF_CLUSTERS}	
Description	Number of AUTOSAR FR NM clusters allowed within one ECU.	
Multiplicity	1	
Type	IntegerParamDef	
Range	1 .. 255	

Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.4.3 FrNmGlobalFeatures

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

SWS Item	--		
Name	FrNmBusSynchronizationEnabled {FRNM_BUS_SYNCHRONIZATION_ENABLED}		
Description	Pre-processor switch for enabling the bus synchronisation.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--		
Name	FrNmControlBitVectorEnabled {FRNM_CONTROL_BIT_VECTOR_ENABLED}		
Description	Pre-processor switch for enabling control bit vector support. calculationFormula = If (FrNmNodeDetectionEnabled == False) then Equal(False) else Equal(False or True)		
Multiplicity	1		
Type	DerivedBooleanParamDef		
Default value	--		
calculationFormula	if (frnmnodedetectionenabled == false) then equal(false) else equal(false or true)		
calculationLanguage	informal		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--
Name	FrNmCycleCounterEmulation

	{FRNM_CYCLE_COUNTER_EMULATION}		
Description	Pre-processor switch for enabling the cycle counter emulation.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--		
Name	FrNmDualChannelPduEnable {FRNM_DUAL_CHANNEL_PDU_ENABLE}		
Description	Pre-processor switch for enabling the support of dual channel transmission and reception of NM messages.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--		
Name	FrNmHwVoteEnable {FRNM_HW_VOTE_ENABLE}		
Description	Pre-processor switch for enabling the processing of FlexRay Hardware aggregated NM-Votes.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--		
Name	FrNmNmDataDisabled {FRNM_NM_DATA_DISABLED}		
Description	Pre-processor switch for enabling the transmission of NM-Data.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--

Name	FrNmNodeDetectionEnabled {FRNM_NODE_DETECTION_ENABLED}		
Description	Pre-processor switch for enabling node detection support. calculationFormula = If (FrNmPassiveModeEnabled == False) then Equal(NmNodeDetectionEnabled) else Equal(False)		
Multiplicity	1		
Type	DerivedBooleanParamDef		
Default value	--		
calculationFormula	if (frnmpassivemodeenabled == false) then equal(nmnodedetectionenabled) else equal(false)		
calculationLanguage	informal		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--		
Name	FrNmPassiveModeEnabled {FRNM_PASSIVE_MODE_ENABLED}		
Description	Pre-processor switch for enabling Passive Mode Configuration support. calculationFormula = Equal(NmPassiveModeEnabled)		
Multiplicity	1		
Type	DerivedBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--		
Name	FrNmPduRxIndicationEnabled {FRNM_PDU_RX_INDICATION_ENABLED}		
Description	Pre-processor switch for enabling PDU reception indication.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--
Name	FrNmRemoteSleepIndicationEnabled {FRNM_REMOTE_SLEEP_INDICATION_ENABLED}

Description	Pre-processor switch for enabling remote sleep indication. calculationFormula = If (FrNmPassiveModeEnabled == True) then Equal(False) else Equal(False or True)		
Multiplicity	1		
Type	DerivedBooleanParamDef		
Default value	--		
calculationFormula	if (frnmpassivemodeenabled == true) then equal(false) else equal(false or true)		
calculationLanguage	informal		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--		
Name	FrNmRepeatMessageBitEnabled {FRNM_REPEAT_MESSAGE_BIT_ENABLED}		
Description	Pre-processor switch for enabling the repeat message bit support. calculationFormula = If (FrNmControlBitVectorEnabled == False) then Equal(False) else Equal(False or True)		
Multiplicity	1		
Type	DerivedBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--		
Name	FrNmSourceNodeIdentifierEnabled {FRNM_SOURCE_NODE_INDENTIFIER_ENABLED}		
Description	Pre-processor switch for enabling SourceNodeIdentifier support.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--
Name	FrNmStateChangeIndicationEnabled {FRNM_STATE_CHANGE_INDICATION_ENABLED}
Description	Pre-processor switch for enabling state change indication.
Multiplicity	1

Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	--		
Name	FrNmUserDataEnabled {FRNM_USER_DATA_ENABLED}		
Description	Pre-processor switch for enabling user data support.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.4.4 FrNmGlobalProperties

