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

| | |
|---|---|
| **Document Version** | 4.2.0 |
| **Document Status** | Final |
| **Part of Release** | 4.0 |
| **Revision** | 3 |

## Document Change History

| Date | Version | Changed by | Change Description |
|---|---|---|---|
| 13.12.2011 | 4.2.0 | AUTOSAR Administration | • Support of a coordinated shutdown if more than one gateway coordinator is connected to the same network<br>• Support of CarWakeup in NM user data<br>• Extension for Partial Network |
| 11.10.2010 | 4.1.0 | AUTOSAR Administration | • Added FrNM066, FrNM220, FRNM395, FRNM387, FRNM388, FRNM389, FRNM390, FRNM391, FRNM392<br>• Update FRNM235, FRNM254<br>• Modified FRNM074,FRNM021 FRNM272, FRNM074, FRNM135, NM192, FRNM154, FRNM155, FRNM324, FRNM3829, FRNM394, FRNM181, FRNM229, FRNM066, FRNM359, FRNM106, FRNM035, FRNM257 |
| 11.12.2009 | 4.0.0 | AUTOSAR Administration | • Improved configurable handling of bus faults to support a smooth transition to sleep or a resynchronisation of the network to ensure its coherency.<br>• Multiple variants of network coordination support dual channel and complex FlexRay networks<br>• Relaxed timing constraints of the FrNm main function<br>• Legal disclaimer revised |
| 02.02.2009 | 3.0.3 | AUTOSAR Administration | Layout adaptations |

# Document Change History

| Date | Version | Changed by | Change Description |
|---|---|---|---|
| 30.01.2009 | 3.0.2 | AUTOSAR Administration | • Incorporation of core partner change requests for R3.0<br>• Legal disclaimer revised |
| 23.01.2008 | 3.0.1 | AUTOSAR Administration | Updated chapter 8/10 after changes in BSW UML model |
| 04.12.2007 | 3.0.0 | AUTOSAR Administration | • FlexRay NM machine has been reworked completely<br>• Support of Hardware NM Vector of communication controller<br>• Separation of NM vote and data (transmission of NM vote and data can be done with different update intervals)<br>• FlexRay NM State machine is now synchronised to FlexRay communication cycle<br>• Document meta information extended<br>• Small layout adaptations made<br>• Added FRNM311 |
| 31.01.2007 | 2.0.0 | AUTOSAR Administration | • FlexRay NM machine has been reworked completely<br>• Support of Hardware NM Vector of communication controller<br>• Separation of NM vote and data (transmission of NM vote and data can be done with different update intervals)<br>• FlexRay NM State machine is now synchronised to FlexRay communication cycle<br>• Legal disclaimer revised<br>• Release Notes added<br>• “Advice for users” revised<br>• “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

# Known Limitations

In Release 4.0.4 the FrNmReadySleepCnt will be derived from a time based parameter.
When ECU Extract is transformed into EcuC the FrNmReadySleepCnt is calculated based on the ReadySleepTime in the system template.
ReadySleepCount will be removed/deprecated from system template and replaced by the time based parameter ReadySleepTime which defines the duration until bus shuts down.
FrNmReadySleepCnt is still necessary for the statemachine because the FrNm mechanism is based on RepetitionCycles.

# 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. Additionally, the DEM provides fault information to the DCM (e.g., read all stored DTCs from the error memory). |
| **SRS** | Software requirements specification |
| **SWS** | Software working specification |
| **API** | Application program interface |
| **Com** | Communication module |
| **OS** | Operating system |
| **SchM** | Schedule Manager |
| **PDU** | Protocol data unit |
| **CPU** | Central processing unit |

| Abbreviation: | Description |
|---|---|
| **FrIf** | Abbreviation for the FlexRay Interface |
| **FrNm** | Abbreviation for the Network Management on FlexRay |
| **Nm** | Abbreviation for the generic Network Management |
| **ECU** | Electronic control unit |
| **ComM** | Communication manager |
| **FrSm** | FlexRay State Manager |
| **HW** | Hardware |

| 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.<br>Caution: The FlexRay Bus contains two FlexRay channels which cannot be handled independent of the other. Therefore the **NM-Network** covers both FlexRay bus channels. This is 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. This is used to improve the reliability of the voting. |
| **NM Slot** | Slot reserved for purposes of network management. |
| **NM Timeout** | Timeout in the NM algorithm that initiates transition into Bus-Sleep Mode. |

| | |
|---|---|
| **NM User Data** | Supplementary application specific data that is sent independent of the NM Vote on the bus. |
| **NM Voting Cycle** | Number of FlexRay cycles necessary for all nodes 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-Vector** | FlexRay NM Vector is the aggregated data available when the FlexRay CC optional NM Hardware Vector Service is used. |
| **NM-Data** | Data related to NM transmitted using the FlexRay Bus. |
| **NM-Cluster** | Obsolete, equivalent to NM-Network |
| **CBV** | Control bit vector |
| **ClusterAwake Vote** | At least one Node other than itself votes for keeping the cluster awake. |

# 3 Related documenation

## 3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Sofware Architecture,
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[4] Requirements on Network Management
AUTOSAR_SRS_NetworkManagement.pdf

## 3.2 Related standards and norms

[5] FlexRay Communications System Specifications, V2.1

## 3.3 Related AUTOSAR documents

[6] AUTOSAR Specification of Communication Manager
AUTOSAR_SWS_COMManager.pdf

[7] Specification of Generic Network Management Interface
AUTOSAR_SWS_NetworkManagement.pdf

[8] Specification of FlexRay Interface
AUTOSAR_SWS _FlexRayInterface. pdf

[9] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

[10] Specification of Diagnostic Event Manager
AUTOSAR_SWS_DiagnosticEventManager.pdf

[11] Specification of Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[12] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[13] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes.pdf

[14] Specification of Compiler Abstraction

AUTOSAR_SWS_CompilerAbstraction.pdf

[15] Specification of Operating System
AUTOSAR_SWS_OS.pdf

[16] Specification of FlexRay State Manager
AUTOSAR_SWS_FlexRayStateManager.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 FlexRay channels of the same FlexRay Bus at the same time.

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

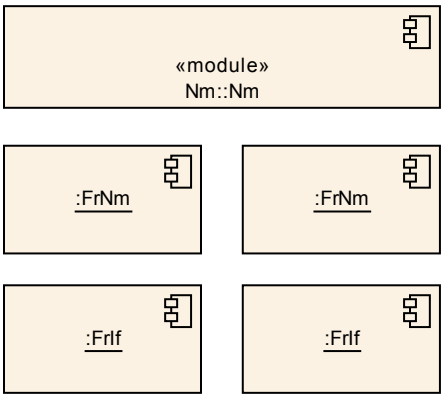



Figure 4-1 AUTOSAR NM Stack on FlexRay

## 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 Generic Network Management Interface (Nm) and uses services of FlexRay Interface

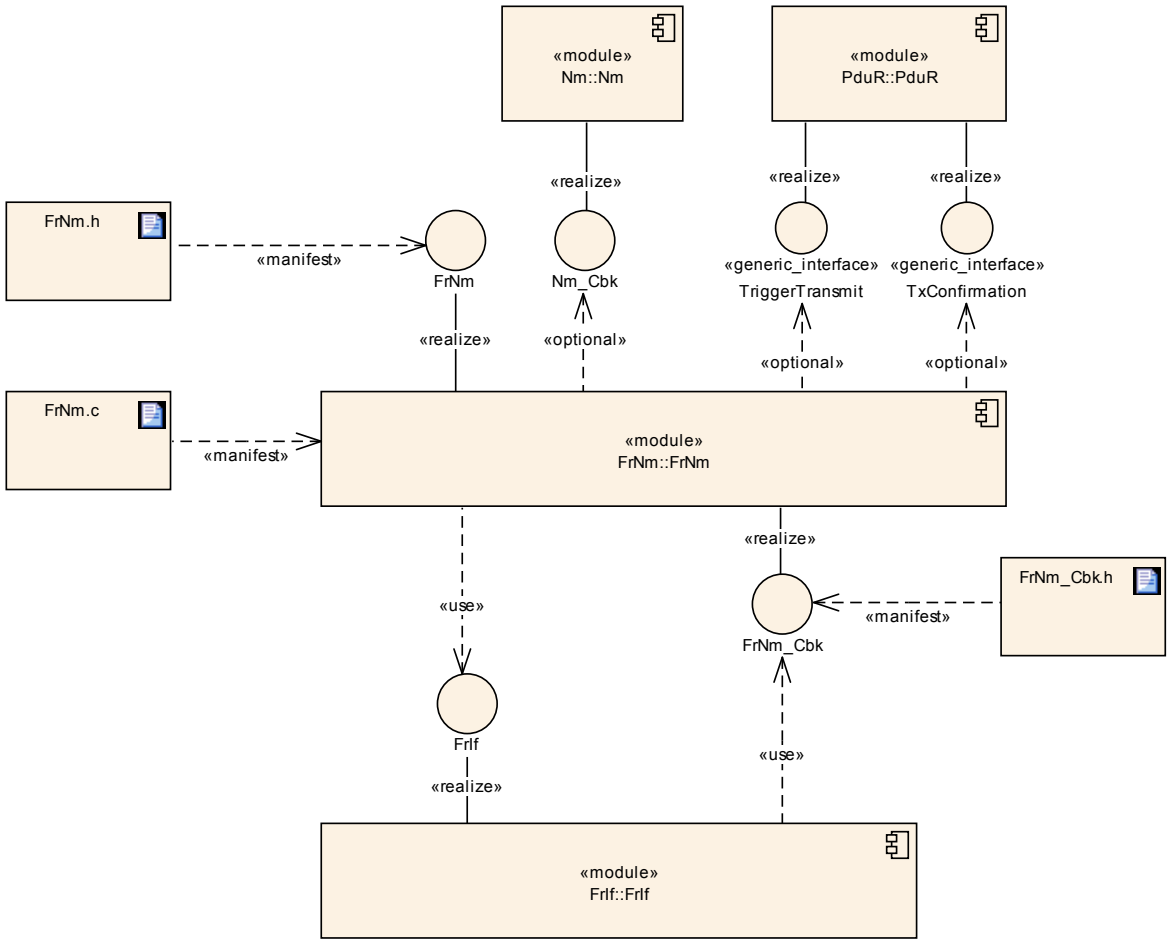



Figure 5-1 NM Overview

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 [15]⌋ (BSW00429)

| ***Module*** | ***Dependencies*** |
|---|---|
| Nm | FlexRay NM provides services to the Generic Network Management Interface (Nm) |
| PduR | FlexRay NM uses services of the PDU Router |
| FrIf | FlexRay NM uses services of the FlexRay Interface. |

| | |
|---|---|
| Det | The FlexRay Network Management informs the Development Error Tracer on development errors |
| Dem | Dem gets production error information from FlexRay Network Management. |
| EcuM | EcuM gets wake up event information from FlexRay Network Management if supported by hardware. |
| RTE | The FlexRay Network Management main function may be scheduled by the by the RTE. |

## 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) ⌋(BSW00380)

### 5.1.2 Header file structure

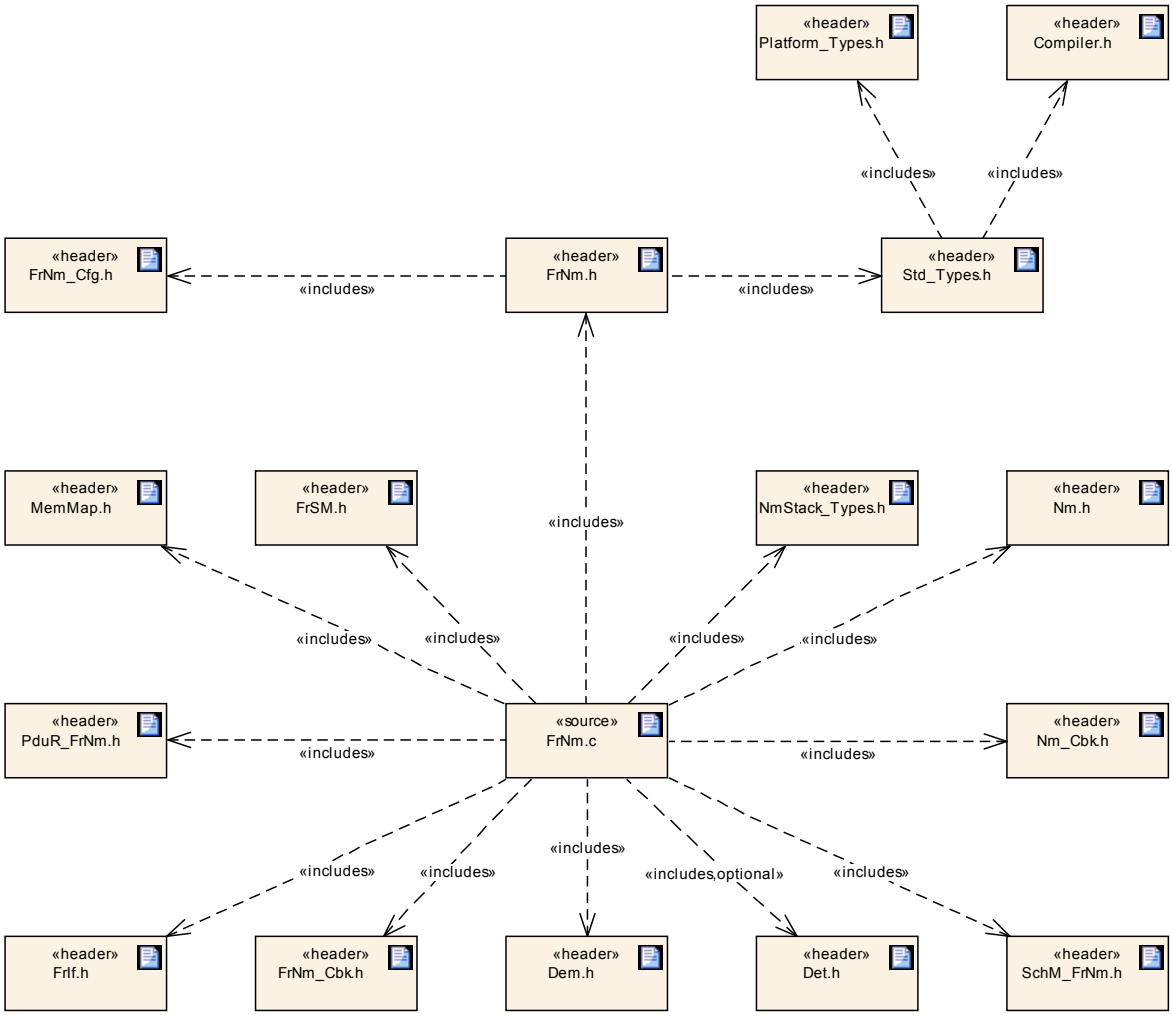



Figure 5-2 Header File Structure

**[FRNM065]** ⌈The following header files shall be provided and included within the FrNm module.
1. FrNm.h
2. FrNm_Cbk.h
3. FrNm_Cfg.h⌋(BSW00381, BSW00412, BSW00370)

**[FRNM367]** ⌈The file FrNm.h contains declaration of provided interface functions.⌋( )

**[FRNM368]** ⌈The file FrNm_Cbk.h contains declaration of provided call-back functions.⌋( )

**[FRNM369]** ⌈The file FrNm_Cfg.h contains pre-compile time parameters.⌋( )

**[FRNM066]** ⌈The following header files shall be included within the FrNm module.
Std_Types.h (for AUTOSAR standard types),

FrIf.h (for interface of FlexRay Interface),
PduR_FrNm.h (for interface of PduR),
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 (for locating software in memory using the memory map),
FrSM.h (for FlexRay State Manager)」( )

**[FRNM371]** 「Std_Types.h shall include
Platform_Types.h (for platform specific types),
Compiler.h (for compiler specific language extensions).」( )

# 6 Requirements traceability

| Requirement | Satisfied by |
|---|---|
| - | FRNM177 |
| - | FRNM160 |
| - | FRNM137 |
| - | FRNM280 |
| - | FRNM345 |
| - | FRNM372 |
| - | FRNM190 |
| - | FRNM273 |
| - | FRNM357 |
| - | FRNM342 |
| - | FRNM174 |
| - | FRNM124 |
| - | FRNM256 |
| - | FRNM408 |
| - | FRNM336 |
| - | FRNM128 |
| - | FRNM283 |
| - | FRNM076 |
| - | FRNM265 |
| - | FRNM379 |
| - | FRNM169 |
| - | FRNM258 |
| - | FRNM154 |
| - | FRNM391 |
| - | FRNM218 |
| - | FRNM429 |
| - | FRNM010 |
| - | FRNM141 |
| - | FRNM396 |
| - | FRNM048 |
| - | FRNM317 |
| - | FRNM409 |
| - | FRNM055 |
| - | FRNM271 |
| - | FRNM228 |
| - | FRNM205 |

| | |
|---|---|
| - | FRNM107 |
| - | FRNM246 |
| - | FRNM129 |
| - | FRNM264 |
| - | FRNM375 |
| - | FRNM058 |
| - | FRNM424 |
| - | FRNM162 |
| - | FRNM356 |
| - | FRNM311 |
| - | FRNM432 |
| - | FRNM434 |
| - | FRNM416 |
| - | FRNM369 |
| - | FRNM120 |
| - | FRNM248 |
| - | FRNM346 |
| - | FRNM359 |
| - | FRNM324 |
| - | FRNM172 |
| - | FRNM235 |
| - | FRNM249 |
| - | FRNM239 |
| - | FRNM105 |
| - | FRNM277 |
| - | FRNM437 |
| - | FRNM255 |
| - | FRNM111 |
| - | FRNM428 |
| - | FRNM113 |
| - | FRNM170 |
| - | FRNM046 |
| - | FRNM131 |
| - | FRNM050 |
| - | FRNM319 |
| - | FRNM066 |
| - | FRNM368 |
| - | FRNM257 |
| - | FRNM390 |

| | |
|---|---|
| - | FRNM127 |
| - | FRNM233 |
| - | FRNM418 |
| - | FRNM229 |
| - | FRNM387 |
| - | FRNM383 |
| - | FRNM006 |
| - | FRNM194 |
| - | FRNM404 |
| - | FRNM148 |
| - | FRNM422 |
| - | FRNM413 |
| - | FRNM316 |
| - | FRNM007 |
| - | FRNM405 |
| - | FRNM138 |
| - | FRNM042 |
| - | FRNM395 |
| - | FRNM261 |
| - | FRNM348 |
| - | FRNM363 |
| - | FRNM143 |
| - | FRNM115 |
| - | FRNM269 |
| - | FRNM378 |
| - | FRNM276 |
| - | FRNM309 |
| - | FRNM189 |
| - | FRNM364 |
| - | FRNM307 |
| - | FRNM147 |
| - | FRNM032 |
| - | FRNM421 |
| - | FRNM123 |
| - | FRNM193 |
| - | FRNM272 |
| - | FRNM361 |
| - | FRNM274 |
| - | FRNM118 |

| | |
|---|---|
| - | FRNM132 |
| - | FRNM142 |
| - | FRNM191 |
| - | FRNM425 |
| - | FRNM134 |
| - | FRNM121 |
| - | FRNM215 |
| - | FRNM295 |
| - | FRNM051 |
| - | FRNM294 |
| - | FRNM433 |
| - | FRNM397 |
| - | FRNM188 |
| - | FRNM338 |
| - | FRNM230 |
| - | FRNM407 |
| - | FRNM109 |
| - | FRNM122 |
| - | FRNM344 |
| - | FRNM435 |
| - | FRNM293 |
| - | FRNM139 |
| - | FRNM380 |
| - | FRNM126 |
| - | FRNM243 |
| - | FRNM108 |
| - | FRNM135 |
| - | FRNM270 |
| - | FRNM133 |
| - | FRNM216 |
| - | FRNM262 |
| - | FRNM116 |
| - | FRNM266 |
| - | FRNM438 |
| - | FRNM059 |
| - | FRNM130 |
| - | FRNM226 |
| - | FRNM320 |
| - | FRNM360 |

| | |
|---|---|
| - | FRNM335 |
| - | FRNM244 |
| - | FRNM394 |
| - | FRNM252 |
| - | FRNM412 |
| - | FRNM322 |
| - | FRNM402 |
| - | FRNM234 |
| - | FRNM430 |
| - | FRNM161 |
| - | FRNM114 |
| - | FRNM219 |
| - | FRNM323 |
| - | FRNM136 |
| - | FRNM321 |
| - | FRNM267 |
| - | FRNM214 |
| - | FRNM308 |
| - | FRNM167 |
| - | FRNM347 |
| - | FRNM366 |
| - | FRNM236 |
| - | FRNM376 |
| - | FRNM436 |
| - | FRNM237 |
| - | FRNM431 |
| - | FRNM100 |
| - | FRNM406 |
| - | FRNM030 |
| - | FRNM410 |
| - | FRNM155 |
| - | FRNM231 |
| - | FRNM157 |
| - | FRNM103 |
| - | FRNM367 |
| - | FRNM365 |
| - | FRNM240 |
| - | FRNM386 |
| - | FRNM112 |

| | |
|---|---|
| - | FRNM385 |
| - | FRNM381 |
| - | FRNM260 |
| - | FRNM426 |
| - | FRNM110 |
| - | FRNM423 |
| - | FRNM156 |
| - | FRNM102 |
| - | FRNM337 |
| - | FRNM117 |
| - | FRNM3829 |
| - | FRNM263 |
| - | FRNM125 |
| - | FRNM362 |
| - | FRNM313 |
| - | FRNM241 |
| - | FRNM371 |
| - | FRNM119 |
| - | FRNM245 |
| - | FRNM392 |
| - | FRNM251 |
| - | FRNM349 |
| - | FRNM013 |
| - | FRNM186 |
| - | FRNM427 |
| - | FRNM192 |
| - | FRNM384 |
| - | FRNM045 |
| - | FRNM151 |
| - | FRNM420 |
| - | FRNM268 |
| - | FRNM242 |
| - | FRNM388 |
| - | FRNM238 |
| - | FRNM314 |
| - | FRNM247 |
| - | FRNM398 |
| - | FRNM340 |
| - | FRNM411 |

| | |
|---|---|
| - | FRNM389 |
| BSW00331 | FRNM021 |
| BSW00337 | FRNM021 |
| BSW00338 | FRNM022, FRNM049 |
| BSW00339 | FRNM023 |
| BSW00369 | FRNM057, FRNM056 |
| BSW00370 | FRNM065 |
| BSW00380 | FRNM064 |
| BSW00381 | FRNM065 |
| BSW004 | FRNM074 |
| BSW00406 | FRNM071, FRNM073 |
| BSW00412 | FRNM065 |
| BSW00416 | FRNM029 |
| BSW00429 | FRNM220 |
| BSW00432 | FRNM176 |
| BSW02503 | FRNM043 |
| BSW02504 | FRNM044 |
| BSW02505 | FRNM222 |
| BSW02506 | FRNM047 |
| BSW02508 | FRNM037 |
| BSW02509 | FRNM185 |
| BSW02511 | FRNM187 |
| BSW045 | FRNM034 |
| BSW046 | FRNM168 |
| BSW048 | FRNM101 |
| BSW050 | FRNM104 |
| BSW051 | FRNM106 |
| BSW052 | FRNM181 |
| BSW053 | FRNM035 |
| BSW054 | FRNM101 |
| BSW101 | FRNM028 |
| BSW137 | FRNM035 |
| BSW144 | FRNM179 |
| BSW147 | FRNM225 |
| BSW150 | FRNM077, FRNM187, FRNM179, FRNM180 |
| BSW151 | FRNM175 |
| BSW154 | FRNM034 |