SWS Item	**--**
Container Name	FrNmGlobalProperties{FrNm_GlobalProperties}
Description	This container contains module properties related to the FlexRay NM functionality.
Configuration Parameters	

SWS Item	**--**		
Name	FrNmDevErrorDetect {FRNM_DEV_ERROR_DETECT}		
Description	Pre-processor switch for enabling development error detection		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	FrNmMainAcrossFrCycle {FRNM_MAIN_ACROSS_FR_CYCLE}		
Description	Parameter describing if the execution of FrNm_Main function crosses the FlexRay cycle boundary or not.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	FrNmVersionInfoApi {FRNM_VERSION_INFO_API}		
Description	Pre-processor switch for enabling version info API support.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.5 Channel configurable parameters

FRNM018: The Channel Scope (FrNm_ChannelConfig) specifies configuration parameters that shall be located in a data structure of type **`FrNm_ConfigType`**.

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

10.5.1 FrNm

Module Name	*FrNm*
Module Description	The Flexray Nm module

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrNmChannelConfig	1	This container contains all configuration parameters of FlexRay NM configured from the channel perspective.
FrNmGlobalConfig	1	This container contains all global configuration parameters for the FrNm module.

10.5.2 FrNmChannelConfig

SWS Item	**--**
Container Name	FrNmChannelConfig{FrNm_ChannelConfigType} [Multi Config Container]
Description	This container contains all configuration parameters of FlexRay NM configured from the channel perspective.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrNmChannelIdentifiers	1	This container contains instance specific identifiers related to the respective FlexRay Channel.
FrNmChannelTiming	1	This container contains instance-specific timing related to the respective FlexRay Channel.

10.5.3 FrNmChannelIdentifiers

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

SWS Item	**--**		
Name	FrNmDataEnabled {FRNM_NM_DATA_ENABLED}		
Description	Enable the separated sending of NM-Data.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-

			BUILD
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	FrNmInstanceId {FRNM_INSTANCE_ID}		
Description	FlexRay NM instance identifier configured for the respective FlexRay Channel. It is used for reporting of development errors to DET. It must be unique for each NM instance within one ECU.		
Multiplicity	1		
Type	IntegerParamDef		
Range	8192 .. 8446		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	FrNmNodeId {FRNM_NODE_ID}		
Description	NM node identifier configured for the respective FlexRay Channel. 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.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	FrNmPduLength {FRNM_PDU_LENGTH}		
Description	Length of the NM-Data PDU.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 8		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	FrNmChannelHandle {FRNM_CHANNEL_HANDLE}		
Description	Channel identifier configured for the respective instance of the NM. The FrNmChannelHandle shall be encoded in the FrNmRxPduId parameter which is passed to FrNm_RxIndication() function called by the FrIf.		
Multiplicity	1		
Type	Reference to FrIfCluster		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE

	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	FrNmChannelIdRef {FRNM_CHANNEL_ID}		
Description	NM-Network identifier configured for the respective FlexRay Channel. It is used for referring to the respective NM-Network handle.		
Multiplicity	1		
Type	Reference to NmChannelConfig		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Instance dependency: It must be unique for each NM instance within one ECU.		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrNmRxPdu	1..2	This container describes the FlexRay NM RX PDU:s.
FrNmTxPdu	0..2	This container describes the FlexRay NM TX PDU:s.

10.5.4 FrNmChannelTiming

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

SWS Item	**--**		
Name	FrNmDataCycle {FRNM_DATA_CYCLE}		
Description	Number of FlexRay Schedule Cycles needed to transmit the NM Data of all ECUs on the FlexRay bus		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FRNM_CYCLE_VALUE_1		
	FRNM_CYCLE_VALUE_16		
	FRNM_CYCLE_VALUE_2		
	FRNM_CYCLE_VALUE_32		
	FRNM_CYCLE_VALUE_4		
	FRNM_CYCLE_VALUE_64		
	FRNM_CYCLE_VALUE_8		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**
Name	FrNmMsgTimeoutTime {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.

Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	FrNmReadySleepCnt {FRNM_READY_SLEEP_CNT}		
Description	Numbers of repetitions in the ready sleep state before NM switches to bus sleep mode. 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.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	dependency: Condition: FrNmReadySleepCnt >= 1		

SWS Item	**--**		
Name	FrNmRemoteSleepIndTime {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. The value "0" denotes that no Remote Sleep Indication functionality is configured.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	dependency: Condition: FrNmRemoteSleepIndTime >= FrNmRepetitionCycle or FrNmRemoteSleepIndTime = 0		

SWS Item	**--**		
Name	FrNmRepeatMessageTime {FRNM_REPEAT_MESSAGE_TIME}		
Description	Timeout for Repeat Message State. Defines the time in seconds how long the NM shall stay in the Repeat Message State. 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.		
Multiplicity	1		
Type	FloatParamDef		
Range	0.0 .. 65.535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE

	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	FrNmRepetitionCycle {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.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FRNM_CYCLE_VALUE_1		
	FRNM_CYCLE_VALUE_16		
	FRNM_CYCLE_VALUE_2		
	FRNM_CYCLE_VALUE_32		
	FRNM_CYCLE_VALUE_4		
	FRNM_CYCLE_VALUE_64		
	FRNM_CYCLE_VALUE_8		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	dependency: Condition: FrNmRepetitionCycle = n * FrNmVotingCycle; n = [1,2,4,8,16,32,64]		

SWS Item	**--**		
Name	FrNmVotingCycle {FRNM_VOTING_CYCLE}		
Description	Number of FlexRay Schedule Cycles needed to transmit the Nm vote of all ECUs on the FlexRay Bus.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FRNM_CYCLE_VALUE_1		
	FRNM_CYCLE_VALUE_16		
	FRNM_CYCLE_VALUE_2		
	FRNM_CYCLE_VALUE_32		
	FRNM_CYCLE_VALUE_4		
	FRNM_CYCLE_VALUE_64		
	FRNM_CYCLE_VALUE_8		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

No Included Containers

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

10.5.5 FrNmRxPdu

SWS Item	**--**		
Container Name	FrNmRxPdu		
Description	This container describes the FlexRay NM RX PDU:s.		
Configuration Parameters			

SWS Item	**--**		
Name	FrNmRxPduContainsData {FRNM_RX_PDU_CONTAINS_DATA}		
Description	This parameted defines if the PDU contains NM Data.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	FrNmRxPduContainsVote {FRNM_RX_PDU_CONTAINS_VOTE}		
Description	This parameted defines if the PDU contains NM Vote information.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	FrNmRxPduId {FRNM_RX_PDU_ID}		
Description	PDU identifier configured for the respective FlexRay Channel. It is used for referring to the FlexRay Interface receive function. It must be consistent with the value configured in the FlexRay Interface. This ID is used for the combined reception of NM Vote and NM Data or for the reception of the NM Vote if NM Data is received in a separate PDU. ImplementationType: PduIdType		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65536		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	FrNmRxPduRef		
Description	The reference to a PDU in the global PDU structure described in the AUTOSAR ECU Configuration Specification. This reference will be used by the FrIf module to derive the PDU Id.		
Multiplicity	1		
Type	Reference to Pdu		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE

	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.5.6 FrNmTxPdu

SWS Item	**--**
Container Name	FrNmTxPdu
Description	This container describes the FlexRay NM TX PDU:s.
Configuration Parameters	

SWS Item	**--**		
Name	FrNmTxPduContainsData {FRNM_TX_PDU_CONTAINS_DATA}		
Description	This parameted defines if the PDU contains NM Data.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	FrNmTxPduContainsVote {FRNM_TX_PDU_CONTAINS_VOTE}		
Description	This parameted defines if the PDU contains NM Vote information.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	FrNmTxPduRef		
Description	The reference to a PDU in the global PDU structure described in the AUTOSAR ECU Configuration Specification. This reference is used to derive the PDU Id that is defined by the FrIf module.		
Multiplicity	1		
Type	Reference to Pdu		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.6 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.

10.7 Configuration constraints

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)
- FrNmReadySleepCnt = 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)

10.8 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 ([4]) contain the necessary information.

10.8.1 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).

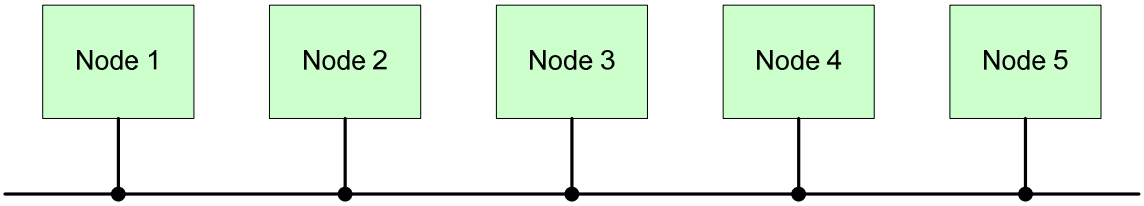

Figure 10-1 Example of five Node Network

The FlexRay Schedule allots 5 slots in the static segment and 6 slots in the dynamic segment as shown in Figure 10-2 (below). 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.