# 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 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 enter the Bus-Sleep Mode, it stops sending NM-messages, but as long as NM-messages from other nodes are received, it postpones transition into the Bus-Sleep Mode. Ultimately, if a designated timer elapses as a result of prolonged absence of NM-messages, then the node initiates transition into the Bus-Sleep Mode.

If any node in the NM-cluster requires bus-communication, then it can “wake-up”[1] the NM-cluster from the Bus-Sleep Mode by transmitting NM-Vote messages. For additional details concerning the wakeup procedure, please refer to the Mode Management (see [9], [16]).

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 Generic 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 7.8.6 on page 51.

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 to indicate 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 on the local node and there are no observable NM-messages on the bus for a configurable amount of time determined by **`FrNmReadySleepCnt`** (configuration parameter), then the FrNm module shall perform the transition into the Bus-Sleep Mode. ⌋ (BSW048, BSW054)

Example: FrNmReadySleepCnt = 3; Repetition Cycle = 4 vote cycles; Vote Cycle = 1 FlexRay Cycle; FlexRay Cycle = 5 msec: Time equivalent of **`FrNmReadySleepCnt`** = 3* 4* 1* 5 msec = 60 msec.

**[FRNM102]** ⌈The AUTOSAR FlexRay NM state machine shall contain states, transitions and triggers required by 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 to selected interface functions or by the expiration of internal timers or counters. ⌋ ( )

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

**[FRNM168]** ⌈The FrNm module shall synchronize state changes in the FlexRay NM state machine with the FlexRay periodic Schedule. ⌋ (BSW046)

Rationale: The FlexRay NM algorithm is based on the fact that all ECUs, which participate in the FlexRay NM, are synchronized to a global time (based on periodic repetition of its communication scheme, the so called Cycle – see [5]). To prevent asymmetric behavior of the ECUs (e.g., only a subset of the ECUs changes to sleep mode, while the remaining subset 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 on all ECUs in the NM cluster.

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

**[FRNM375]** ⌈The FrNm module shall access the FlexRay bus communication cycle via the API FrIf_getGlobalTime. ⌋ ( )

## 7.2 Operational modes

In this chapter, the operational modes of the AUTOSAR FlexRay NM coordination algorithm are described in detail.
Figure (in chapter 7.4 on page 38) shows the detailed UML state chart of the FlexRay NM.

**[FRNM105]** ⌈The AUTOSAR FrNm shall consist of three operational modes:
Bus-Sleep Mode
Synchronize Mode
Network Mode⌋( )

**[FRNM106]** ⌈Changes in the AUTOSAR FrNm operational modes shall be notified by the FrNm module to the Generic NM Interface module by calling callback function **`Nm_StateChangeNotification`**.⌋(BSW051)

**[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]** ⌈FlexRay NM module shall be configurable with the configuration parameter FrNmMainAcrossFrCycle if the execution of FrNmMain function crosses the FlexRay cycle boundary (set to TRUE) or not (set to FALSE).⌋( )

**[FRNM372]** ⌈If the FlexRay NM Vector is only available at the end of a FlexRay cycle then evaluation takes place in the following FlexRay cycle.⌋( )

Note: This depends on the configuration and implementation of the FlexRay communication controller hardware.

Evaluation of the condition RepetitionCycleCompleted shall be evaluated according to the following rules:

a) If FrNmMainAcrossFrCycle is set to FALSE, then a RepetitionCycleCompleted event shall be generated in the last FlexRay bus cycle of the repetition cycle after we receive all the votes.

Example: If the votes are scheduled in cycles 1 and 2, and the repetition cycle period is 4 FlexRay bus cycles, then the cycle completed event can be generated (once only) any time during cycle 3, 7, 11, 15, etc..

b) If FrNmMainAcrossFrCycle is set to TRUE, then a RepetitionCycleCompleted event shall be generated if a REPETITION cycle boundary has been crossed since the previous call of the FlexRay NM main function before we transmit any vote.

Example: If the votes are scheduled in cycles 1 and 2, and the repetition cycle period is 4 FlexRay bus cycles, then the cycle completed event can be generated (once only) any time during cycle 0, 4, 8, 12, 16, etc..

Please see also Figure 7-2 FrNm Mainfunction Execution for the scheduling constraints of the FlexRay NM main function.

Note: CycleNumber stands for the FlexRay communication cycle number whose value is anywhere between 0 and 63 as integers. RepetitionCycleLength is the number of communication cycles within one NM Repetition Cycle. FlexRay "CycleEnd" Event is the event generated at the boundary of two consecutive FlexRay communication cycles.

### 7.2.1 Bus-Sleep Mode

The Bus Sleep Mode is the default mode after the initialization 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 when the power of the CPU is switched off.
In the Bus Sleep Mode, the communication controller can be switched into the sleep mode where wakeup detection mechanisms are activated and power consumption is reduced to a minimal level. The corresponding functionality (shut down of FlexRay, power down of the CPU) will be implemented in other modules. The FlexRay NM will only indicate the readiness of Sleep Mode.

**[FRNM134]** ⌈When Bus-Sleep Mode is entered, the FlexRay NM shall notify the Generic Network Management Interface by calling **`Nm_StateChangeNotification`**, except when the Bus-Sleep Mode is entered by default during initialization.⌋( )

Note: The FlexRay NM will notify the ComM via the Generic NM Interface (macro adaptation layer). This notification is an optional interface.

**[FRNM135]** ⌈When Bus-Sleep Mode is entered, except by default at initialization, the FrNm module shall notify the upper layer by calling the call-back function **`Nm_BusSleepMode`**.⌋( )

**[FRNM320]** ⌈When the FrNm module enters the Bus-Sleep Mode, the module shall set the flag **`FrNm_RepeatMessage`** to FALSE.⌋( )

**[FRNM137]** ⌈When the FrNm module enters the Bus-Sleep Mode, the module shall deactivate the transmission of both NM-Data and NM-Vote. ⌋( )

**[FRNM175]** ⌈When the FrNm module successfully receives a NM-message, and is currently in the Bus-Sleep Mode, it shall notify the ComM via the NmIf module by calling **`Nm_NetworkStartIndication()`** FRNM257⌋(BSW151)

Rationale: FRNM175 is required to avoid race conditions and state inconsistency between Network Management 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.

**[FRNM143]** ⌈The FrNm module is allowed to transition to the Network mode after the reception of a NM repetition cycle event in the case of FRNM had entered the Synchronize state in the first place because of an active or a local wakeup request.. ⌋( )

**[FRNM308]** ⌈When the FrNm module enters the Synchronize mode, it shall deactivate the transmission of NM-Data and deactivate the transmission of the NM-Vote. ⌋( )

**[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 FrNm module shall consist of three internal states:
Repeat Message State
Normal Operation State
Ready Sleep State ⌋( )

**[FRNM115]** ⌈The FrNm module shall synchronize all state changes into and within the Network Mode with 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 59.

**[FRNM108]** ⌈On entering the Network Mode state, the FlexRay Network Management shall first enter the internal sub-state Repeat message state. ⌋( )

**[FRNM109]** ⌈When the FrNm module has entered the Network Mode, it shall set the flag **`FrNm_RepeatMessage`** to TRUE. ⌋( )

**[FRNM110]** ⌈When the FrNm module has entered the Network Mode, it shall notify the Generic NM Interface module by calling **`Nm_NetworkMode`**. ⌋( )

**[FRNM133]** ⌈The FlexRay NM state machine shall leave the Network Mode and enter the Bus-Sleep Mode when the Repetition Cycle completes and when FrNmReadySleepCnt < 1 condition evaluates to TRUE. ⌋( )

**[FRNM111]** ⌈When the FrNm 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`**) expires in the Network Mode, then the FrNm module shall set the flag **`FrNm_RepeatMessage`** to FALSE. ⌋( )

**[FRNM119]** ⌈When the FrNm module enters the Network Mode, it shall initialize the NM Repeat Message Timer (**FrNm_RepeatMessageTimer**) to a configurable amount of time determined by **FrNmRepeatMessageTime** (configuration parameter).⌋( )

**[FRNM307]** ⌈The FrNm module shall immediately leave the Network mode when the function **FrNm_Init** is called.⌋( )

**[FRNM309]** ⌈The **FrNmReadySleepCnt** shall be set to a type that is capable of supporting 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 the 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 FrNm module enters 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 allow voting.⌋( )

**[FRNM117]** ⌈When the FrNm module enters the Repeat Message State, it shall start the NM Repeat Message Timer (**FrNm_RepeatMessageTimer**).⌋( )

**[FRNM120]** ⌈The FrNm module shall leave the Repeat Message State when a Repetition Cycle is completed and if the flag **FrNm_RepeatMessage** is set to FALSE, and then 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 state machine leaves 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 state machine leaves 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). For such a node, it is up to the implementation to optimize this state and remove the code corresponding to this state.

**[FRNM123]** ⌈When the FrNm module enters the Normal Operation State, it shall activate the transmission of NM-Data and the Node shall vote to keep the cluster awake (send “positive” NM-Votes). ⌋( )

**[FRNM124]** ⌈The FrNm 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 FrNm 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 FrNm module enters the Ready Sleep State, it shall deactivate the transmission of NM-Data and deactivate the transmission of the NM-Vote for keeping the cluster awake.⌋( )

**[FRNM127]** ⌈When the FrNm module enters the Ready Sleep State from a state other than the Ready Sleep State, it shall initialize the Ready Sleep Counter (**`FrNmReadySleepCnt`**) with the start value **`FrNmReadySleepCnt`** (configurable parameter) – see FRNM101.⌋( )

**[FRNM128]** ⌈When the FrNm module is in Ready Sleep state and detects a NM-Vote to keep the cluster awake it shall initialize the Ready Sleep Counter (**`FrNmReadySleepCnt`**) with the start value **`FrNmReadySleepCnt`** (configurable parameter).⌋( )

**[FRNM129]** ⌈The FrNm module shall leave the Ready Sleep State (and the Network Mode) and enter the Bus-Sleep Mode if the Ready Sleep Counter has expired (**`FrNmReadySleepCnt`** < 1) at the end of the NM Repetition Cycle.⌋( )

Note: As all transitions regarding the **`FrNmReadySleepCnt`** are guarded and the order of evaluation is implicitly defined by the guards—the **`FrNmReadySleepCnt`** is only decremented if no Repeat Message Request is active (**`FrNm_RepeatMessage`** is set to FALSE), the network has been released (**`FrNm_NetworkRequested`** is set to FALSE) and if no Cluster awake vote has been received on the bus.

**[FRNM130]** ⌈The FrNm 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 (**`FrNmReadySleepCnt`** >0) and a Repeat Message Request is active (the flag **`FrNm_RepeatMessage`** is set to TRUE – see FRNM111).⌋( )

**[FRNM131]** ⌈The FrNm 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 (**`FrNmReadySleepCnt`** >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 FrNm 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 (**FrNmReadySleepCnt** >0).⌋( )

**[FRNM314]** ⌈When both event "ClusterAwakeVote detected" and "Repetition Cycle completed" in state "Ready Sleep" occurs at the same time the transition dependent on "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.

**[FRNM337]** ⌈It shall be configurable with 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 FrNm module.⌋( )

**[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 exists in parallel to the state machine described in chapter 7.4. Network states distinguish between 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  shows an UML state diagram with respect to the API specification.

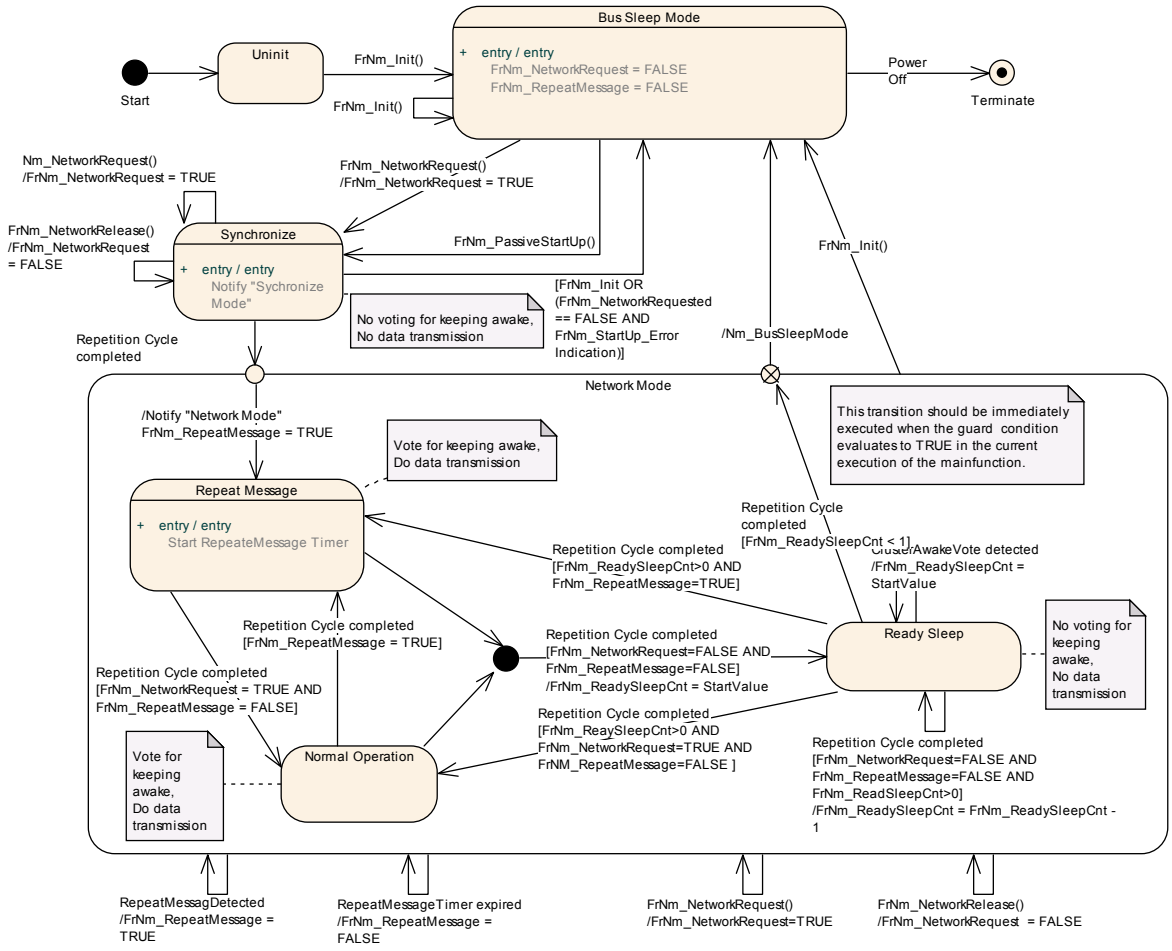



Figure 7-1 UML State Chart

## 7.5 Initialization and Startup

**[FRNM071]** ⌈The FrNm module shall store the initialization status in a private variable.⌋ (BSW00406)

**[FRNM029]** ⌈The FrNm module’s environment shall initialize the FrNm module after the corresponding FlexRay Interface is initialized and before any other FlexRay NM service is called.⌋ (BSW00416)

**[FRNM032]** ⌈If the FrNm module is not initialized, then the FrNm module shall reject a call of any FlexRay NM function, with the exception of **`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 to coordinate shutdown. These messages are sent according to the schedule configured within each ECU and coordinated across the NM Cluster.

### 7.6.1 General requirements

The FrIf shall be configured for every node 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 [5]).
The FlexRay Controller calculates this NM-Vector 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 transmission slots.
The FlexRay communication controller will maintain an aggregated network management vector throughout each communication cycle by applying a bit-wise OR between each received Network Management Vectors (regardless of whether the frame is subscribed to a receive buffer).
This method is a powerful way to receive the NM-Vector from nodes on the network without involving the CPU, but requires at least one send-slot for every node (participating 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 FrNm module shall be able to separately transmit NM Data and NM Vote.⌋( )

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 [5]). 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 with the configuration parameter **`FRNM_PDU_SCHEDULE_VARIANT`** which NM-message transmission format (NM-Data, NM-Vote and the combined NM-Data/Vote format) are recognized by the FrNm module. ⌋ ( )

Rationale: The FrNm module must be capable of receiving and processing 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 of sending the NM-Vote (in the either the static or the dynamic segment) of the FlexRay bus schedule if the node is not configured as “Passive”. ⌋ ( )

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]** ⌈In every FlexRay NM node it shall be independently configurable through the configuration parameter **`FrNmHwVoteEnable`** to use the FlexRay NM HW support for reception of NM-Votes that are transmitted in the static segment. ⌋ ( )

**[FRNM151]** ⌈Every FlexRay NM node shall be capable of sending the NM-Data in either a static slot or in a dynamic slot. ⌋ ( )

Note: The FrIf and FrDv configuration is responsible for the actual FlexRay schedule configuration. This requirement requires that the FrNm 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.

**[FRNM335]** ⌈When NM-Vector hardware service from the FlexRay CC is used, then **`FrIf_GetNmVector`** will be used to retrieve the aggregated NM-Vector. ⌋ ( )

**[FRNM336]** ⌈The FlexRay Network Management must react (execute) synchronously on reception of the indication **`FrNm_StartupError`** from the FlexRay State Manager even when the FrNM Mainfunction is no longer executing. ⌋ ( )

Rationale: The FlexRay NM must be able to go to Bus Sleep state after synchronization is lost.