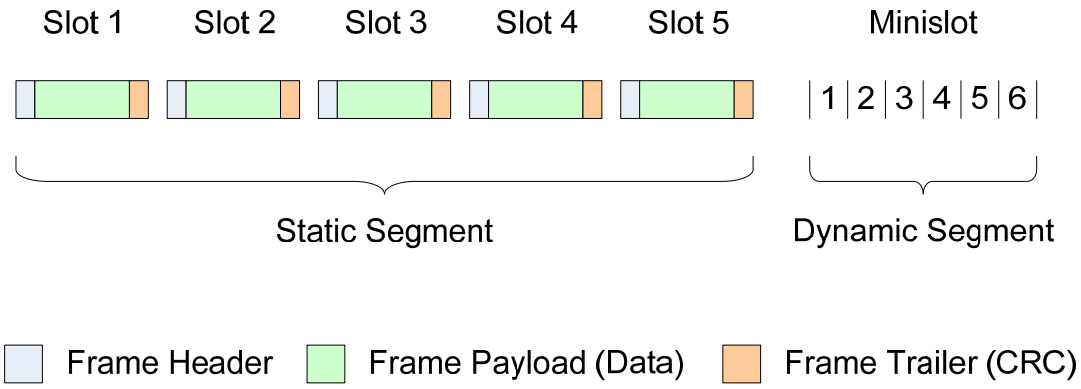

Figure 10-2 Example of Bus Schedule

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 96).

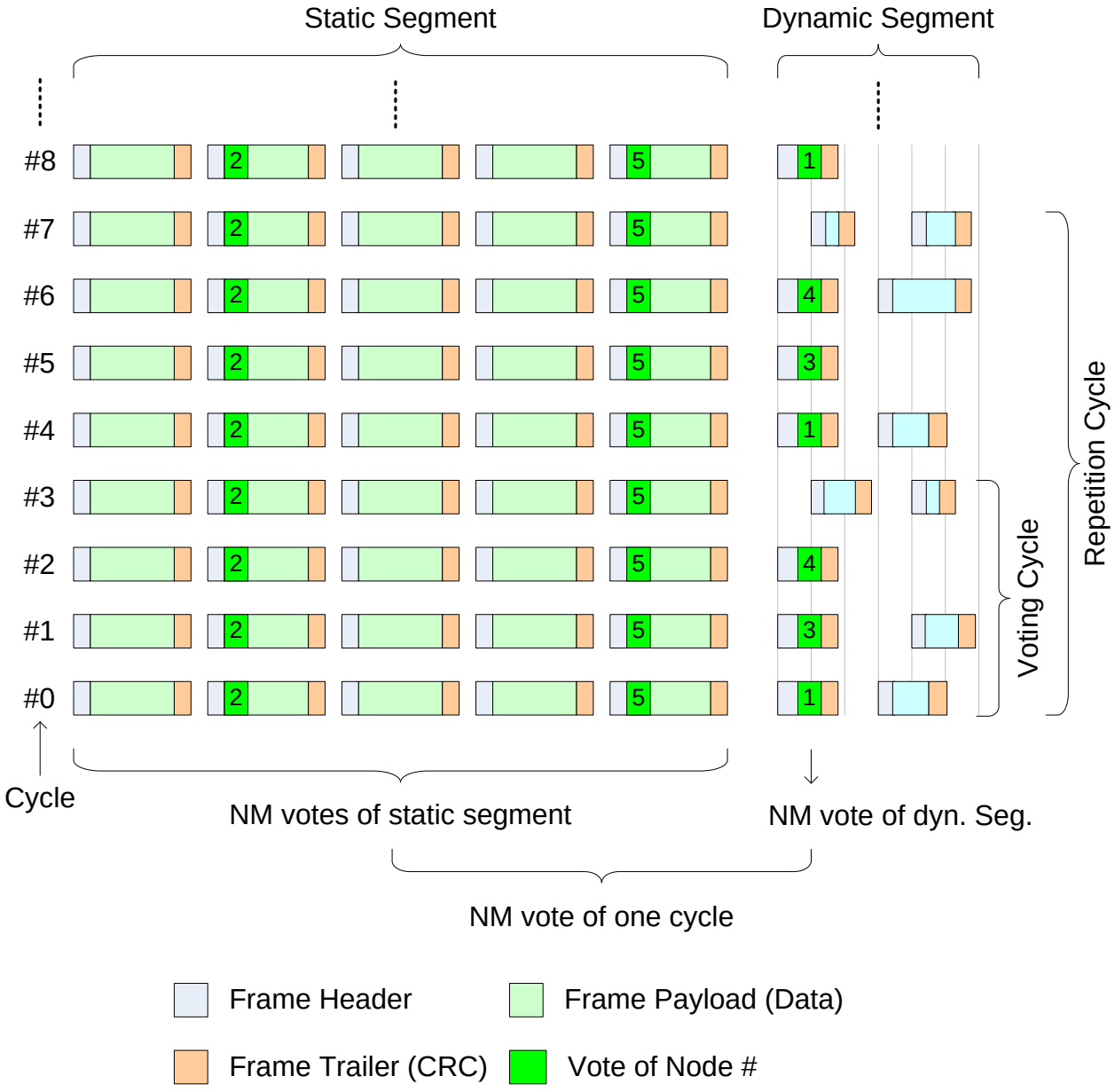

Figure 10-3 Example of NM-Vote in dynamic and static segment

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.

10.8.2 Example of Bus Schedule with NM-Data PDUs

This example uses the same setup as in the previous example (10.8.1) - five nodes (Figure 10-1 on page 95), 5 slot in the static segment and 6 slots in the dynamic segment (Figure 10-2 on page 95).

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.