### 7.6.2 FlexRay NM-PDU format

As specified in FRNM147 the FlexRay NM is capable of sending 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 by the FrNm module shall be configurable at pre-compile time by the configuration parameter **FrNmControlBitVectorEnabled** (see chapter 10). ⌋( )

**[FRNM222]** ⌈The support of the Source Node Identifier by the FrNm module shall be configurable at pre-compile time by the configuration parameter **FRNM_SOURCE_NODE_IDENTIFIER_ENABLED** (see chapter 10). ⌋(BSW02505)

**[FRNM381]** ⌈If the parameter FRNM_**SOURCE_NODE_IDENTIFIER_ENABLED** (see chapter 10) is set to FALSE, this byte shall be used for user data. ⌋( )

**[FRNM154]** ⌈The length of the NM-Data PDU for the FrNm module shall be configurable at pre-compile time to any integer value from 1 to 8 by the configuration parameter **FrNmPduLength** (see chapter 10). The **FrNmPduLength** is defined by the PduLength parameter [EcuC003_Conf] in the "global" ECUC module (see Ecu Configuration specification). ⌋( )

**[FRNM313]** ⌈The ability of the FrNm module to send NM-Data PDU shall be configurable at pre-compile time by the configuration parameter **FRNM_NM_USER_DATA_ENABLED** (see chapter 10). ⌋( )

**[FRNM155]** ⌈The difference between applied standardized bytes and the NM-Data PDU length in the FrNm module shall be user data.⌋( )

**[FRNM324]** ⌈If `FrNmControlBitVectorEnabled` is set to FALSE, Control Bit Vector of the NM-Data PDU shall not be used for other Data. It remains reserved.⌋( )

**[FRNM3829]** ⌈If `FrNmControlBitVectorEnabled` is set to FALSE, then Control Bit Vector of the NM-Data PDU shall be set to 0x00.⌋( )

**[FRNM156]** ⌈The NM-Data PDU Control Bit Vector of the FrNm 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 | Cluster Request Information Bit | Res | Active Wakeup Bit | NM Coordinator Sleep Ready | Res | Res | RptMsg Request |

Table 7-2 Control Bit Vector Format⌋( )

**[FRNM055]** ⌈The Control Bit Vector of the FrNm module shall contain a Repeat Message Request Bit (RptMsgRequest) with the following meaning:
0: Repeat Message State not requested
1: Repeat Message State requested
The Control Bit Vector of the FlexRay NM module shall contain an Active Wakeup Bit with the following meaning.
0: Node has not woken up the network
1: Node has woken up the network
The Control Bit Vector of the FlexRay NM module shall contain a Partial Cluster Request Information Bit (CRI)
0: NM message contains no Partial Network request information
1: NM message contains Partial Network request information
NM Coordinator Sleep Ready Bit
0: NM cluster is not ready to sleep
1: NM cluster is ready to sleep (All nodes of the NM cluster are ready to sleep)⌋( )

Hint: In AutoSAR Release 3.2 Bit 1 and 2 of the CBV are used for the NM-Coordinatior ID.

**[FRNM157]** ⌈The FrNm 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 FrNm module shall set Bit 7 of the Control Bit Vector to $0_b$.⌋( )

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

**[FRNM161]** ⌈The FrNm module shall set Bit 1 to 6 of the Control Bit Vector to $0_b$ which are reserved for future extensions.⌋( )

### 7.6.2.2 FlexRay NM-Vote PDU format

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

**[FRNM218]** ⌈The FrNm 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 together with an NM-Vote.

**[FRNM219]** ⌈The FrNm 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.

**Note:** The FlexRay Interface and the FlexRay Driver modules shall place the PDU containing the NM-Vote at the start of the FlexRay frame if the NM-Vote is to be 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 [5]). 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 59) 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 | Cluster Request Information Bit | Res | Active Wakeup Bit | NM Coordinator Sleep Ready | Res | Res | RptMsg Request |
| + | | | | | | | | |
| NM-Vote PDU | Vote | Not available | | | | | | |
| Combined CBV and Vote | Vote | Cluster Request Information Bit | Res | Active Wakeup Bit | NM Coordinator Sleep Ready | Res | Res | RptMsg Request |

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

**[FRNM162]** ⌈The FrNm module shall combine the NM-Vote PDU Format with the Control Bit Vector Format of the NM-Data PDU in case the FrNm 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 [8].

### 7.6.4 FlexRay NM-PDU reception

The FlexRay Reception Indication is used to indicate reception of FlexRay NM-PDU receptions. For more details see FlexRay Interface SWS [8].

### 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 using the configuration parameter **`FrNmNodeId`** (see chapter 10).⌋ (BSW02508)

## 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_< FrNmChannelIdRef >) 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 (and availability of the repeat message bit if node detection is enabled) 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.

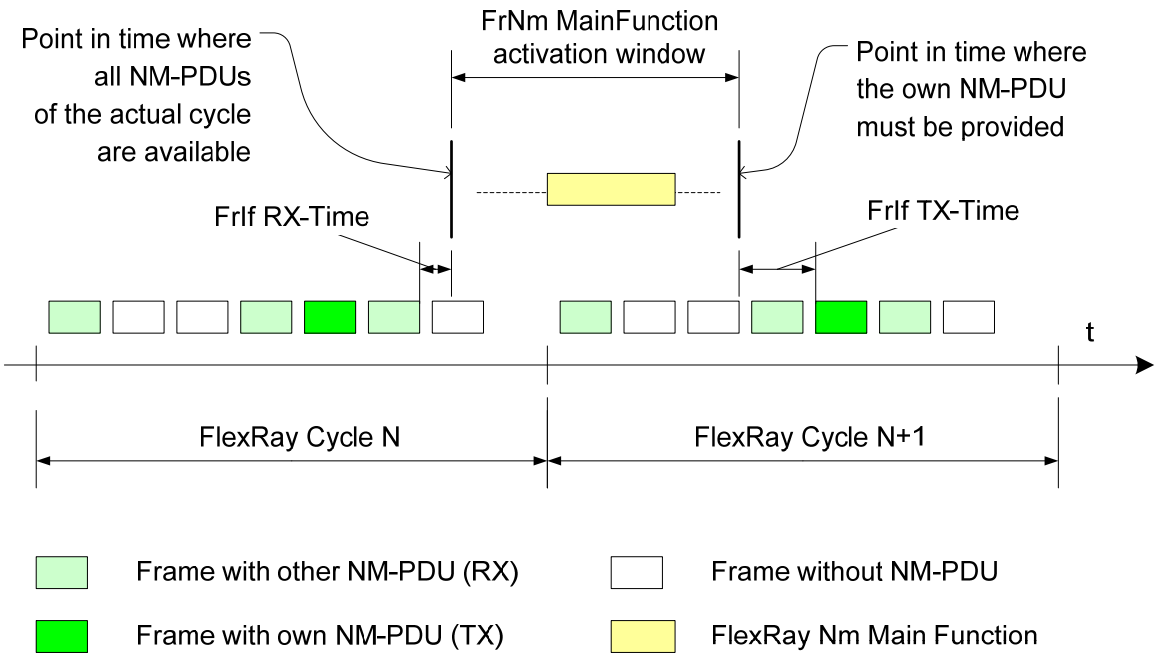



Figure 7-2 FrNm Mainfunction Execution

### 7.7.1 General requirements

**[FRNM225]** ⌈The FrNm 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. ⌋(BSW147)

**[FRNM176]** ⌈The FlexRay NM shall realize FlexRay NM functions for reception and for transmission of NM Messages.⌋(BSW00432)

Note: It is upon the implementer to implement the above functionality as one NM-Task, which is activated by the **FrNm_<Rx|Tx>Confirmation** callbacks, or as several NM-Tasks, each with separate functionalities, but [4][BSW00432] requires the split of the functions.

Note: The FlexRay NM shall realize a Transmit function only if the node is configured as an active node, i.e., only if **FRNM_PASSIVE_MODE_ENABLED** (configuration parameter) is set as OFF.

### 7.7.2 FlexRay NM-Task structure

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

**[FRNM010]** ⌈The FrNm 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.⌋( )

**[FRNM360]** ⌈The FlexRay Interface module shall call the FrNm module function **FrNm_TxConfirmation** when a NM-message is successfully transmitted and a transmit confirmation is supported and configured within the FlexRay Interface.⌋( )

**[FRNM363]** ⌈If transmission confirmation is supported by the FlexRay Interface, and the FlexRay Interface provides a **FrNm_TxConfirmation**, then the FlexRay NM shall provide a callback called **PduR_FrNmTxConfirmation.**⌋( )

**[FRNM361]** ⌈The FlexRay Interface module shall call the FrNm module function **FrNm_RxIndication** when a NM-message is received.⌋( )

Note: It is up to the implementation as to how the FrNm module is to handle the data from the NM-message. It can be immediately processed (to reduce the memory consumption), or it can be stored to be available to be processed (to reduce computing time).

### 7.7.3 FlexRay NM-Task execution

#### 7.7.3.1 Synchronous FlexRay NM-Task execution

**[FRNM048]** ⌈The FlexRay NM module integrator shall define a schedule to activate the FlexRay NM task synchronously to the FlexRay communication cycle. ⌋( )

Implementation hint:The execution has to occur at least once within a window starting at the time where all NM-Votes of a vote cycle are available, and ending at the time where the node has to provide its next NM-Vote PDU or the first NM-Vote of the next vote cycle is received, before "repetition cycle completed" occurs.
To guarantee synchronous transition into Bus Sleep Mode, the last Nm main task within a vote cycle has to consider the same votes as the last Nm main task of all other nodes."

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 [15] 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. The AUTOSAR FlexRay NM shall try to use the second absolute timer whenever possible as race conditions might occur between the FlexRay NM access to the first absolute timer and the FlexRay Interface Job List execution function access to the first absolute timer.

**[FRNM007]** ⌈The FrNm module's environment should execute the FrNm module (**`FrNm_MainFunction_<FrNmChannelIdRef>`**) at least once a FlexRay communication cycle, synchronous to the FlexRay global time when FlexRay global time is available, and the FrNm module's execution should not cross a FlexRay cycle boundary. The **`FrNm_MainFunction_<FrNmChannelIdRef>`** should be executed periodically even when FlexRay global time is no longer available. ⌋( )

**[FRNM356]** ⌈The state machine guards, transitions, conditions, and actions should be evaluated at most once in each execution of the **`FrNm_MainFunction_<`**FrNmChannelIdRef **`>`**.⌋( )

#### 7.7.3.2 Asynchronous FlexRay NM-Task execution

**[FRNM177]** ⌈The FrNm 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 yet been found.

## 7.8 Additional Features

### 7.8.1 Cluster size

**[FRNM179]** ⌈The AUTOSAR FlexRay NM algorithm shall support up to 64 nodes per NM-Cluster.⌋(BSW150, BSW144)

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). This upper limit 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 resulting 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 FrNm module shall be configurable at pre-compile time by the configuration parameter **`FrNmRemoteSleepIndicationEnabled`** (see chapter 10).⌋(BSW150)

Note: If a node is configured as Passive (FRNM187), then the remote sleep indication shall not be used because the node cannot 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, then the NM shall notify the Generic NM Interface module that all other nodes in the cluster are ready to sleep (the ‘Remote Sleep Indication’) by calling **`Nm_RemoteSleepIndication.`**⌋(BSW052)

**[FRNM186]** ⌈The FrNm 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** when not in Network Mode and shall not execute any functionality.⌋( )

**[FRNM229]** ⌈If a Remote Sleep Indication has been previously detected by the FrNm module and if an NM-message with an indication to keep the bus awake is received in the Normal Operation State or Ready Sleep State, then the NM shall notify the Generic NM Interface module that some nodes in the cluster are not ready to sleep anymore (Remote Sleep Cancellation) by calling **Nm_RemoteSleepCancellation**. The Remote Sleep cancellation needs to be provided as soon as a vote is received to keep the network awake.⌋( )

Note: Specifically, this should not be delayed until the end of the repetition cycle boundary.

**[FRNM230]** ⌈If Remote Sleep Indication has been previously detected and the FrNm enters the Repeat Message State from the Normal Operation State or from the Ready Sleep State, then the NM shall notify the Generic NM Interface module that some nodes in the cluster are not ready to sleep anymore (the 'Remote Sleep Cancellation') by calling **Nm_RemoteSleepCancellation.** ⌋( )

**[FRNM322]** ⌈In order to support the NM Coordination algorithm in the NmIf module, an indication **Nm_SyncrhonizationPoint** provided to the NmIf module only at the repetition cycle boundaries, when the FrNm module is in Network Mode, and when the configuration parameter **FRNM_SYNCHRONIZATIONPOINT_ENABLED** is set to TRUE.⌋( )

**[FRNM323]** ⌈**FRNM_SYNCHRONIZATIONPOINT_ENABLED** is allowed to be set to true only when **FRNM_REMOTE_SLEEP_INDICATION_ENABLED** is set to TRUE.⌋( )

### 7.8.3 Detection of Nodes (optional)

Nodes that 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 in the Normal Operation State. These data can be received by other nodes using the **FrNm_GetNodeIdentifier** function.

A node can identify (detect) nodes on the FlexRay Bus by repeatedly calling the previously mentioned function.

To ensure that all nodes (that 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 FrNm module shall be configurable at pre-compile time by the configuration parameter **FrNmNodeDetectionEnabled** (see chapter 10). **FrNmSourceNodeIdentifierEnabled** should also be set as ON to use this feature.. ⌋( )

### 7.8.4 User data (optional)

**[FRNM077]** ⌈The support for user data in the FrNm module shall be configurable at pre-compile time by the configuration parameter **FrNmUserDataEnabled** (see chapter 10). ⌋(BSW150)

**[FRNM362]** ⌈If **FrNmUserDataEnabled** is set as TRUE then **PduR_FrNmTriggerTransmit** will query the NM User data (NM I-PDU) from PduR and populate the User Data part of the NM User data message as to be transmitted by the FrNm. ⌋( )

**[FRNM364]** ⌈The FrNm shall collect the NM User Data from the referenced NM I-PDU by calling “PduR_FrNmTriggerTransmit” and combine the user data with the further NM bytes each time before it requests the transmission of the corresponding NM message. ⌋( )

**[FRNM394]** ⌈FrNm shall call **PduR_FrNmTriggerTransmit** within the function **FrNm_TriggerTransmit** called by the FrIf. ⌋( )

**[FRNM365]** ⌈The FrNm shall call “PduR_FrNmTxConfirmation” within the message transmission confirmation function “FrNm_TxConfirmation” called by the FrIf. ⌋( )

**[FRNM366]** ⌈The FrNm implementation shall provide an API FrNm_Transmit (see FRNM359). This API shall never be called by PduR as the FrNm will always query the data by means of PduR_FrNmTriggerTransmit. FrNm_Transmit is an empty function returning E_NOT_OK at any time. This requirement is relevant to avoid linker errors as PduR expects this API to be provided. ⌋( )

Hint: NM user data is handled by a SW-C like a “normal signal”. User data consistency is guaranteed even if more than one SWC send user data. Relocation of a SW-C / VFB concept supported.

Hint: SW-C sends a signal containing user data, RTE propagates signal to COM, COM have to be configured that all user data signals of a channel are aggregated in one IPDU. Send type of that IPDU have to be configured “NONE” (-> COM never sends this signal)

Note: Missing abstraction of user data content and structure (length and structure of NM user data is OEM specific

### 7.8.5 NM Data (optional)

**[FRNM321]** ⌈The support for NM data in the FrNm module shall be configurable at pre-compile time by the configuration parameter **`FrNmUserDataEnabled`**. If data needs to be enabled then **`FrNmUserDataEnabled`** will be set to TRUE, otherwise it will be set to FALSE.⌋( )

### 7.8.6 Passive Node Configuration (optional)

Nodes that are configured as a “Passive” Mode shall participate in the cluster NM only in a passive way—unable to transmit any NM vote or NM data. Such nodes can only receive NM-Votes, but can not transmit votes that keep the cluster awake. Such a passive node can never change to the Normal Operation State (in the State Machine). It is a configuration error if the ComM calls 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_MODE_ENABLED`** (see chapter 10).⌋(BSW150, BSW02511)

**[FRNM188]** ⌈If Passive Node Configuration is enabled (FRNM187), then the FrNm module shall not use the Remote Sleep Indication options (FRNM180).⌋( )

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

### 7.8.7 NM PDU Rx Indication (optional)

Upper layers could use the optional PDU Rx Indication to detect NM activity (reception of a NM-Vote, NM-Data, or combined NM-Data and Vote PDU) on the FlexRay bus. However, since many NM PDUs could 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), then the FrNm module shall call the function **`NM_PduRxIndication`** at the successful reception of an NM-PDU.⌋( )

### 7.8.8 State change notification (optional)

**[FRNM191]** ⌈The inclusion of 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), then the FrNm module shall notify all its state changes to the NmIf module by calling `Nm_StateChangeNotification.`⌋( )

### 7.8.9 Dual FlexRay Channel PDU support (optional)

As described in more detail in 7.9, the FlexRay NM shall support the transmission and reception 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 FlexRay channel A and one for FlexRay channel B). The following requirements describe the required functionality.

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

**[FRNM357]** ⌈Vote changes for all nodes except one in a FlexRay cluster in certain dual-FlexRay channel topologies[2] shall be forbidden in the next-to-last repetition cycle before the Ready Sleep Counter expires, i.e., the NM votes transmitted by the Single-FlexRay channel nodes will be identical in both the last repetition cycle and the next-to-last repetition cycle if the Ready Sleep Counter were to expire. This shall be statically configurable at pre-compile time by the configuration parameter **`FrNmVotingNextToLastRepetitionCycleDisable.`**⌋( )

Rationale: One of the challenges with extending the current strategy to dual-FlexRay channel configurations is that votes transmitted by single-FlexRay channel nodes are not visible to single-FlexRay channel nodes on the opposite FlexRay channel.

[2] The challenging topology to manage effectively is the topology that has nodes attached only to Channel A, nodes attached only to Channel B, and nodes that are dual-channel.

Avoidable race conditions might arise due to an inherent delay in forwarding the vote by the dual-FlexRay channel node for topologies where certain nodes are connected to only a single FlexRay channel and certain other nodes are connected to the other FlexRay channel. However, it can be readily mitigated with a slight modification where vote changes by nodes voting only on a single FlexRay channel are forbidden to change their vote in the next-to-last repetition cycle before the possible expiration of the Ready Sleep Counter.

### 7.8.10 Car Wakeup (optional)

**[FRNM402]** ⌈The position of the CWU bit in NM messages shall be defined by FrNmCarWakeUpBytePosition and FrNmCarWakeUpBitPosition.⌋( )

Rx Path

**[FRNM410]** ⌈If the car wakeup bit within any received NM-PDU is `1` and `FrNmCwuRxEnabled` is `TRUE`, FrNm shall call `Nm_CarWakeUpIndication` and perform the standard Rx indication handling.⌋( )

**[FRNM411]** ⌈If `FrNm_GetPduData` is called in the context of `Nm_CarWakeUpIndication`, FrNm shall return the PDU data of the PDU that causes the call of `Nm_CarWakeUpIndication`.⌋( )

**Note**: This is required to enable the ECU to identify detail about the sender of the car wakeup request.

**[FRNM412]** ⌈If `FrNmCarWakeUpFilterEnabled` is `TRUE`, the car wakeup bit within any received NM-PDU is `1`, `FrNmCwuRxEnabled` is `TRUE` and the Node ID in the received NM-PDU is equal to `FrNmCarWakeUpFilterNodeId` the FrNm module shall call `Nm_CarWakeUpIndication` and perform the standard Rx Indication handling.⌋( )

**Note**: The car wakeup filter is necessary to realize sub gateways that only consider the car wakeup of the central Gateway to avoid wrong wakeups.

Tx Path

The transmission of the car wakeup bit shall be handled by the application using the NM user data mechanism provided by the FrNm module.

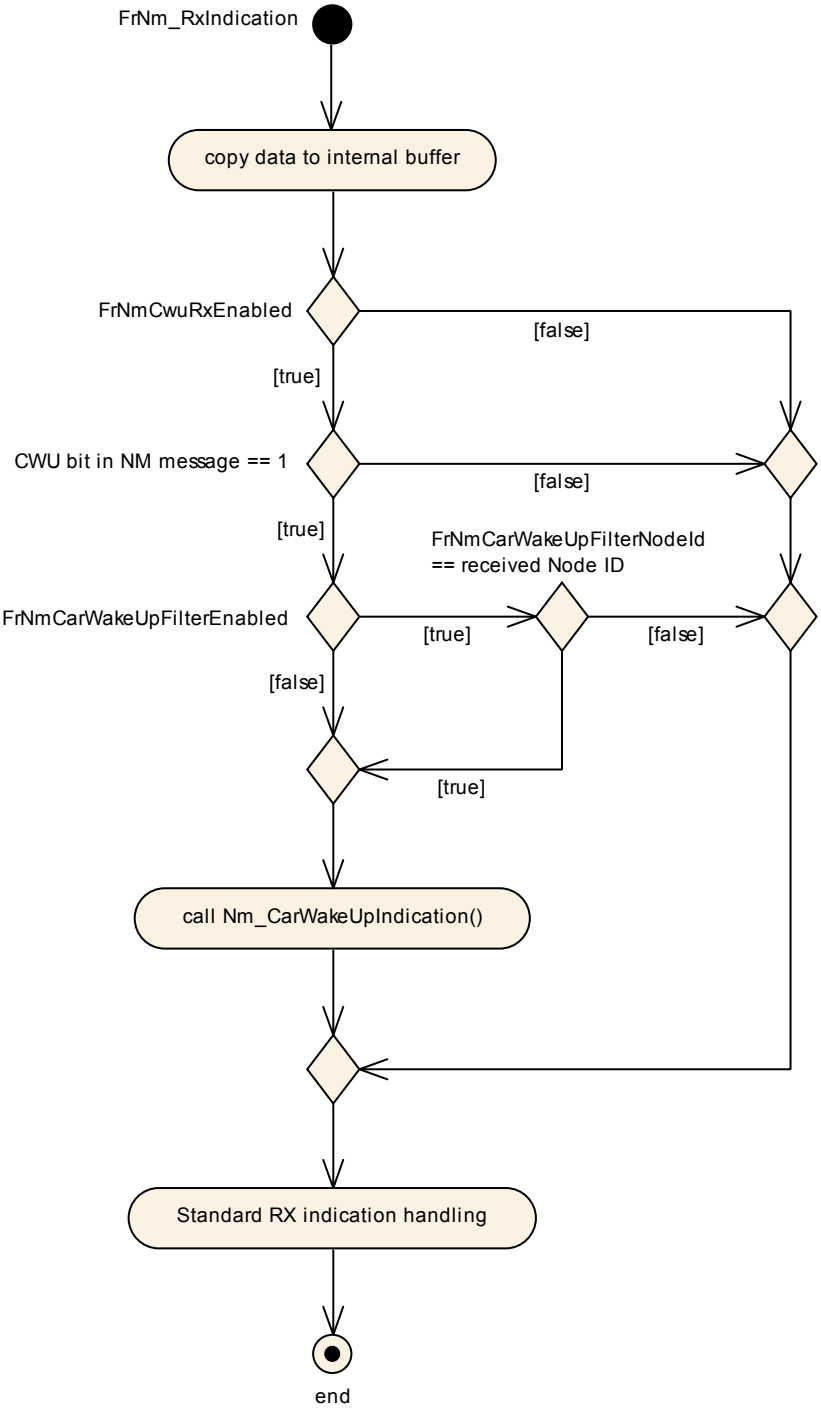



Figure 7-3 CarWakeUp indication handling

### 7.8.11 Coordinated Bus Shutdown with Backup Coordinator (optional)

The network management stack allows to have a coordinated shutdown of more than one bus if an ECU exists which is connected to the busses which are to be coordinated. The main functionality is included in the NmIf module.

**[FRNM396]** ⌈If the FrNm receives a NM message with *NMcoordinatorSleepReady* bit set (see CBV) it shall indicate this to the NmIf by calling `Nm_CoordReadyToSleepIndication`.⌋( )

**[FRNM398]** ⌈The *NMcoordinatorSleepReady* bit in the Control Bit Vector shall be set via the API `FrNm_SetSleepReadyBit.`⌋( )

**[FRNM397]** ⌈This feature is optional and only available if FrNmCoordinatorSyncSupport is set to TRUE⌋( )

### 7.8.12 Extension for Partial Network (optional)

To reduce the power consumption of communication domains, it shall be possible to switch off the communication stack of ECUs during active bus communication. To control the shutdown and wakeup of these ECUs (cluster) in a standardized way, the AUTOSAR Network Management shall be used. Thus the AUTOSAR FrNm must be extended by the following Features.

**Note**: It is not possible to switch of only one ECU in a FlexRay cluster.

**[FRNM405]** ⌈The FrNm extensions for partial networking are enabled with the configuration switch `FrNmPnEnabled`.⌋( )

**[FRNM404]** ⌈FrNm shall be able to distinguish between an NM PDU without PN request information and an NM PDU with PN request information contained in the NM user data. This is indicated by the Cluster Request Inforamtion bit (CRI) in the CBV of the NM PDU.
Meaning of the CRI bit:
0 -> NM PDU contains no PN request information
1 -> NM PDU contains PN request information⌋( )

**[FRNM411]** ⌈The range of the PN request information in the NM user data is defined by the parameters `FrNmPnInfoOffset` and `FrNmPnInfoLength`. Each bit in the PN range shall have the following meaning:
Bit value Meaning:
0 -> The PN is not requested
1 -> The PN is requested

**Note**: FrNmPnInfoOffset + FrNmPnInfoLength shall not exceed the Length of the NM PDU. ⌋( )

**[FRNM418]** ⌈By means of the configuration parameter FrNmPnFilterMaskByte the FrNm is able to detect which PN request information is relevant for the ECU and which not. Each bit of FrNmPnFilterMaskByte has the following meaning:
0 -> The PN request information is irrelevant for the ECU.
1 -> The PN request information is relevant for the ECU. ⌋( )

**[FRNM416]** ⌈The PN filter mask FrNmPnFilterMaskByte shall be a byte array with the length of FrNmPnInfoLength. ⌋( )

**[FRNM420]** ⌈If at least one bit within the PN request information Range of the received NM-PDU (defined by the parameters FrNmPnInfoOffset and FrNmPnInfoLength) matches with the corresponding bit in the PN filter mask, the PN request information is relevant for the ECU. ⌋( )

#### 7.8.12.1 RX-Path

**[FRNM406]** ⌈If FrNmPnEnabled is FALSE, FrNm shall perform the standard FrNm RxIndication. ⌋( )

**[FRNM407]** ⌈If FrNmPnEnabled is TRUE and the CRI bit in the NM PDU provided by FrNm_RxIndication() has the value 0, FrNm shall perform the standard FrNm RxIndication. ⌋( )

**[FRNM408]** ⌈If FrNmPnEnabled is TRUE and the CRI bit in the NM PDU provided by FrNm_RxIndication() has the value 1, FrNm module shall process the Partial Networking Information of the NM-PDU as described in chapter 7.8.12.3 Aggregation of internal and external requested PNs and 7.8.12.4 Aggregation of external requested PNs⌋( )

#### 7.8.12.2 TX-Path

**[FRNM409]** ⌈If FrNmPnEnabled is TRUE, FrNm shall set the value of the CRI bit to 1 in transmitted NM PDUs.

**Note**: If partial networking is used on the ECU, this ECU always has to send cluster request information in its NM user data. ⌋( )

**[FRNM410]** ⌈If FrNmPnEnabled is FALSE, FrNm shall set the value of the CRI bit to 0 in transmitted NM PDUs. ⌋( )

#### 7.8.12.3 Aggregation of internal and external requested PNs

This feature is used by every ECU that has to switch on/off Pdu-Groups because of the activity of partial networks. (e.g. to prevent false timeouts). The Pdu-Groups shall be switched on if the corresponding PN is requested internally or externally. The Pdu-Groups shall be switched off not until all internal and external requests for the corresponding PN are released. The FrNm only provides the information if the PN is internally/externally requested or not. The logic for switching the Pdu-Groups shall be implemented by an upper layer.

**[FRNM412]** ⌈If `FrNmPnEiraCalcEnabled` is `TRUE`, FrNm shall calculate the aggregation of external and internal PN requests information. ⌋ ( )

**[FRNM413]** ⌈If `FrNmPnEiraCalcEnabled` is `FALSE`, FrNm shall skip the aggregation of external and internal PN requests information. ⌋ ( )

The aggregated state of the internal/external requested PNs is called External Internal Requests Aggregated (EIRA).

**[FRNM421]** ⌈The EIRA shall have the size of `FrNmPnInfoLength` and shall be initialized with the value `0` (not requested) for every PN request bit. ⌋ ( )

**[FRNM422]** ⌈The FrNm shall only consider the PN request bits in the NM PDUs that are relevant for the ECU (defined by PN filter mask). All other PN request bits shall be ignored. Thus the EIRA only contains those PN requests, which are relevant for the ECU. ⌋ ( )

**[FRNM423]** ⌈If a NM-PDU is received, the FrNm shall set for every requested and relevant PN request the corresponding EIRA bit to 1. ⌋ ( )

**[FRNM424]** ⌈If a NM PDU is send by the FrNm, FrNm shall set every PN request bit in the EIRA to 1 that is also requested in its transmitted NM PDU. ⌋ ( )

**[FRNM425]** ⌈FrNm shall provide an EIRA reset timer for every PN request bit together for all physical FlexRay channels. ⌋ ( )

**Note**: This means, only one timer is required to handle one PN on multiple connected physical channels. For example: only 8 EIRA reset timers are required to handle the requests of a Gateway with 6 physical channels and 8 partial networks.
This is possible because the external request is mirrored back to the requesting bus and provided to all other (required) physical channels. Thus it is not required to detect the physical channel that is the source of the request bit.

**[FRNM426]** ⌈If a NM-PDU is received, the FrNm module shall restart the EIRA reset timer for every PN request bit that has been requested in the received NM-PDU with `FrNmPnResetTime`.⌋( )

**[FRNM427]** ⌈If a NM-PDU is send by the FrNm, the FrNm module shall restart the EIRA reset timer for every PN request bit that has been requested in its transmitted NM PDU with `FrNmPnResetTime`.⌋( )

**Note**: `FrNmPnResetTime` should be configured to a value greater than FlexRay NM cycle time.
Rational: If `FrNmPnResetTime` is configurd to a value smaller than FlexRay NM cycle time and only one ECU requests the PN, the request state toggles in the EIRA because request state is rested before the requesting ECU is able to send the next NM message.

**[FRNM428]** ⌈If an EIRA reset timer expires, the FrNm module shall set the corresponding EIRA PN request bit to `0`.⌋( )

**[FRNM429]** ⌈If the content of EIRA changes (any bit changes from `1` to `0` or from `0` to `1`) because of a received or transmitted NM PDU or an EIRA reset timer expiration, FrNm shall inform the upper layers by calling `PduR_FrNmRxIndication()`. By means of the Rx Indication function the EIRA data shall be provided to the COM module.⌋( )

#### 7.8.12.4 Aggregation of external requested PNs

**Note**: This feature is used by the Gateways to collect **only the external** PN requests. The external PN Requests are mirrored back to the requesting bus and provided to other (required) physical channels in Case of Central-Gateway. In Case of a Sub-Gateway the requests bit mustn't be mirrored back to the requesting physical channel to avoid static waking between Central-Gateway and Sub-Gateways. This logic shall be implemented by an upper layer (e.g.:ComM). The FrNm only provides the information if the PN is externally requested or not. The COM is used for data transmission to the upper layer.

**[FRNM430]** ⌈If `FrNmPnEraCalcEnabled` is `TRUE`, FrNm shall calculate the aggregation of external PN requests information.⌋( )

**[FRNM431]** ⌈If `FrNmPnEraCalcEnabled` is `FALSE`, FrNm shall skip the aggregation of external PN requests.⌋( )

The aggregated state of the external requested PNs is called "External Requests Aggregated" (ERA).

**[FRNM432]** ⌈The ERA module shall have a size of `FrNmPnInfoLength` and shall be initialized with value `0` (not requested) for every PN request bit.⌋( )

**[FRNM433]** ⌈The FrNm shall only consider the PN request bits in the NM-PDU that are relevant for the ECU (defined by PN filter mask). All other PN request bits are ignored. Thus the ERA only contains those PN requests, which are relevant for the ECU.⌋( )

**[FRNM434]** ⌈If a NM-PDU is received the FrNm module shall set every PN request bit in the ERA to 1 that has been requested by the PN request bits of the received NM-PDU.⌋( )

**[FRNM435]** ⌈The FrNm module shall provide an ERA reset timer for every PN request bit for every physical Fr channel.⌋( )

**Note**: This means, a separate timer is required to handle one PN on multiple connected physical channels. For example: 48 ERA reset timers are required to handle the requests of a Gateway with 6 physical channels and 8 partial networks. It is not possible to combine the reset timer like EIRA timers, because the external request mustn't be mirrored back to the requesting bus by a Sub-Gateway. Thus it is required to detect the physical channel that is the source of the request bit.

**[FRNM436]** ⌈If a NM PDU is received, FrNm shall start/restart the ERA reset timer for every PN request bit that is requested in the NM PDU with `FrNmPnResetTime`.⌋( )

**Note**: `FrNmPnResetTime` should be configured to a value greater than FlexRay NM cycle time.
Rational: If `FrNmPnResetTime` is configurd to a value smaller than FlexRay NM cycle time and only one ECU requests the PN, the request state toggles in the ERA because request state is rested before the requesting ECU is able to send the next NM message.

**[FRNM437]** ⌈If an ERA reset timer expires, the FrNm module shall set the corresponding ERA PN request bit to `0`.⌋( )

**[FRNM438]** ⌈If the content of ERA changes (any bit changes from `1` to `0` or from `0` to `1`) because of a received NM PDU or an ERA reset timer expiration, FrNm shall inform the upper layers by calling `PduR_FrNmRxIndication()`. By means of the Rx Indication function the ERA data shall be provided to the COM module.⌋( )

## 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 seven 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 FlexRay Channel A, FlexRay Channel B or on both FlexRay Channels. For the dynamic segment a transmission on both FlexRay channels requires two transmission- and receive-buffers as the FlexRay Protocol does not support shared transmission on FlexRay 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 7.8.9).

**[FRNM233]** ⌈The schedule variant for the FrNm module shall be statically configurable at pre-compile time by the configuration parameter **`FRNM_PDU_SCHEDULE_VARIANT`** (see chapter 10).⌋ ( )

**[FRNM234]** ⌈In the case when a combined NM-Vote and CBV is transmitted in static and dynamic segment (option 7 in Table 7-5), the FrNm module shall use the combined NM-Vote and CBV in the static part for the 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 FrNm module's integrator shall define the FlexRay NM Voting Cycle (**`FrNmVotingCycle`** configuration parameter) as the number of cycles needed to transmit—at least once—the NM-Vote of every node.⌋( )

Note: The value of the NM-Voting Cycle is typically determined by the number of cycles needed to transmit the votes of all nodes. For example, if only one slot in the dynamic segment is used, then the 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 119).

**[FRNM194]** ⌈The FrNm 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, then the NM-Data Cycle is set to 8 – see also FRNM195 and Figure 10-4 (on page 120).

**[FRNM195]** ⌈The FlexRay NM schedule specific cycle configuration parameters **`FrNmVotingCycle`**, **`FrNmDataCycle`** and **`FrNmRepetitionCycle`** shall have a value chosen from among the values 1, 2, 4, 8, 16, 32 or 64.⌋( )

Rationale: The restriction of the configuration values to the mentioned values is because FlexRay Cycle Multiplexing is used, which are 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 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 119).

### 7.9.2 NM-Message scheduled requirements

There are no scheduling requirements for the FlexRay NM messages. Anyhow, it is highly recommended for frames containing NM information that 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-messages are sent then less bandwidth is consumed)
- Flexibility (different nodes can transmit in the same slot but using 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 Debugging concept

**[FRNM345]** ⌈Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable. ⌋ ( )

**[FRNM346]** ⌈All type definitions of variables which shall be debugged, shall be accessible by the header file FrNm.h. ⌋ ( )

**[FRNM347]** ⌈The declaration of variables in the header file shall be such, that it is possible to calculate the size of the variables by C-“sizeof”. ⌋ ( )

**[FRNM348]** ⌈Variables available for debugging shall be described in the respective Basic Software Module Description. ⌋ ( )

**[FRNM349]** ⌈The FrNm states BusSleep, Synchronize, Repeat Message, Normal Operation, and Ready Sleep shall be available for debugging. ⌋ ( )

## 7.11 Transmission Error Handling

**[FRNM035]** ⌈When periodic NM-message transmissions are expected and when those NM-messages are not transmitted within a time interval given by **`FrNmMsgTimeoutTime`** (configuration parameter), the FlexRay NM shall notify the Generic NM Interface module that a transmission failure has occurred; this notification occurs by calling **`Nm_TxTimeoutException`**. ⌋ (BSW053, BSW137)

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”.

## 7.12 Error classification

**[FRNM294]** ⌈Values for production code Event Ids are externally assigned during 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_UINIT | 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 |
| API service called with invalid PDU ID as input parameter | Development | FRNM_E_PDU_ID_INVALID | 04h |

⌋(BSW00337, BSW00331)

## 7.13 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. ⌋(BSW00338)

**[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. ⌋(BSW00369)

**[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`**, when applicable. ⌋(BSW00369)

**[FRNM050]** ⌈If DET is enabled and APIs other than FrNm_Init or FrNm_GetVersionInfo are called when FrNm has not been initialized, then the called function shall not be executed and shall return NM_E_NOT_OK (if possible) and the FrNm module shall report the error code FRNM_E_UNINIT to the Development Error Tracer.⌋( )

**[FRNM051]** ⌈If DET is enabled and the FlexRay NM API service is called with an invalid handle, then the corresponding function shall report FRNM_E_INVALID_CHANNEL error to the Development Error Tracer and it shall return NM_E_NOT_OK.⌋( )

**[FRNM295]** ⌈The detection of production code errors cannot be switched off.⌋( )

## 7.14 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).⌋(BSW00338)

**[FRNM023]** ⌈Production errors shall be reported to the Diagnostic Event Manager.⌋(BSW00339)

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

Note: Currently no production errors are defined.

## 7.15 Version check

**[FRNM074]** ⌈The FrNm module shall perform Inter Module Checks to avoid integration of incompatible files. The imported included files shall be checked by preprocessing directives.⌋(BSW004)

The following version numbers shall be verified:
- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_RELEASE_MINOR_VERSION

Where <MODULENAME> is the module short name of the other (external) modules which provide header files included by the FrNm module.

If the values are not identical to the expected values, an error shall be reported.

## 7.16 Send ActiveWakeupBit in CBV

To identify the ECU that awakes the Network, the FrNm shall send an information with every NM message whehter the ECU is responsible for the communication start or has been woken up by the communication. This information is pending until the ECU leaves the “Network Mode”.

**FRNM297** If the FrNm performs a state change from state Synchronize to state NetworkMode and the previous state change from state Bus Sleep Mode to Syncronize was caused by a call of FrNm_NetworkRequest() (due to an active wakeup), the FrNm shall set the ActiveWakeupBit in the CBV.

**FRNM298** If the FrNm leaves the NetworkMode, the FrNm shall reset the ActiveWakeupBit in the CBV.

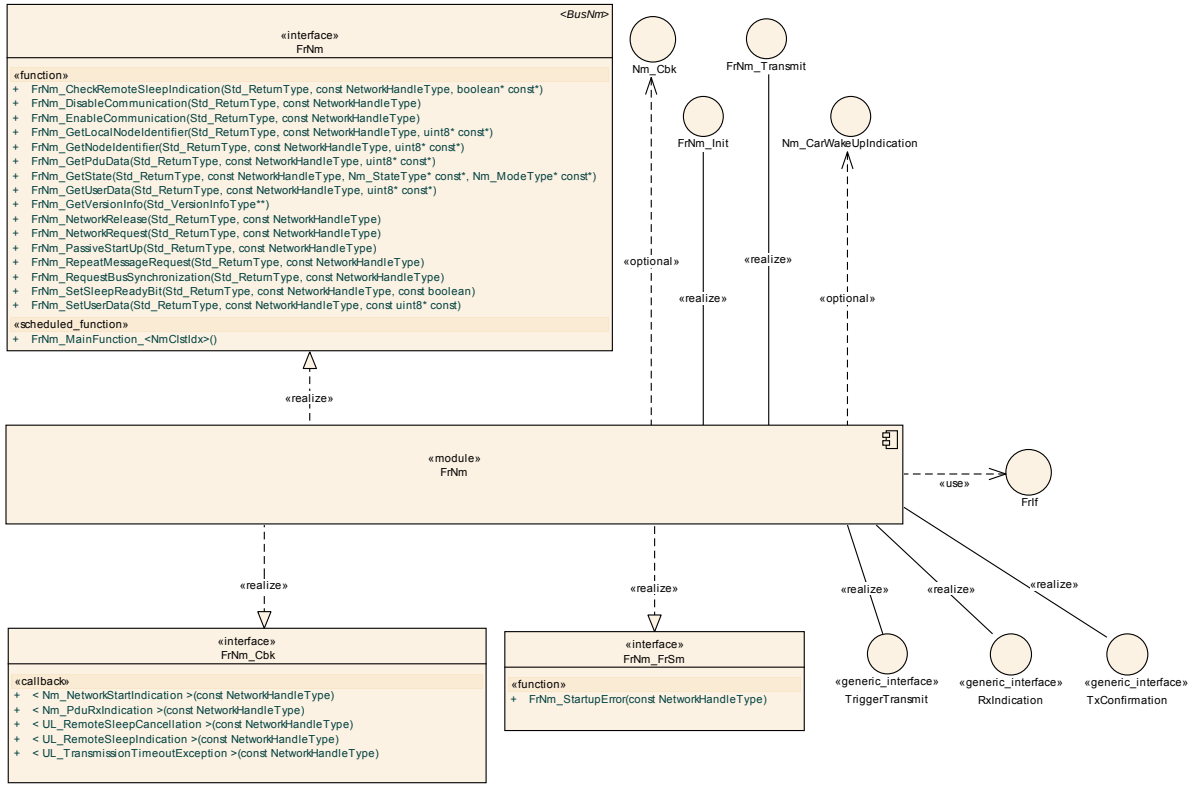
# 8 API specification

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. ⌋ (BSW045, BSW154)

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 interface line. FlexRay offers two channels per Communication controller and these FlexRay channels are NOT independent, and can therefore should not confused with an independent “NM cluster”.