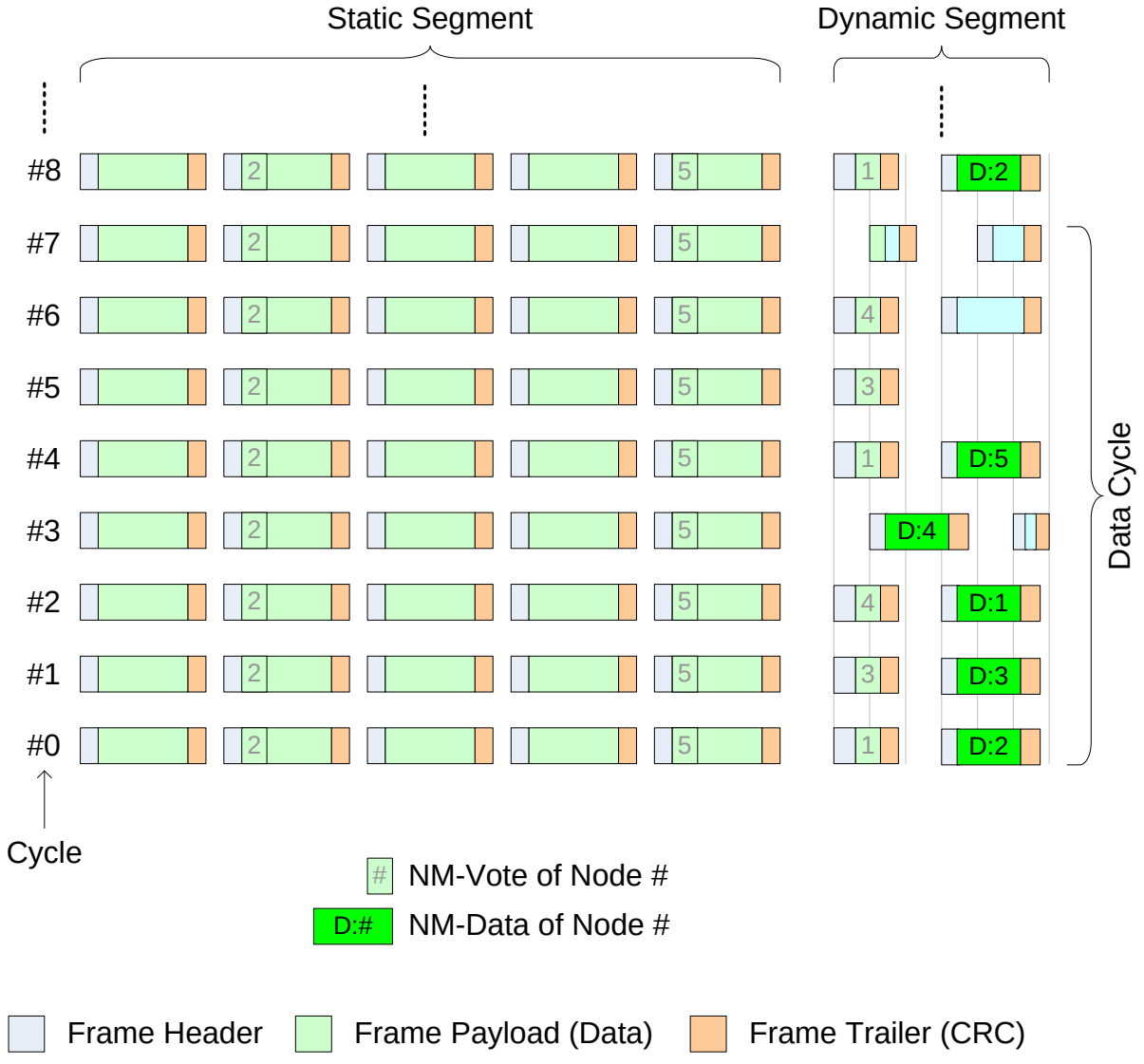

Figure 10-4 Example of Bus Schedule with NM-Data

11 Changes to Release 2.0.0

11.1 Deleted SWS Items

SWS Item	*Rationale*
FRNM165	Removed due to duplicated requirement with FRNM163
FRNM068	Removed as no required information was covered
FRNM003	Removed due to issue #18325 – the FrIf is responsible for the schedule of the FlexRay.
FRNM253, FRNM281, FRNM282 and FRNM278	Removed due to issue #19797 – the function is not used, and will not be called by the FrIf
FRNM002	Not needed – The FrIf is responsible for the FlexRay Schedule
FRNM201	Not needed – The FrIf is responsible for the FlexRay Schedule
FRNM150	Not needed – The FrIf is responsible for the FlexRay Schedule
FRNM075	Duplicate of better formulated FRNM225
FRNM288, FRNM289	"Duplicate" of FRNM024
FRNM158, FRNM200, FRNM202, FRNM203	Not needed – The FrIf is responsible for the FlexRay Schedule
FRNM060, FRNM061, FRNM146	Legacy: not nedded anymore
FRNM197, FRNM199	Not needed – The FrIf is responsible for the FlexRay Schedule

11.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*

11.3 Changed SWS Items

SWS Item	*Rationale*
FRNM119	Refined the requirement
FRNM120	Refined the requirement
FRNM143	Refined the requirement
FRNM130, FRNM131	Refined the requirement
FRNM132	Refined the requirement

11.4 Added SWS Items

SWS Item	*Rationale*
FRNM297	Gave ID to existing requirement of UL_NetworkStartIndication
FRNM298	Gave ID to existing requirement of UL_PduRxIndication
FRNM299	Gave ID to existing requirement of UL_RemoteSleepCancellation
FRNM300	Gave ID to existing requirement of UL_RemoteSleepIndication
FRNM301	Gave ID to existing requirement of UL_StateChangeNotification
FRNM302	Gave ID to existing requirement of UL_TransmissionTimeoutException
FRNM309	Introduced type definition of FrNm_ReadySleepCnt

12 Changes during SWS Improvements by Technical Office

12.1 Deleted SWS Items

SWS Item	*Rationale*
FRNM001	Requirement on the Generic NM module and not on the FrNm module
FRNM015	Requirement on the FrIf module and not on the FrNm module
FRNM012	Requirement on the FrIf module and not on the FrNm module
FRNM011	Requirement on the FrIf module and not on the FrNm module
FRNM221	Requirement on other module (COM Manager)
FRNM254	Removed FrNm_NmVectorIndication in accordance to # 16607
FRNM279	Removed FrNm_NmVectorIndication in accordance to # 16607
FRNM297	Removed Nm_NetworkStartIndication, moved it to the mandatory interfaces
FRNM298	Removed Nm_PduRxIndication, moved it to the mandatory interfaces
FRNM301	Removed Nm_StateChangeNotification, moved it to the mandatory interfaces

12.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
FRNM025	FRNM284, FRNM285	Made requirement atomic.
FRNM026	FRNM286, FRNM287	Made requirement atomic.
FRNM027	FRNM288, FRNM289	Made requirement atomic.