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




API Specification (Overview)

## 8.1 Imported types

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

**[FRNM235]** ⌈

| *Module* | *Imported Type* |
|---|---|
| ComStack_Types | PduIdType |
| | PduInfoType |

| | |
|---|---|
| | NetworkHandleType |
| Dem | Dem_EventIdType |
| | Dem_EventStatusType |
| Nm | Nm_ModeType |
| | Nm_StateType |
| Std_Types | Std_ReturnType |
| | Std_VersionInfoType |

⌋ ( )

## 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 ([7]) 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 104.

## 8.3 Function definitions: FrNm Services provided to Generic NM Interface Module

### 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 FrNm module.⌋(BSW101)

**[FRNM073]** ⌈After a successful initialization of the FrNm module, the function **`FrNm_Init`** shall set the initialization status to **`NM_INIT`**, otherwise to **`NM_UNINIT`**.⌋(BSW00406)

**[FRNM059]** ⌈The function **`FrNm_Init`** shall select an 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 FrNm module (see FRNM100, FRNM137).⌋( )

**[FRNM042]** ⌈After the initialization of the FrNm 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 FrNm module the function **`FrNm_Init`** shall set the User Data bytes to FFh.⌋( )

**[FRNM395]** ⌈If parameter nmConfigPtr of FrNm_Init equals NULL_PTR and if development error detection is enabled (i.e. FRNM_DEV_ERROR_DETECT equals ON), the function FrNm_Init shall report development error code FRNM_E_INVALID_POINTER to the Det_ReportError service of the DET module.⌋( )

### 8.3.2 FrNm_PassiveStartUp

**[FRNM237]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_PassiveStartUp | |
| ***Syntax:*** | `Std_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:*** | Std_ReturnType | E_OK: No error<br>E_NOT_OK: Start of network management has failed |
| ***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 FrNm module if the current state is not Bus-Sleep Mode. In this case the function FrNm_PassiveStartUp shall return `NM_E_NOT_OK` (see FRNM138). ⌋ ( )

### 8.3.3 FrNm_NetworkRequest

**[FRNM238]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_NetworkRequest | |
| ***Syntax:*** | `Std_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:*** | Std_ReturnType | E_OK: No error<br>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 FrNm module’s environment calls the function **`FrNm_NetworkRequest`** in the Bus-Sleep Mode, then the function **`FrNm_NetworkRequest`** shall cause the FlexRay NM to leave the Bus-Sleep Mode, set the Network Request Flag **`FrNm_NetworkRequested`** to TRUE and then enter the Synchronize Mode. ⌋ ( )

**[FRNM141]** ⌈If the FrNm module’s environment calls the function **`FrNm_NetworkRequest`** in the Synchronize Mode, then the function **`FrNm_NetworkRequest`** shall set the Network Request Flag **`FrNm_NetworkRequested`** to TRUE. ⌋ ( )

**[FRNM113]** ⌈If the FrNm module's environment calls the function **`FrNm_NetworkRequest`** in the Network Mode, then 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_MODE_ENABLED is set to OFF.⌋( )

### 8.3.4 FrNm_NetworkRelease

**[FRNM239]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_NetworkRelease | |
| ***Syntax:*** | `Std_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:*** | Std_ReturnType | E_OK: No error<br>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 FrNm module's environment calls the function **`FrNm_NetworkRelease`** in the Synchronize Mode, then the function **`FrNm_NetworkRelease`** shall set the Network Request Flag **`FrNm_NetworkRequested`** to FALSE.⌋( )

**[FRNM114]** ⌈If the FrNm module's environment calls the function **`FrNm_NetworkRelease`** in the Network Mode, then 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_MODE_ENABLED is set to OFF.⌋( )

### 8.3.5 FrNm_SetUserData

**[FRNM240]** ⌈

| | |
|---|---|
| ***Service name:*** | FrNm_SetUserData |

| | | |
|---|---|---|
| ***Syntax:*** | `Std_ReturnType FrNm_SetUserData(`<br>`  const NetworkHandleType NetworkHandle,`<br>`  const uint8* const nmUserDataPtr`<br>`)` | |
| ***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:*** | Std_ReturnType | E_OK: No error<br>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 FrNm module, then the function **`FrNm_SetUserData`** shall set the user data. ⌋(BSW02503)

**[FRNM319]** ⌈If FrNm_SetUserData is called and the FrNm module is in UnInit state, then NM_E_NOT_OK shall be returned. ⌋( )

**[FRNM263]** ⌈The function FrNm_SetUserData shall be only available if the configuration parameter FrNmUserDataEnabled is set as ON (see FRNM077). ⌋( )

### 8.3.6 FrNm_GetUserData

**[FRNM241]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_GetUserData | |
| ***Syntax:*** | `Std_ReturnType FrNm_GetUserData(`<br>`  const NetworkHandleType NetworkHandle,`<br>`  uint8* const nmUserDataPtr`<br>`)` | |
| ***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:*** | Std_ReturnType | E_OK: No error<br>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 FrNm module, then the function **`FrNm_GetUserData`** shall provide the user data from the last received NM-Data PDU. ⌋(BSW02504)

**[FRNM264]** ⌈The function FrNm_GetUserData shall be only available if the configuration parameter FrNmUserDataEnabled is set as ON (see FRNM077). ⌋( )

### 8.3.7 FrNm_GetPduData

**[FRNM242]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_GetPduData | |
| ***Syntax:*** | `Std_ReturnType FrNm_GetPduData(`<br>`  const NetworkHandleType NetworkHandle,`<br>`  uint8* const nmPduData`<br>`)` | |
| ***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:*** | Std_ReturnType | E_OK: No error<br>E_NOT_OK: Getting of NM PDU data has failed |
| ***Description:*** | Gets PDU data. | |

⌋( )

**[FRNM265]** ⌈The function **`FrNm_GetPduData`** shall get the whole NM PDU data out of the most recently received NM message. ⌋( )

**[FRNM266]** ⌈The function **`FrNm_GetPduData`** shall only be available if the configuration parameter **`FrNmControlBitVectorEnabled`** or **`FrNmSourceNodeIdentifierEnabled`** or **`FrNmUserDataEnabled`** is set to ON. ⌋( )

### 8.3.8 FrNm_RepeatMessageRequest

**[FRNM243]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_RepeatMessageRequest | |
| ***Syntax:*** | `Std_ReturnType FrNm_RepeatMessageRequest(`<br>`  const NetworkHandleType NetworkHandle`<br>`)` | |
| ***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:*** | Std_ReturnType | E_OK: No error<br>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 the node detection feature is enabled, then the function **`FrNm_RepeatMessageRequest`** shall request node detection on the FlexRay Bus NM nodes. ⌋( )

**[FRNM226]** ⌈If the FrNm module's environment is calling the function **`FrNm_RepeatMessageRequest`** in the Network Mode and if the Node detection service is activated (**`FrNmNodeDetectionEnabled`**), then 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:*** | `Std_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:*** | Std_ReturnType | E_OK: No error<br>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 the node detection feature is enabled, then the function **`FrNm_GetNodeIdentifier`** shall provide the node identifier from the most recently received NM-message. ⌋(BSW02506)

**[FRNM267]** ⌈The function FrNm_GetNodeIdentifier shall be only available if the configuration parameter FrNmSourceNodeIdentifierEnabled is set as ON (see FRNM170). ⌋( )

### 8.3.10 FrNm_GetLocalNodeIdentifier

**[FRNM245]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_GetLocalNodeIdentifier | |
| ***Syntax:*** | `Std_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:*** | Std_ReturnType | E_OK: No error<br>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, then 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 FrNmSourceNodeIdentifierEnabled is set as ON (refer to FRNM170). ⌋ ( )

### 8.3.11 FrNm_RequestBusSynchronization

**[FRNM246]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_RequestBusSynchronization | |
| ***Syntax:*** | `Std_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:*** | Std_ReturnType | E_OK: No error<br>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 FrNm 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 FrNm cannot influence the bus synchronization. This functionality is handled by the FlexRay Controller and FlexRay Driver. Since this function might be used in future extensions (e.g., Gateways) this function shall remain available.

**[FRNM269]** ⌈The function FrNm_RequestBusSynchronization shall be only available if the configuration parameter FrNmBusSynchronizationEnabled is set as ON.⌋( )

### 8.3.12 FrNm_CheckRemoteSleepIndication

**[FRNM247]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_CheckRemoteSleepIndication | |
| ***Syntax:*** | `Std_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:*** | Std_ReturnType | E_OK: No error<br>E_NOT_OK: Checking of remote sleep indication bits has failed |
| ***Description:*** | This function checks if remote sleep indication has taken place or not. | |

⌋( )

**[FRNM270]** ⌈The FrNm module and the Nm module shall be initialized correctly before the FrNm 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., whether a Remote Sleep Indication has already been detected or not).⌋(BSW02509)

**[FRNM271]** ⌈The function FrNm_CheckRemoteSleepIndication shall be only available if the configuration parameter FrNmRemoteSleepIndicationEnabled is set as ON.⌋( )

### 8.3.13 FrNm_GetState

**[FRNM248]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_GetState | |
| ***Syntax:*** | `Std_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:*** | Std_ReturnType | E_OK: No error<br>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. ⌋ (BSW050)

Note: Consistency between the provided values and the current values of the state and mode should be ensured.

### 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 the API FrNm_GetVersionInfo is available, then the FrNm 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 as ON. ⌋( )

### 8.3.15 FrNm_StartupError

8.3.16 [FRNM393] ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_StartupError | |
| ***Syntax:*** | `void FrNm_StartupError(`<br>`  const NetworkHandleType NetworkHandle`<br>`)` | |
| ***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:*** | None | |
| ***Description:*** | This function is called by the FrSM when synchronization of the FlexRay cluster could not be achieved. | |

### 8.3.17 FrNm_Transmit

**[FRNM359]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_Transmit | |
| ***Syntax:*** | `Std_ReturnType FrNm_Transmit(`<br>`  PduIdType FrNmTxPduId,`<br>`  const PduInfoType* PduInfoPtr`<br>`)` | |
| ***Service ID[hex]:*** | 0x11 | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Reentrant | |
| ***Parameters (in):*** | FrNmTxPduId | L-PDU handle of FlexRay L-PDU to be transmitted.<br>This handle specifies the corresponding FlexRay L-PDU ID and implicitly the FlexRay Driver instance as well as the corresponding FlexRay controller device. |
| | PduInfoPtr | Pointer to a structure with FlexRay L-PDU related data: DLC and pointer to FlexRay L-SDU buffer. |

| | | |
|---|---|---|
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | Std_ReturnType | E_OK: Transmit request has been accepted<br>E_NOT_OK: Transmit request has not been accepted (FrNm is not in RM or NO) |
| ***Description:*** | This function is used by the PduR to trigger a spontaneous transmission of an NM message with the provided NM User Data.<br>On FlexRay it is not possible to send spontaneous NM messages. So this function is only a dummy to avoid linker errors of the PduR and standardize the NM handling between FrNm and CanNm. | |

⌋( )

### 8.3.18 FrNm_EnableCommunication

**[FRNM387]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_EnableCommunication | |
| ***Syntax:*** | `Std_ReturnType FrNm_EnableCommunication( const NetworkHandleType nmChannelHandle )` | |
| ***Service ID[hex]:*** | 0x05 | |
| ***Sync/Async:*** | Asynchronous | |
| ***Reentrancy:*** | Reentrant (but not for the same NM-channel) | |
| ***Parameters (in):*** | nmChannelHandle | Identification of the NM-channel |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | Std_ReturnType | E_OK: No error<br>E_NOT_OK: Enabling of NM PDU transmission ability has failed |
| ***Description:*** | Enable the NM PDU transmission ability due to a ISO14229 Communication Control (28hex) service | |

⌋( )

**[FRNM388]** ⌈The service FrNm_EnableCommunication shall return NM_E_NOT_OK, if the current mode is not Network Mode. ⌋( )

**[FRNM389]** ⌈If the module operates in passive mode (FRNM_PASSIVE_MODE_ENABLED), then the service FrNm_EnableCommunication shall have no effects and directly return NM_E_NOT_OK. ⌋( )

### 8.3.19 FrNm_DisableCommunication

**[FRNM390]** ⌈

| | |
|---|---|
| ***Service name:*** | FrNm_DisableCommunication |

| | | |
|---|---|---|
| ***Syntax:*** | `Std_ReturnType FrNm_DisableCommunication( const NetworkHandleType nmChannelHandle )` | |
| ***Service ID[hex]:*** | 0x0c | |
| ***Sync/Async:*** | Asynchronous | |
| ***Reentrancy:*** | Reentrant (but not for the same NM-channel) | |
| ***Parameters (in):*** | nmChannelHandle | Identification of the NM-channel |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | Std_ReturnType | E_OK: No error<br>E_NOT_OK: Disabling of NM PDU transmission ability has failed |
| ***Description:*** | Disable the NM PDU transmission ability due to a ISO14229 Communication Control (28hex) service | |

⌋( )

**[FRNM391]** ⌈The service FrNm_DisableCommunication shall return NM_E_NOT_OK if the current mode is not Network Mode.⌋( )

**[FRNM392]** ⌈If the module operates in passive mode (FRNM_PASSIVE_MODE_ENABLED), then the service FrNm_DisableCommunication shall have no effect and directly return NM_E_NOT_OK.⌋( )

### 8.3.20 FrNm_SetSleepReadyBit

| | | |
|---|---|---|
| ***Service name:*** | FrNm_SetSleepReadyBit | |
| ***Syntax:*** | `Std_ReturnType FrNm_SetSleepReadyBit( const NetworkHandleType nmChannelHandle, const boolean nmSleepReadyBit )` | |
| ***Service ID[hex]:*** | 0x12 | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Non Reentrant | |
| ***Parameters (in):*** | nmChannelHandle | Identification of the NM-channel |
| | nmSleepReadyBit | Value written to ReadySleep Bit in CBV |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | Std_ReturnType | E_OK: No error<br>E_NOT_OK: Writing of remote sleep indication bit has failed |
| ***Description:*** | Set the NM Coordinator Sleep Ready bit in the Control Bit Vector | |

⌋( )

## 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(`<br>`  PduIdType RxPduId,`<br>`  PduInfoType* PduInfoPtr`<br>`)` | |
| ***Service ID[hex]:*** | 0x42 | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Reentrant for different PduIds. Non reentrant for the same PduId. | |
| ***Parameters (in):*** | RxPduId | ID of the received I-PDU. |
| | PduInfoPtr | Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU. |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | None | |
| ***Description:*** | Indication of a received I-PDU from a lower layer communication module. | |

⌋ ( )

**[FRNM013]** ⌈The function **`FrNm_RxIndication`** shall handle the data from an NM-message—this means that the function shall copy the received FlexRay NM PDU and store it locally associated with the received FlexRay NM PDU ID. ⌋ ( )

**[FRNM276]** ⌈The FrIf module and the FrNm module must be initialized correctly before the FrNm module's environment calls the function FrNm_RxIndication. ⌋ ( )

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

### 8.4.2 FrNm_TriggerTransmit

**[FRNM252]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | FrNm_TriggerTransmit | |
| ***Syntax:*** | `Std_ReturnType FrNm_TriggerTransmit(`<br>`  PduIdType TxPduId,`<br>`  PduInfoType* PduInfoPtr`<br>`)` | |
| ***Service ID[hex]:*** | 0x41 | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Reentrant for different PduIds. Non reentrant for the same PduId. | |
| ***Parameters (in):*** | TxPduId | ID of the SDU that is requested to be transmitted. |
| | PduInfoPtr | Contains a pointer to a buffer (SduDataPtr) to where the SDU shall be copied to. On return, the service will indicate the length of the copied SDU data in SduLength. |

| | | |
|---|---|---|
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | Std_ReturnType | E_OK: SDU has been copied and SduLength indicates the number of copied bytes.<br>E_NOT_OK: No SDU has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data. |
| ***Description:*** | The lower layer communication module requests the buffer of the SDU for transmission from the upper layer module. | |

⌋( )

**[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 FrNm module's environment calls the function FrNm_TriggerTransmit.⌋( )

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

### 8.4.3 FrNm_TxConfirmation

| | | |
|---|---|---|
| ***Service name:*** | FrNm_TxConfirmation | |
| ***Syntax:*** | `void FrNm_TxConfirmation(`<br>`  PduIdType TxPduId`<br>`)` | |
| ***Service ID[hex]:*** | 0x40 | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Reentrant for different PduIds. Non reentrant for the same PduId. | |
| ***Parameters (in):*** | TxPduId | ID of the I-PDU that has been transmitted. |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | None | |
| ***Description:*** | The lower layer communication module confirms the transmission of an I-PDU. | |

## 8.5 Scheduled functions: FrNm Services provided to BSW Scheduler

### 8.5.1 FrNm_MainFunction_< FrNmChannelIdRef >

**[FRNM255]** ⌈

| | |
|---|---|
| ***Service name:*** | FrNm_MainFunction_<NmClstIdx> |
| ***Syntax:*** | `void FrNm_MainFunction_<NmClstIdx>(`<br>`  void`<br>`)` |

| | |
|---|---|
| ***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 called cyclically from a task body provided by the BSW Scheduler.

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

⌋( )

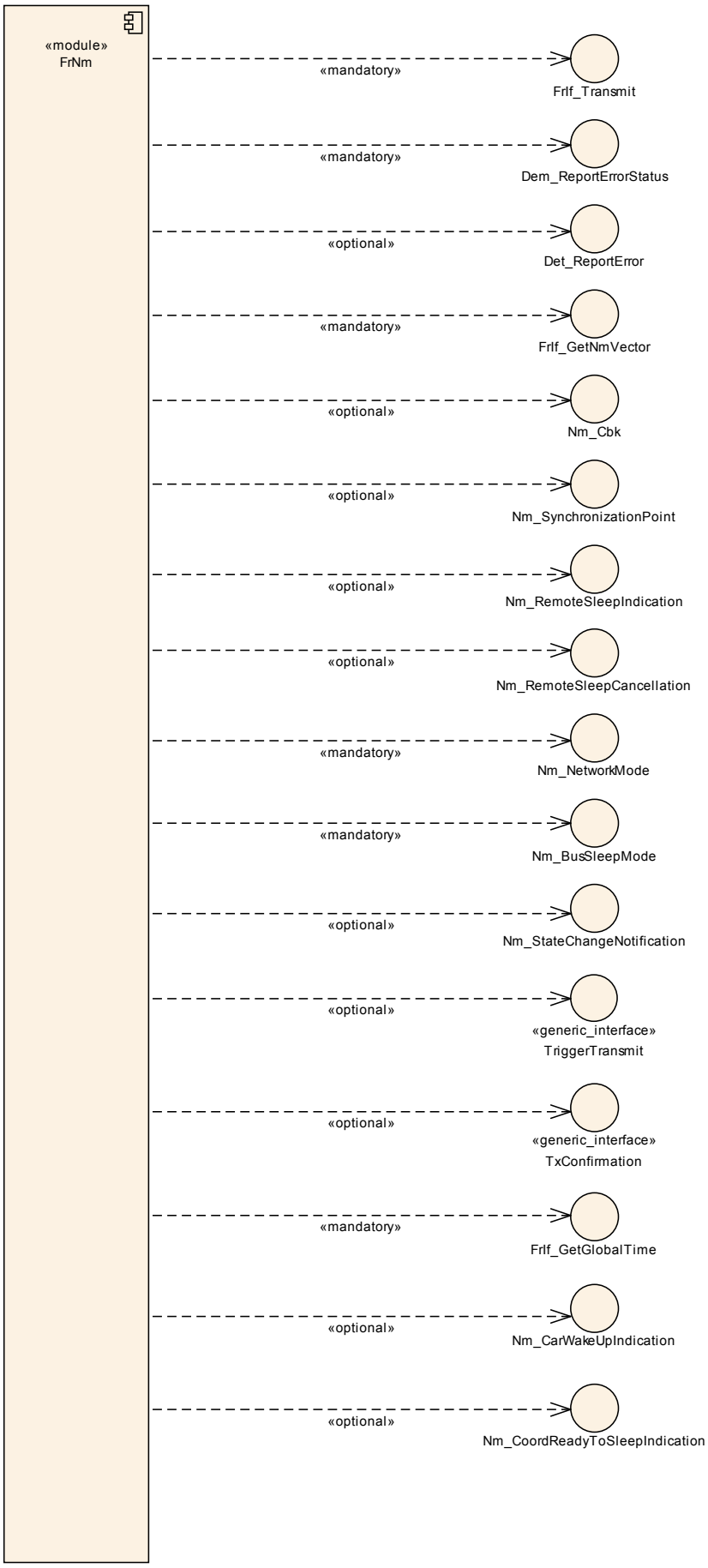
**[FRNM283]** ⌈There shall be one dedicated FlexRay NM Main Function for each NM cluster. The API names are therefore:
FrNm_MainFunction_0() for FlexRay NM cluster associated with FlexRay NM cluster 0
FrNm_MainFunction_1() for FlexRay NM cluster associated with FlexRay NM cluster 1
Etc. ⌋( )

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

**[FRNM344]** ⌈The Service ID of **FrNm_MainFunction_<NmClstIdx>** shall be 0xf0 + NmClstIdx. ⌋( )

## 8.6 Expected interfaces

In this chapter all interfaces required from other modules are listed.