12.3 Changed SWS Item

Many requirements have been changed to improve understandability without changing the technical contents.

12.4 Added SWS Items

SWS Item	*Rationale*
FRNM235	UML model linking of the imported types
FRNM236	UML model linking of the function FrNm_Init
FRNM237	UML model linking of the function FrNm_PassiveStartUp
FRNM238	UML model linking of the function FrNm_NetworkRequest
FRNM239	UML model linking of the function FrNm_NetworkRelease
FRNM240	UML model linking of the function FrNm_SetUserData
FRNM241	UML model linking of the function FrNm_GetUserData
FRNM242	UML model linking of the function Nm_GetPduData
FRNM243	UML model linking of the function FrNm_RepeatMessageRequest
FRNM244	UML model linking of the function FrNm_GetNodeIdentifier
FRNM245	UML model linking of the function FrNm_GetLocalNodeIdentifier
FRNM246	UML model linking of the function FrNm_RequestBusSynchronization
FRNM247	UML model linking of the function FrNm_CheckRemoteSleepIndication
FRNM248	UML model linking of the function FrNm_GetState
FRNM249	UML model linking of the function FrNm_GetVersionInfo
FRNM250	UML model linking of the function FrNm_TxConfirmation
FRNM251	UML model linking of the function FrNm_RxIndication

FRNM252	UML model linking of the function FrNm_TriggerTransmit
FRNM253	UML model linking of the function FrNm_CycleStartIndication
FRNM254	UML model linking of the function FrNm_NmVectorIndication
FRNM255	UML model linking of the function FrNm_MainFunction_<NmClstIdx>
FRNM256	UML model linking of the mandatory interfaces
FRNM257	UML model linking of the optional interfaces
FRNM258	Gave ID to existing requirement of FrNm_PassiveStartUp
FRNM260	Gave ID to existing requirement of FrNm_PassiveStartUp
FRNM261	Gave ID to existing requirement of FrNM_NetworkRequest
FRNM262	Gave ID to existing requirement of FrNM_NetworkRelease
FRNM263	Gave ID to existing requirement of FrNm_SetUserData
FRNM264	Gave ID to existing requirement of FrNm_GetUserData
FRNM265	Gave ID to existing requirement of Nm_GetPduData
FRNM266	Gave ID to existing requirement of Nm_GetPduData
FRNM267	Gave ID to existing requirement of FrNm_GetNodeIdentifier
FRNM268	Gave ID to existing requirement of FrNm_GetLocalNodeIdentifier
FRNM269	Gave ID to existing requirement of FrNm_RequestBusSynchronization
FRNM270	Gave ID to existing requirement of FrNm_CheckRemoteSleepIndication
FRNM271	Gave ID to existing requirement of FrNm_CheckRemoteSleepIndication
FRNM272	Gave ID to existing requirement of FrNm_GetVersionInfo
FRNM273	Gave ID to existing requirement of FrNm_GetVersionInfo
FRNM274	Gave ID to existing requirement of FrNm_GetVersionInfo
FRNM275	Gave ID to existing requirement of FrNm_TxConfirmation
FRNM276	Gave ID to existing requirement of FrNm_RxIndication
FRNM277	Gave ID to existing requirement of FrNm_TriggerTransmit
FRNM278	Gave ID to existing requirement of FrNm_CycleStartIndication
FRNM279	Gave ID to existing requirement of FrNm_NmVectorIndication
FRNM280	Gave ID to existing requirement of FrNm_TriggerTransmit
FRNM281	Gave ID to existing requirement of FrNm_CycleStartIndication
FRNM282	Gave ID to existing requirement of FrNm_CycleStartIndication
FRNM283	Gave ID to existing requirement of FrNm_MainFunction_<NmClstIdx>
FRNM290	Every variant gets a requirement ID
FRNM291	Every variant gets a requirement ID
FRNM292	Every variant gets a requirement ID
FRNM293	Standard requirement for Error classification
FRNM294	Standard requirement for Error classification
FRNM295	Standard requirement for Error detection