«module»
FrNm
«mandatory»
FrIf_Transmit
«mandatory»
Dem_ReportErrorStatus
«optional»
Det_ReportError
«mandatory»
FrIf_GetNmVector
«optional»
Nm_Cbk
«optional»
Nm_SynchronizationPoint
«optional»
Nm_RemoteSleepIndication
«optional»
Nm_RemoteSleepCancellation
«mandatory»
Nm_NetworkMode
«mandatory»
Nm_BusSleepMode
«optional»
Nm_StateChangeNotification
«optional»
«generic_interface»
TriggerTransmit
«optional»
«generic_interface»
TxConfirmation
«mandatory»
FrIf_GetGlobalTime
«optional»
Nm_CarWakeUpIndication
«optional»
Nm_CoordReadyToSleepIndication

### 8.6.1 Mandatory Interfaces

This chapter defines all interfaces which are required to fulfill the core functionality of the module.

**[FRNM256]** ⌈

| *API function* | *Description* |
|---|---|
| Dem_ReportErrorStatus | Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. |
| FrIf_GetGlobalTime | Wraps the FlexRay Driver API function Fr_GetGlobalTime(). |
| FrIf_GetNmVector | Derives the FlexRay NM Vector. |
| FrIf_Transmit | Requests the sending of a PDU. |
| Nm_BusSleepMode | Notification that the network management has entered Bus-Sleep Mode. |
| Nm_NetworkMode | Notification that the network management has entered Network Mode. |

⌋ ( )
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 and FrSm. See [7].

### 8.6.2 Optional Interfaces

This chapter defines all interfaces which are required to fulfill an optional functionality of the module.

**[FRNM257]** ⌈

| *API function* | *Description* |
|---|---|
| Det_ReportError | Service to report development errors. |
| Nm_CarWakeUpIndication | This function is called by a <Bus>Nm to indicate reception of a CWU request. |
| Nm_CoordReadyToSleepIndication | Sets an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set |
| 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. |
| Nm_RemoteSleepCancellation | Notification that the network management has detected that not all other nodes on the network are longer ready to enter Bus-Sleep Mode. |
| Nm_RemoteSleepIndication | Notification that the network management has detected that all other nodes on the network are ready to enter Bus-Sleep Mode. |
| Nm_StateChangeNotification | Notification that the state of the lower layer <BusNm> has changed. |
| Nm_SynchronizationPoint | Notification to the NM Coordinator functionality that this is a suitable point in time to initiate the coordination algorithm on. |
| PduR_FrNmTriggerTransmit | The lower layer communication module requests the buffer of the SDU for transmission from the upper layer module. |

| | |
|---|---|
| PduR_FrNmTxConfirmation | The lower layer communication module confirms the transmission of an I-PDU. |

The usage of certain external APIs by FlexRay network Manangement is defined by configuration:

`FrIf_GetNmVector` Configured by configuration parameter FrIfGetNmVectorSupport (see FrIf06100_Conf)

`Nm_NetworkStartIndication` Configured if FrNmPduRxIndicationEnabled is configured (see FrNm0046_Conf)

`NM_PduRxIndication` Configured if FrNmPduRxIndicationEnabled is configured (see FrNm0046_Conf)

`Nm_RemoteSleepCancellation` Configured by FrNmRemoteSleepIndicationEnabled (see FrNm0044_Conf)

`Nm_RemoteSleepIndication` Configured by FrNmRemoteSleepIndicationEnabled (see FrNm0044_Conf)

`Nm_StateChangeNotification` Configured by FrNmStateChangeIndicationEnabled (see FrNm0047_Conf)

`Nm_SynchronizationPoint` Configured by FrNmSynchronizationPointEnabled (see FrNm0021_Conf)

`Nm_TxTimeoutException` Configured with the parameter FrNmMsgTimeoutTime (see FrNm0028_Conf). Nm_TxTimeoutException is available, if the value of FrNmMsgTimeoutTime is configured as greater than '0'(see FrNm0018_Conf)

⌋ ( )

### 8.6.3 Configurable interfaces

In this chapter all interfaces are listed where the target function could be configured. The target function is usually a call-back function. The names of these kind of interfaces is not fixed because they are configurable.

There are no configurable interfaces.

# 9 Sequence diagrams

## 9.1 Use Case 01 – Initialization

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

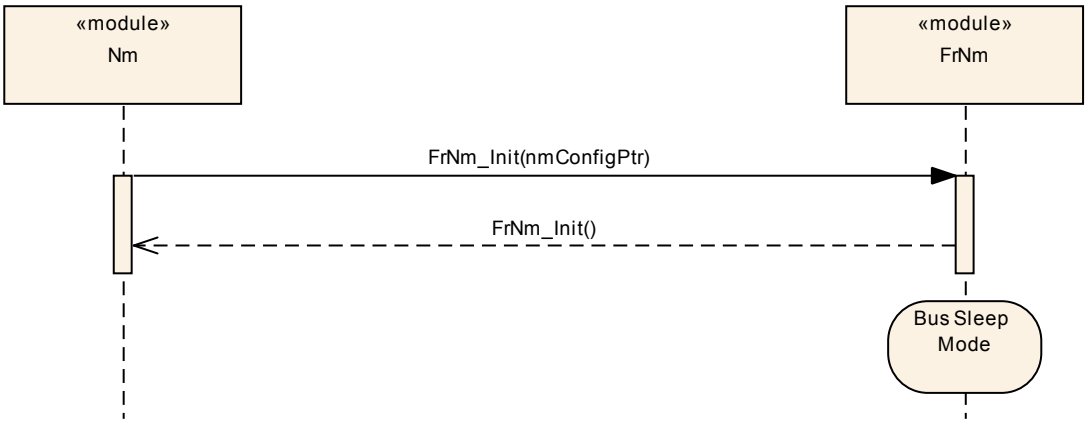



Figure 9-1 FrNm Init Sequence

## 9.2 Use Case 02 .- Passive Startup

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

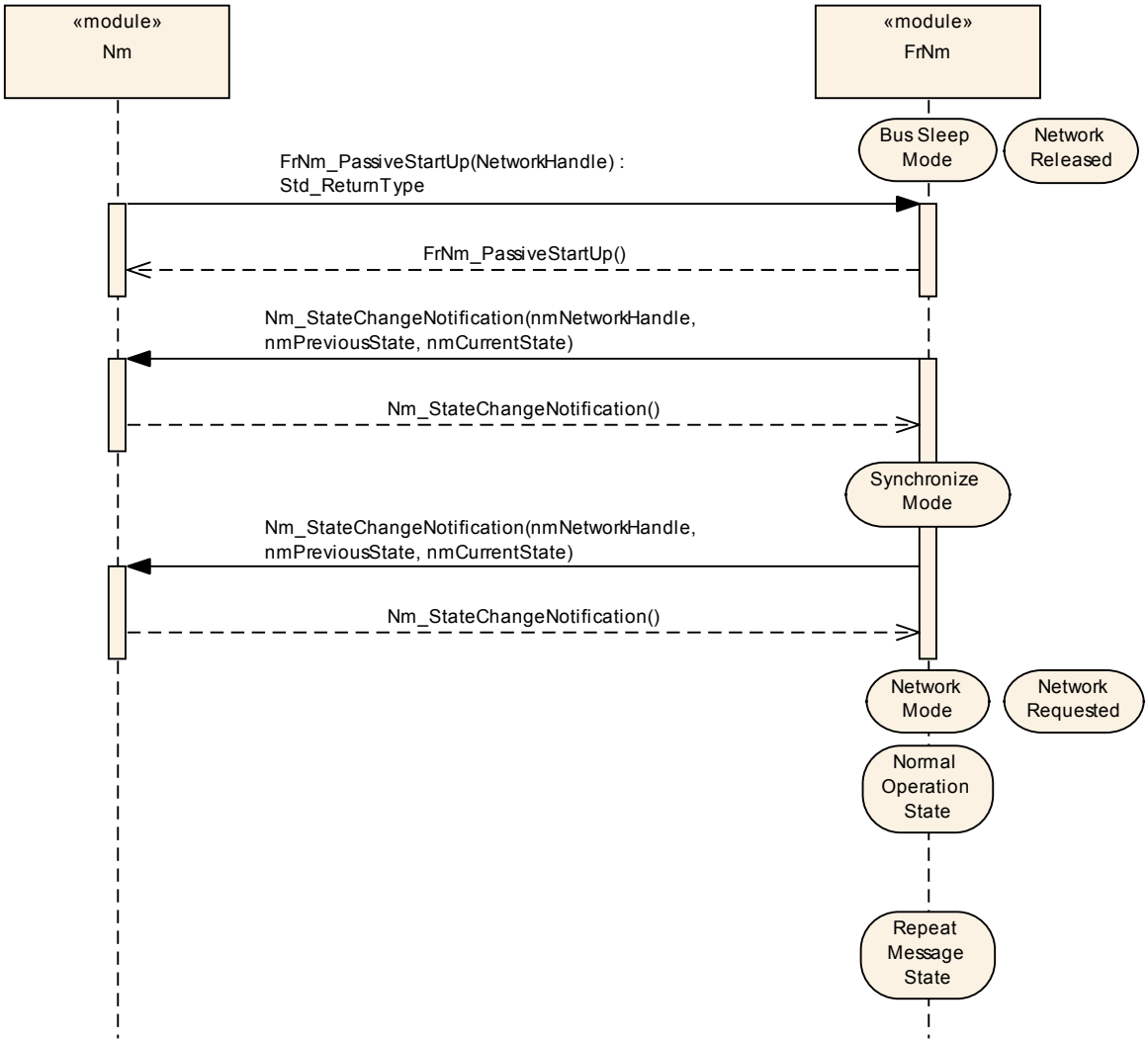



Figure 9-2 FrNm passive startup sequence

## 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.

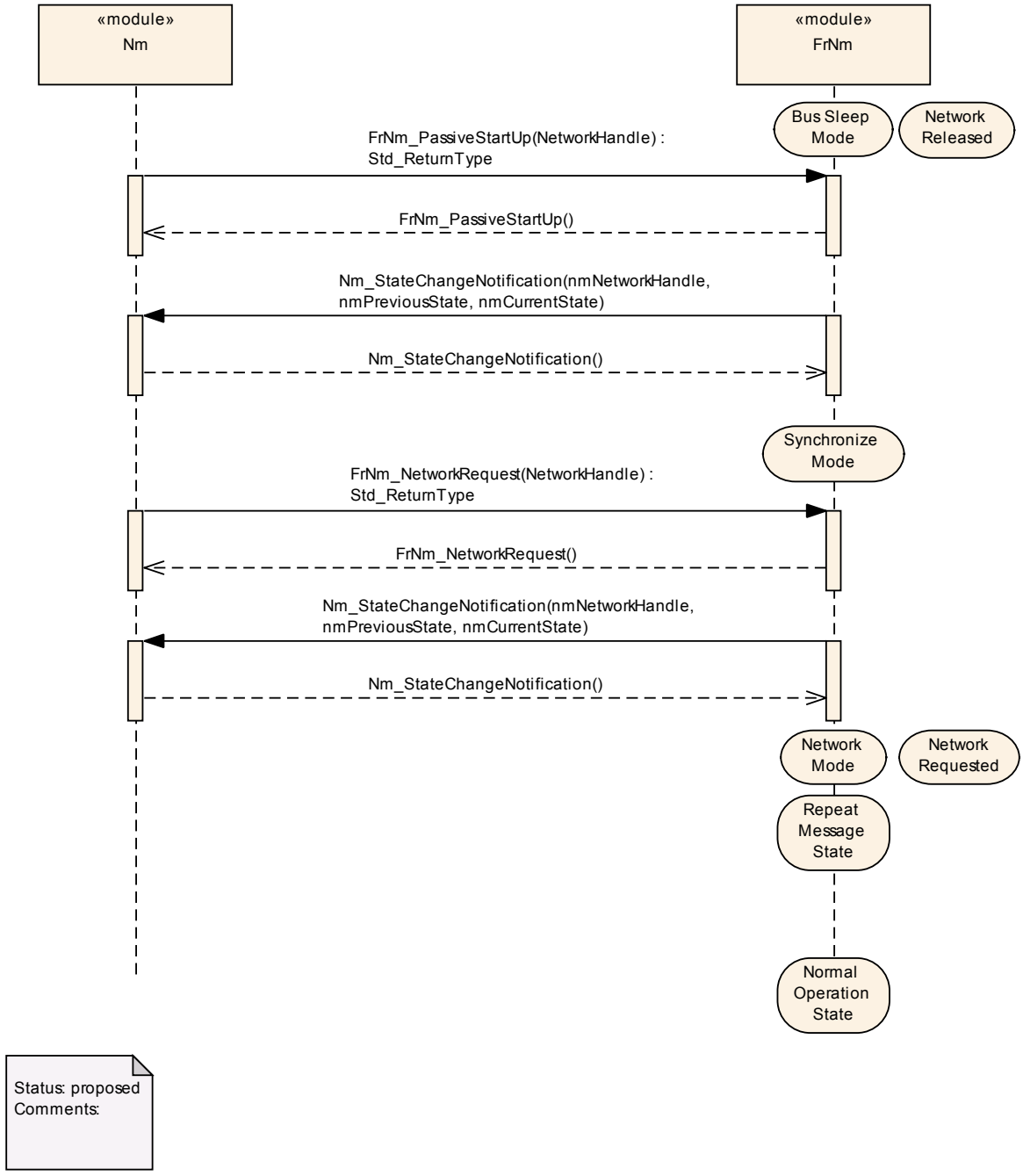



Figure 9-3 FrNm passive startup with a “active” network request sequence

## 9.4 Use Case 04 – Normal Operation

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

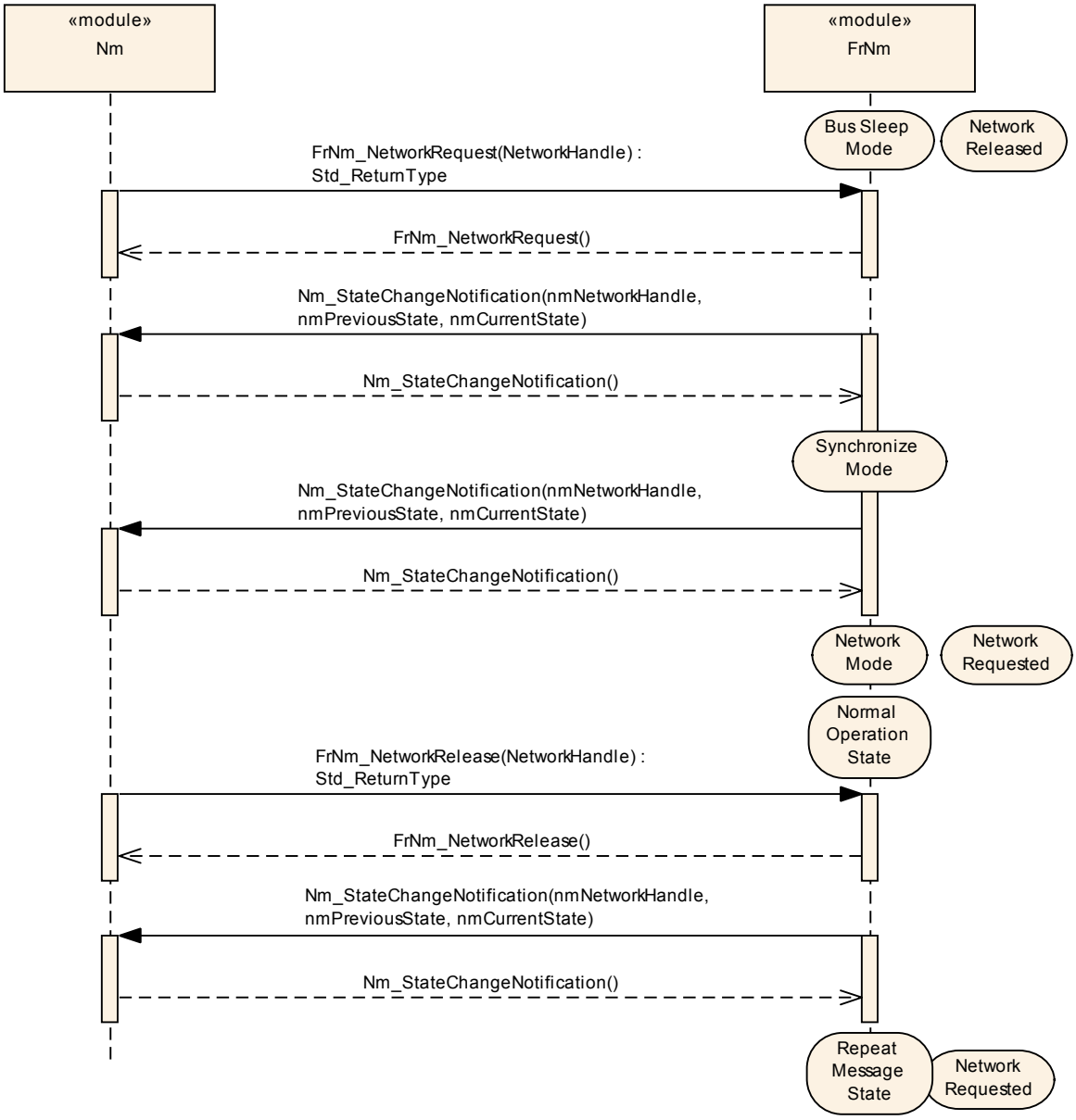



Figure 9-4 FrNm normal operation sequence

## 9.5 Use Case 05 – Shutdown

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

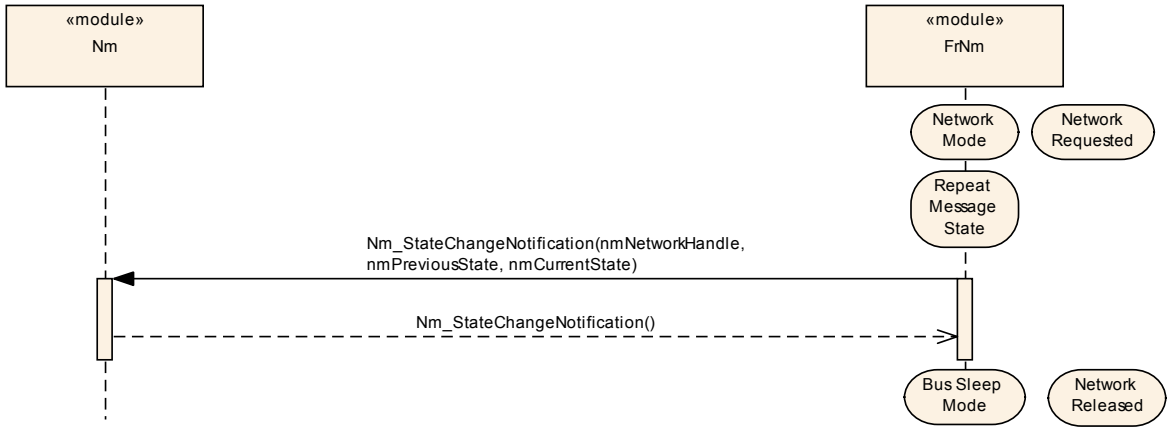



Figure 9-5 FrNm Shutdown

# 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.⌋ (BSW159)

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: VARIANT-PRE-COMPILE

**[FRNM290]** ⌈All configuration parameters are configured at pre-compile time.

Use case: Source code optimizations⌋( )

### 10.2.2 Variant 2: VARIANT-LINK-TIME

**[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.

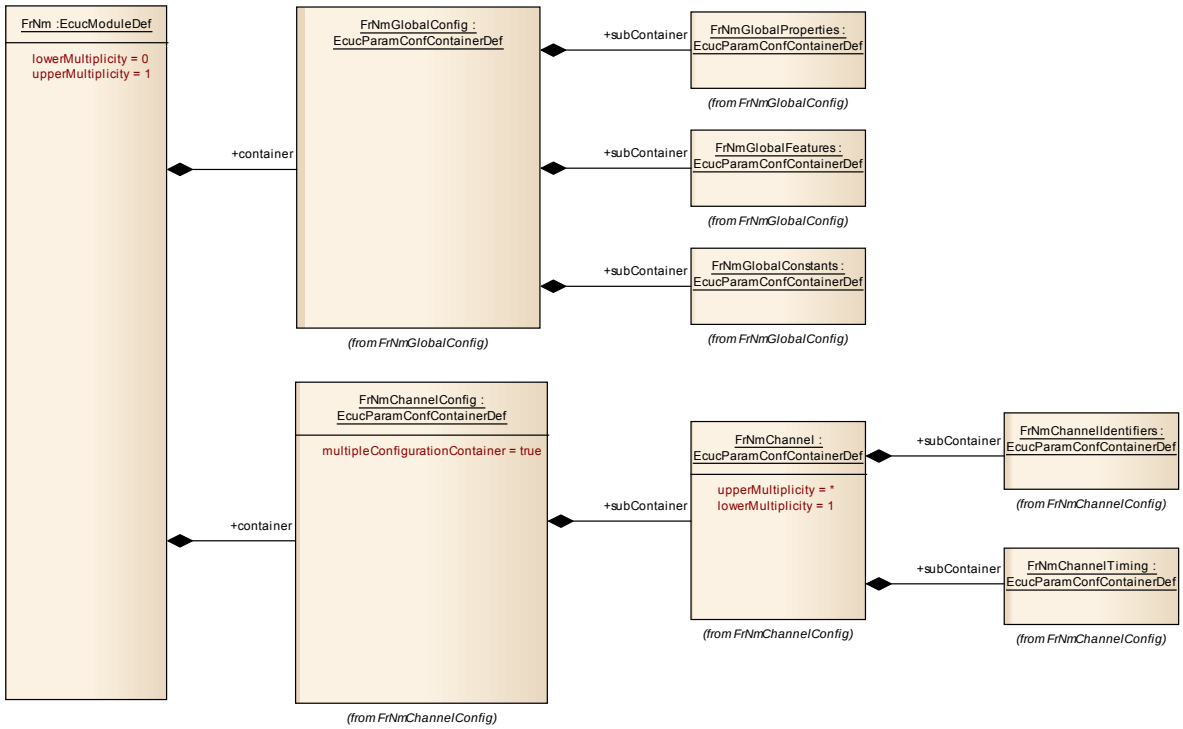
Use case: Object code libraries⌋( )

### 10.2.3 Variant 3: VARIANT-POST-BUILD

**[FRNM292]** ⌈The parameters contained in **`FrNmChannelConfig`** 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 the configuration parameters for all FlexRay NM channels. |
| FrNmGlobalConfig | 1 | This container contains all global configuration parameters for the FrNm module. |

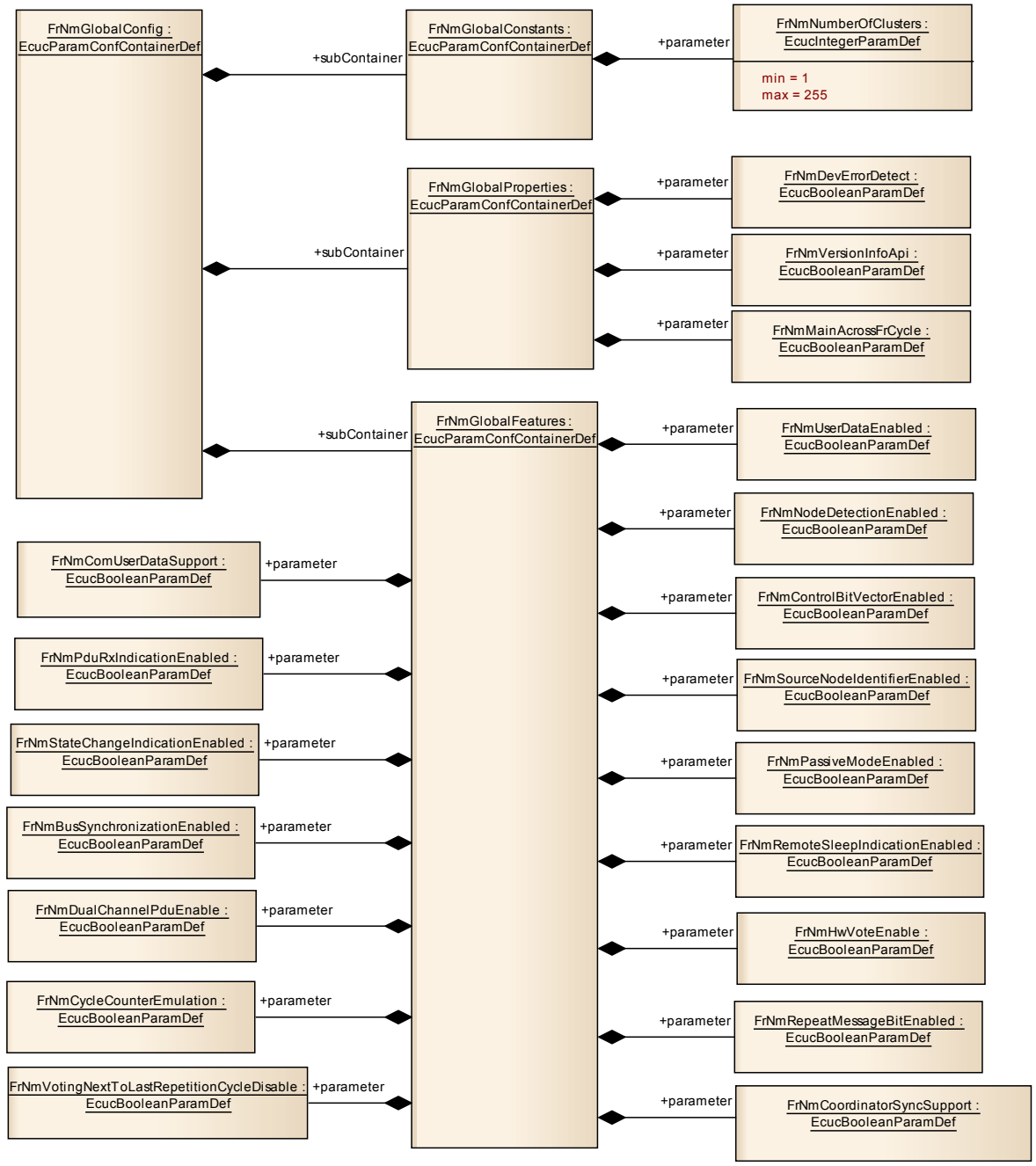
## 10.4 Global configurable parameters

**[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* | **FrNm0001_Conf :** |
|---|---|
| *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*** | **FrNm0005_Conf :** |
| ***Container Name*** | FrNmGlobalConstants{FrNm_GlobalConstants} |
| ***Description*** | This container contains module constants related to the FlexRay NM functionality. |
| ***Configuration Parameters*** | |

| | |
|---|---|
| ***SWS Item*** | **FrNm0034_Conf :** |
| ***Name*** | FrNmNumberOfClusters {FRNM_NUMBER_OF_CLUSTERS} |
| ***Description*** | Number of AUTOSAR FR NM clusters allowed within one ECU. |

| | | | |
|---|---|---|---|
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucIntegerParamDef | | |
| ***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*** | scope: module | | |

| |
|---|
| ***No Included Containers*** |

### 10.4.3 FrNmGlobalFeatures

| | |
|---|---|
| ***SWS Item*** | **FrNm0004_Conf :** |
| ***Container Name*** | FrNmGlobalFeatures{FrNm_GlobalFeatures} |
| ***Description*** | This container contains module features related to the FlexRay NM functionality. |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0048_Conf :** | | |
| ***Name*** | FrNmBusSynchronizationEnabled {FRNM_BUS_SYNCHRONIZATION_ENABLED} | | |
| ***Description*** | Pre-processor switch for enabling the bus synchronisation. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0054_Conf :** | | |
| ***Name*** | FrNmComUserDataSupport {FRNM_COM_USER_DATA_SUPPORT} | | |
| ***Description*** | Enable/disable the user data support. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0041_Conf :** | | |
| ***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*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |

| | | | |
|---|---|---|---|
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0081_Conf :** | | |
| ***Name*** | FrNmCoordinatorSyncSupport {FRNM_COORDINATOR_SYNC_SUPPORT} | | |
| ***Description*** | Enables/disables the coordinator synchronisation support. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0060_Conf :** | | |
| ***Name*** | FrNmCycleCounterEmulation {FRNM_CYCLE_COUNTER_EMULATION} | | |
| ***Description*** | Pre-processor switch for enabling the cycle counter emulation. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0049_Conf :** | | |
| ***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*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0050_Conf :** | | |
| ***Name*** | FrNmHwVoteEnable {FRNM_HW_VOTE_ENABLE} | | |
| ***Description*** | Pre-processor switch for enabling the processing of FlexRay Hardware aggregated NM-Votes. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | |
|---|---|
| ***SWS Item*** | **FrNm0040_Conf :** |
| ***Name*** | FrNmNodeDetectionEnabled {FRNM_NODE_DETECTION_ENABLED} |
| ***Description*** | Pre-processor switch for enabling node detection support.<br>calculationFormula = If (FrNmPassiveModeEnabled == False) then Equal(NmNodeDetectionEnabled) else Equal(False) |

| | | | |
|---|---|---|---|
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module<br>dependency: FrNmNodeDetectionEnabled should be identically TRUE or FALSE to FrNmSourceNodeIdentifierEnabled | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0043_Conf :** | | |
| ***Name*** | FrNmPassiveModeEnabled {FRNM_PASSIVE_MODE_ENABLED} | | |
| ***Description*** | Pre-processor switch for enabling Passive Mode Configuration support.<br>calculationFormula = Equal(NmPassiveModeEnabled) | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0046_Conf :** | | |
| ***Name*** | FrNmPduRxIndicationEnabled<br>{FRNM_PDU_RX_INDICATION_ENABLED} | | |
| ***Description*** | Pre-processor switch for enabling PDU reception indication. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0067_Conf :** | | |
| ***Name*** | FrNmPnEiraCalcEnabled {FRNM_PN_EIRA_CALC_ENABLED} | | |
| ***Description*** | Specifies if FrNm calculates the PN request information for internal an external requests. (EIRA) true: PN request are calculated false: PN request are not calculated | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | false | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module<br>dependency: only available if FrNmPnEnabled == true for at least one FrNm Channel | | |

| | |
|---|---|
| ***SWS Item*** | **FrNm0068_Conf :** |
| ***Name*** | FrNmPnResetTime {FRNM_PN_RESET_TIME} |
| ***Description*** | Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA. The value shall be the same for every channel. Thus it is a global config parameter. |
| ***Multiplicity*** | 1 |

| | | | |
|---|---|---|---|
| ***Type*** | EcucFloatParamDef | | |
| ***Range*** | `0.001 .. 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*** | scope: Module<br>dependency: only available if FrNmPnEnabled == true for at least one FrNm Channel. | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0044_Conf :** | | |
| ***Name*** | FrNmRemoteSleepIndicationEnabled<br>{FRNM_REMOTE_SLEEP_INDICATION_ENABLED} | | |
| ***Description*** | Pre-processor switch for enabling remote sleep indication.<br>calculationFormula = If (FrNmPassiveModeEnabled == True) then Equal(False) else Equal(False or True) | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0059_Conf :** | | |
| ***Name*** | FrNmRepeatMessageBitEnabled<br>{FRNM_REPEAT_MESSAGE_BIT_ENABLED} | | |
| ***Description*** | Pre-processor switch for enabling the repeat message bit support. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0042_Conf :** | | |
| ***Name*** | FrNmSourceNodeIdentifierEnabled<br>{FRNM_SOURCE_NODE_INDENTIFIER_ENABLED} | | |
| ***Description*** | Pre-processor switch for enabling SourceNodeIdentifier support. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module<br>dependency: FrNmSourceNodeIdentifierEnabled should be identically TRUE or FALSE to FrNmNodeDetectionEnabled | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0047_Conf :** | | |
| ***Name*** | FrNmStateChangeIndicationEnabled<br>{FRNM_STATE_CHANGE_INDICATION_ENABLED} | | |
| ***Description*** | Pre-processor switch for enabling state change indication. | | |
| ***Multiplicity*** | 1 | | |

| | | | |
|---|---|---|---|
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0039_Conf :** | | |
| ***Name*** | FrNmUserDataEnabled {FRNM_USER_DATA_ENABLED} | | |
| ***Description*** | Pre-processor switch for enabling user data support. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0073_Conf :** | | |
| ***Name*** | FrNmVotingNextToLastRepetitionCycleDisable {FRNM_VOTING_NEXT_TO_LAST_REPETITION_CYCLE_DISABLE} | | |
| ***Description*** | Pre-processor switch for disabling vote changes in the last two repetition cycles before the Ready Sleep Counter expires. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0069_Conf :** | | |
| ***Name*** | FrNmPnEiraRxNSduRef {FRNM_PN_EIRA_RX_NSDU_REF} | | |
| ***Description*** | Reference to a Pdu in the COM-Stack. Only one SduRef is required for FrNm because the EIRA is the aggregation over all FlexRay Channels. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | Reference to [ Pdu ] | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module<br>dependency: only available if FrNmPnEnabled == true for at least one FrNm Channel. | | |

| ***Included Containers*** | | |
|---|---|---|
| ***Container Name*** | ***Multiplicity*** | ***Scope / Dependency*** |
| FrNmPnInfo | 1 | PN information configuration |

### 10.4.4 FrNmPnInfo

| | |
|---|---|
| ***SWS Item*** | **FrNm0061_Conf :** |
| ***Container Name*** | FrNmPnInfo |
| ***Description*** | PN information configuration |

| Configuration Parameters |
|---|

| SWS Item | FrNm0063_Conf : | | |
|---|---|---|---|
| Name | FrNmPnInfoLength {FRNM_PN_INFO_LENGTH} | | |
| Description | Specifies the length of the PN request information in the NM message. | | |
| Multiplicity | 1 | | |
| Type | EcucIntegerParamDef | | |
| Range | 0 .. 7 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | Post-build time | -- | |
| Scope / Dependency | scope: Module<br>dependency: only available if FrNmPnEnabled == true for at least one FrNm Channel. | | |

| SWS Item | FrNm0062_Conf : | | |
|---|---|---|---|
| Name | FrNmPnInfoOffset {FRNM_PN_INFO_OFFSET} | | |
| Description | Specifies the offset of the PN request information in the NM message. | | |
| Multiplicity | 1 | | |
| Type | EcucIntegerParamDef | | |
| Range | 1 .. 7 | | |
| Default value | 0 | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | Post-build time | -- | |
| Scope / Dependency | scope: Module<br>dependency: only available if FrNmPnEnabled == true for at least one FrNm Channel. | | |

| Included Containers | | |
|---|---|---|
| Container Name | Multiplicity | Scope / Dependency |
| FrNmPnFilterMaskByte | 0..7 | Filter mask byte configuration |

### 10.4.5 FrNmPnFilterMaskByte

| SWS Item | FrNm0064_Conf : |
|---|---|
| Container Name | FrNmPnFilterMaskByte |
| Description | Filter mask byte configuration |
| Configuration Parameters | |

| SWS Item | FrNm0065_Conf : | | |
|---|---|---|---|
| Name | FrNmPnFilterMaskByteIndex {FRNM_PN_FILTER_MASK_BYTE_INDEX} | | |
| Description | Index of the filter mask byte. Specifies the position within the filter mask byte array. | | |
| Multiplicity | 1 | | |
| Type | EcucIntegerParamDef | | |
| Range | 0 .. 6 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME, VARIANT-POST- |

| | | | |
|---|---|---|---|
| | | | BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module<br>dependency: only available if FrNmPnEnabled == true for at least one FrNm Channel.FrNmPnFilterMaskByteIndex < FrNmPnFilterMaskLength | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0066_Conf :** | | |
| ***Name*** | FrNmPnFilterMaskByteValue {FRNM_PN_FILTER_MASK_BYTE_VALUE} | | |
| ***Description*** | Parameter to configure the filter mask byte. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucIntegerParamDef | | |
| ***Range*** | 0 .. 255 | | |
| ***Default value*** | 0 | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module<br>dependency: only available if FrNmPnEnabled == true for at least one FrNm Channel. | | |

| |
|---|
| ***No Included Containers*** |

### 10.4.6 FrNmGlobalProperties

| | |
|---|---|
| ***SWS Item*** | **FrNm0003_Conf :** |
| ***Container Name*** | FrNmGlobalProperties{FrNm_GlobalProperties} |
| ***Description*** | This container contains module properties related to the FlexRay NM functionality. |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0036_Conf :** | | |
| ***Name*** | FrNmDevErrorDetect {FRNM_DEV_ERROR_DETECT} | | |
| ***Description*** | Pre-processor switch for enabling development error detection | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

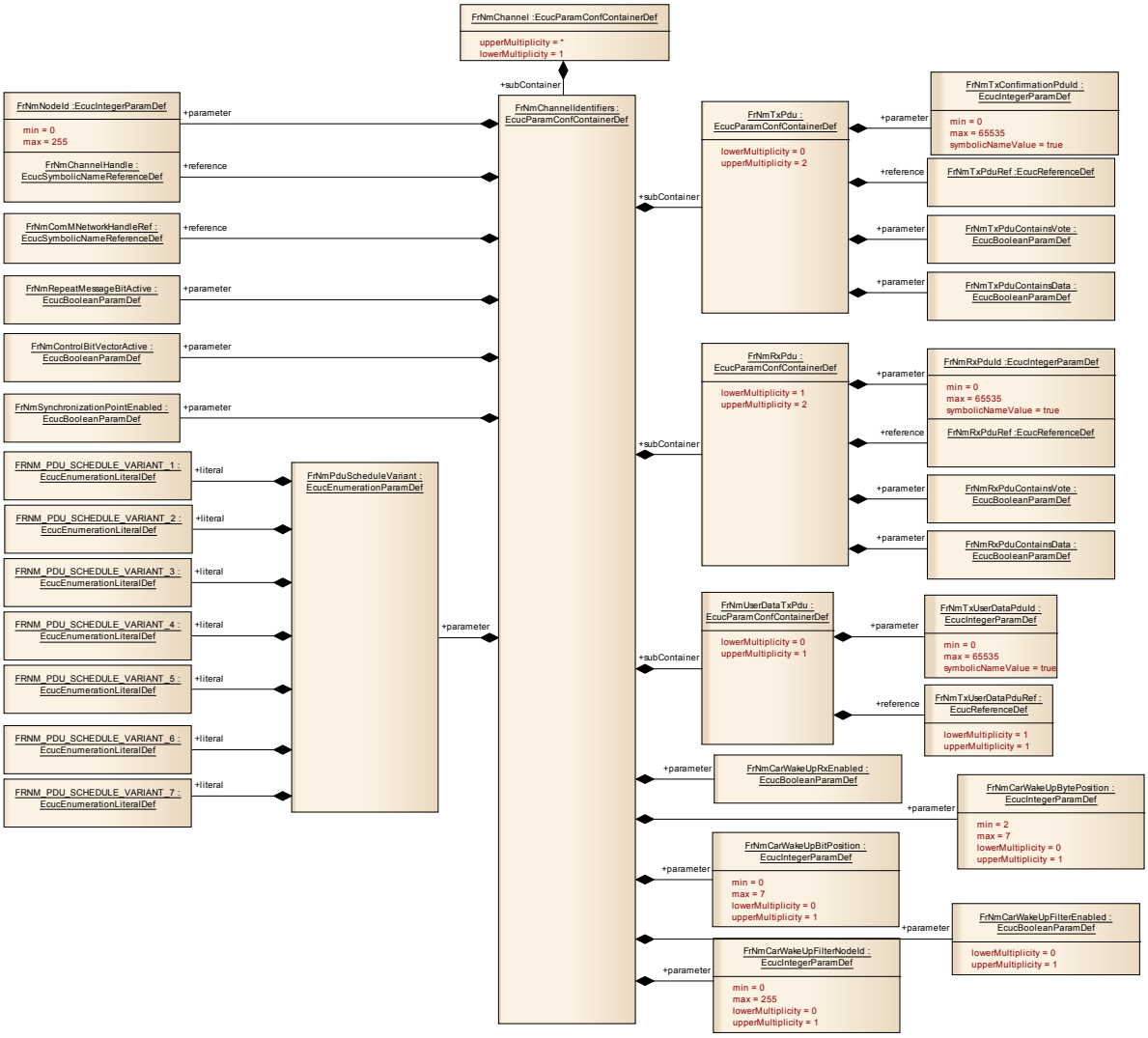
| | |
|---|---|
| ***SWS Item*** | **FrNm0038_Conf :** |
| ***Name*** | FrNmMainAcrossFrCycle {FRNM_MAIN_ACROSS_FR_CYCLE} |
| ***Description*** | Parameter describing if the execution of FrNm_Main function crosses the FlexRay cycle boundary or not. |
| ***Multiplicity*** | 1 |
| ***Type*** | EcucBooleanParamDef |
| ***Default value*** | -- |

| | | | |
|---|---|---|---|
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0037_Conf :** | | |
| ***Name*** | FrNmVersionInfoApi {FRNM_VERSION_INFO_API} | | |
| ***Description*** | Pre-processor switch for enabling version info API support. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| |
|---|
| ***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`**.⌋(BSW00345)

**[FRNM036]** ⌈The following runtime configurable parameters shall be configurable for each channel separately.⌋(BSW149)

### 10.5.1 FrNmChannelConfig

| *SWS Item* | **FrNm0002_Conf :** |
|---|---|
| ***Container Name*** | FrNmChannelConfig{FrNm_ChannelConfigType} [Multi Config Container] |
| ***Description*** | This container contains the configuration parameters for all FlexRay NM channels. |
| ***Configuration Parameters*** | |

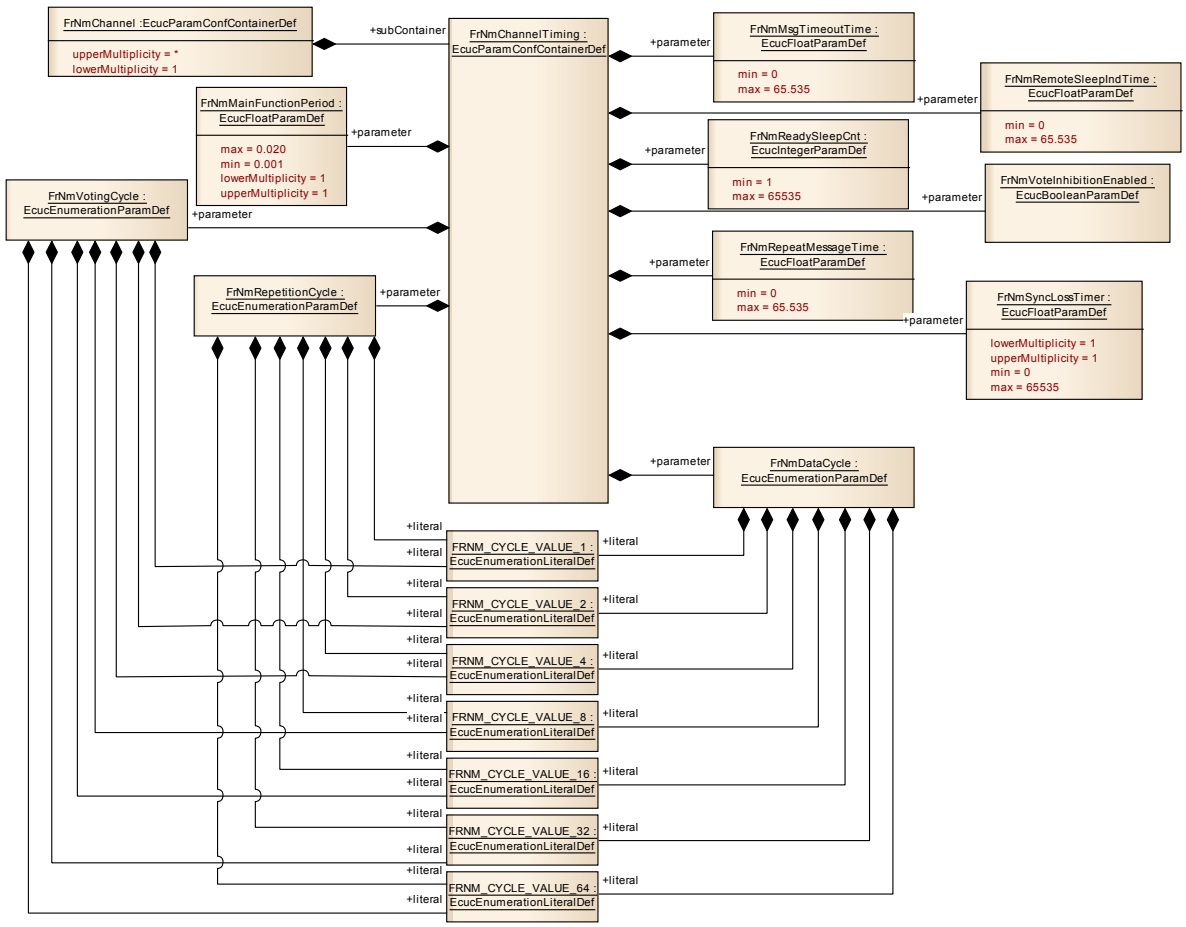
| ***Included Containers*** |
|---|

| Container Name | Multiplicity | Scope / Dependency |
|---|---|---|
| FrNmChannel | 1..* | This container contains the configuration parameters for a FlexRay NM Channel. |

### 10.5.2 FrNmChannel

| SWS Item | FrNm0006_Conf : |
|---|---|
| Container Name | FrNmChannel |
| Description | This container contains the configuration parameters for a FlexRay NM Channel. |
| 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 FrNmChannelTiming

| | |
|---|---|
| ***SWS Item*** | **FrNm0008_Conf :** |
| ***Container Name*** | FrNmChannelTiming{Channel Timing} |
| ***Description*** | This container contains instance-specific timing related to the respective FlexRay Channel. |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0031_Conf :** | | |
| ***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*** | EcucEnumerationParamDef | | |
| ***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*** | scope: instance | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0035_Conf :** | | |
| ***Name*** | FrNmMainFunctionPeriod {FRNM_MAIN_FUNCTION_PERIOD} | | |
| ***Description*** | This parameter defines the processing cycle of the main function of FrNm module in seconds. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucFloatParamDef | | |
| ***Range*** | `0.001 .. 0.02` | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0028_Conf :** | | |
| ***Name*** | FrNmMsgTimeoutTime {FRNM_MSG_TIMEOUT_TIME} | | |
| ***Description*** | Timeout of a NM-message. It determines in seconds how long the NM shall wait with notification of transmission failure while communication errors occur on the bus. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucFloatParamDef | | |
| ***Range*** | `0 .. 65.535` | | |
| ***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*** | scope: instance | | |

| | |
|---|---|
| ***SWS Item*** | **FrNm0051_Conf :** |

| | | | |
|---|---|---|---|
| ***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*** | EcucIntegerParamDef | | |
| ***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*** | scope: instance<br>dependency: Condition: FrNmReadySleepCnt ≥ 1 | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0029_Conf :** | | |
| ***Name*** | FrNmRemoteSleepIndTime {FRNM_REMOTE_SLEEP_IND_TIME} | | |
| ***Description*** | Timeout for Remote Sleep Indication. It defines the time in seconds 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*** | EcucFloatParamDef | | |
| ***Range*** | `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*** | scope: instance<br>dependency: Condition: FrNmRemoteSleepIndTime ≥ FrNmRepetitionCycle or FrNmRemoteSleepIndTime = 0 | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0030_Conf :** | | |
| ***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*** | EcucFloatParamDef | | |
| ***Range*** | `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*** | scope: instance | | |

| | |
|---|---|
| ***SWS Item*** | **FrNm0033_Conf :** |
| ***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*** | EcucEnumerationParamDef |

| | | | |
|---|---|---|---|
| ***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*** | scope: instance<br>dependency: Condition: FrNmRepetitionCycle = n * FrNmVotingCycle; n = [1,2,4,8,16,32,64] | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0079_Conf :** | | |
| ***Name*** | FrNmSyncLossTimer | | |
| ***Description*** | Initial value for the SyncLossTimer in seconds. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucFloatParamDef | | |
| ***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*** | scope: instance<br>dependency: This parameter shall be used only if FRNM_CYCLE_COUNTER_EMULATION is set as TRUE | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0053_Conf :** | | |
| ***Name*** | FrNmVoteInhibitionEnabled | | |
| ***Description*** | Pre-processor switch for enabling the inhibition of vote changes from the next-to-last repetition cycle to the last repetition cycle before the Ready Sleep Counter expires. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0032_Conf :** | | |
| ***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*** | EcucEnumerationParamDef | | |
| ***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*** | scope: instance | | |

| ***No Included Containers*** |
|---|

### 10.5.4 FrNmChannelIdentifiers

| | |
|---|---|
| ***SWS Item*** | **FrNm0007_Conf :** |
| ***Container Name*** | FrNmChannelIdentifiers{Channel Identifiers} |
| ***Description*** | This container contains instance specific identifiers related to the respective FlexRay Channel. |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0076_Conf :** | | |
| ***Name*** | FrNmCarWakeUpBitPosition {FRNM_CAR_WAKE_UP_BIT_POSITION} | | |
| ***Description*** | Specifies the Bit position of the CWU within the NM-Message. | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | EcucIntegerParamDef | | |
| ***Range*** | 0 .. 7 | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module<br>dependency: only available if FrNmCarWakeUpRxEnabled == TRUE | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0075_Conf :** | | |
| ***Name*** | FrNmCarWakeUpBytePosition {FRNM_CAR_WAKE_UP_BYTE_POSITION} | | |
| ***Description*** | Specifies the Byte position of the CWU within the NM-Message. | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | EcucIntegerParamDef | | |
| ***Range*** | 2 .. 7 | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module<br>dependency: only available if FrNmCarWakeUpRxEnabled == TRUE | | |

| | |
|---|---|
| ***SWS Item*** | **FrNm0077_Conf :** |
| ***Name*** | FrNmCarWakeUpFilterEnabled {FRNM_CAR_WAKE_UP_FILTER_ENABLED} |
| ***Description*** | If CWU filtering is supported, only the CWU bit within the NM message with source node identifier FrNmCarWakeUpFilterNodeId is considered as CWU request. FALSE - CWU Filtering is not supported TRUE - CWU Filtering is supported |
| ***Multiplicity*** | 0..1 |
| ***Type*** | EcucBooleanParamDef |

| | | | |
|---|---|---|---|
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module<br>dependency: only available if FrNmCarWakeUpRxEnabled == TRUE | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0078_Conf :** | | |
| ***Name*** | FrNmCarWakeUpFilterNodeId {FRNM_CAR_WAKE_UP_FILTER_NODE_ID} | | |
| ***Description*** | Source node identifier for CWU filtering. If CWU filtering is supported, only the CWU bit within the NM message with source node identifier FrNmCarWakeUpFilterNodeId is considered as CWU request. | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | EcucIntegerParamDef | | |
| ***Range*** | 0 .. 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*** | scope: Module<br>dependency: only available if FrNmCarWakeUpRxEnabled == TRUE | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0074_Conf :** | | |
| ***Name*** | FrNmCarWakeUpRxEnabled {FRNM_CAR_WAKE_UP_RX_ENABLED} | | |
| ***Description*** | Enables or disables support of CarWakeUp bit evaluation in received NM messages. FALSE - CarWakeUp not supported TRUE - CarWakeUp supported | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0020_Conf :** | | |
| ***Name*** | FrNmControlBitVectorActive {FRNM_CONTROL_BIT_VECTOR_ACTIVE} | | |
| ***Description*** | This parameter is used to activate or deactivate the control bit vector support for a Fr Nm Channel. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: instance<br>dependency: This parameter may only be set to TRUE if FrNmControlBitVectorEnabled is set to True | | |

| | |
|---|---|
| ***SWS Item*** | **FrNm0017_Conf :** |
| ***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*** | EcucIntegerParamDef | | |
| ***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*** | scope: instance | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0022_Conf :** | | |
| ***Name*** | FrNmPduScheduleVariant {FRNM_PDU_SCHEDULE_VARIANT} | | |
| ***Description*** | This parameter defines the PDU scheduling variant that should be used for this channel. Option 1 NM-Vote and NM-Data in static segment (one PDU) Option 2 NM-Vote and NM-Data in dynamic segment (one PDU) Option 3 NM-Vote and NM-Data in static segment (separate PDU) Option 4 NM-Vote in static segment and NM-Data in dynamic segment Option 5 NM-Vote in dynamic segment and NM-Data in static segment Option 6 NM-Vote and NM-Data in dynamic segment (separate PDU) Option 7 Combined NM-Vote and CBV in static segment and NM-Data in dynamic segment | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucEnumerationParamDef | | |
| ***Range*** | FRNM_PDU_SCHEDULE_VARIANT_1 | | NM-Vote and NM-Data in static segment (one PDU) |
| | FRNM_PDU_SCHEDULE_VARIANT_2 | | NM-Vote and NM-Data in dynamic segment (one PDU) |
| | FRNM_PDU_SCHEDULE_VARIANT_3 | | NM-Vote and NM-Data in static segment (separate PDU) |
| | FRNM_PDU_SCHEDULE_VARIANT_4 | | NM-Vote in static segment and NM-Data in dynamic segment |
| | FRNM_PDU_SCHEDULE_VARIANT_5 | | NM-Vote in dynamic segment and NM-Data in static segment |
| | FRNM_PDU_SCHEDULE_VARIANT_6 | | NM-Vote and NM-Data in dynamic segment (separate PDU) |
| | FRNM_PDU_SCHEDULE_VARIANT_7 | | Combined NM-Vote and CBV in static segment and NM-Data in dynamic segment |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Instance | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0072_Conf :** | | |
| ***Name*** | FrNmPnEnabled {FRNM_PN_ENABLED} | | |
| ***Description*** | Enables or disables support of partial networking. false: Partial networking Range not supported true: Partial networking supported | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | false | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |

| | | | |
|---|---|---|---|
| | ***Link time*** | X | VARIANT-LINK-TIME |
| | ***Post-build time*** | X | VARIANT-POST-BUILD |
| ***Scope / Dependency*** | scope: Module | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0071_Conf :** | | |
| ***Name*** | FrNmPnEraCalcEnabled {FRNM_PN_ERA_CALC_ENABLED} | | |
| ***Description*** | Specifies if FrNm calculates the PN request information for external requests. (ERA) false: PN request are not calculated true: PN request are calculated | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | false | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module<br>dependency: only available if FrNmPnEnabled == true | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0019_Conf :** | | |
| ***Name*** | FrNmRepeatMessageBitActive {FRNM_REPEAT_MESSAGE_BIT_ACTIVE} | | |
| ***Description*** | This parameter is used to activate or deactivate the repeat message bit support for a Fr Nm Channel. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: instance<br>dependency: This parameter may only be set to TRUE if FrNmRepeatMessageBitEnabled is set to True | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0021_Conf :** | | |
| ***Name*** | FrNmSynchronizationPointEnabled {FRNM_SYNCHRONIZATIONPOINT_ENABLED} | | |
| ***Description*** | This parameter defines if this channel shall provide the synchronization point indication to the NM Interface. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: instance | | |

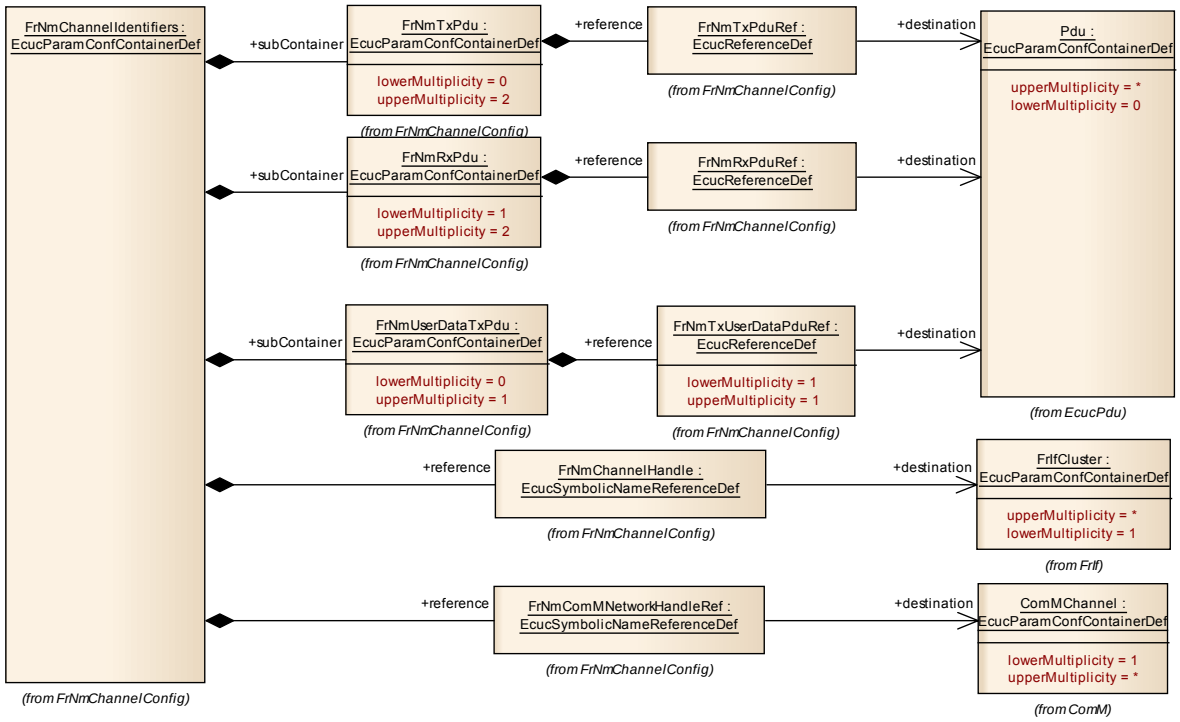
| | |
|---|---|
| ***SWS Item*** | **FrNm0013_Conf :** |
| ***Name*** | FrNmChannelHandle {FRNM_CHANNEL_HANDLE} |
| ***Description*** | Channel identifier configured for the respective instance of the NM. The FrNmChannelHandle shall be encoded in the FrNmRxPduId parameter which is passed to FrNm_RxIndication() 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*** | scope: ECU | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0014_Conf :** | | |
| ***Name*** | FrNmComMNetworkHandleRef {FRNM_CHANNEL_ID} | | |
| ***Description*** | This reference points to the unique channel defined by the ComMChannel and provides access to the unique channel index value in ComMChannelId. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | Reference to [ ComMChannel ] | | |
| ***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<br>dependency: It must be unique for each NM instance within one ECU. | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0070_Conf :** | | |
| ***Name*** | FrNmPnEraRxNSduRef {FRNM_PN_ERA_RX_NSDU_REF} | | |
| ***Description*** | Reference to a Pdu in the COM-Stack. Only one SduRef is required for FrNm because the EIRA is the aggregation over all FlexRay Channels. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | Reference to [ Pdu ] | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | VARIANT-PRE-COMPILE |
| | ***Link time*** | X | VARIANT-LINK-TIME, VARIANT-POST-BUILD |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: Module<br>dependency: only available if FrNmPnEraCalcEnabled == true | | |

| ***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. |
| FrNmUserDataTxPdu | 0..1 | This optional container is used to configure the UserNm PDU. This container is only available if FrNmComUserDataSupport is enabled. |




### 10.5.5 FrNmRxPdu

| | |
|---|---|
| ***SWS Item*** | **FrNm0010_Conf :** |
| ***Container Name*** | FrNmRxPdu |
| ***Description*** | This container describes the FlexRay NM RX PDU:s. |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0027_Conf :** | | |
| ***Name*** | FrNmRxPduContainsData {FRNM_RX_PDU_CONTAINS_DATA} | | |
| ***Description*** | This parameted defines if the PDU contains NM Data. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***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*** | scope: instance | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0026_Conf :** | | |
| ***Name*** | FrNmRxPduContainsVote {FRNM_RX_PDU_CONTAINS_VOTE} | | |
| ***Description*** | This parameted defines if the PDU contains NM Vote information. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***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*** | scope: instance | | |

| | |
|---|---|
| ***SWS Item*** | **FrNm0025_Conf :** |

| | | | |
|---|---|---|---|
| ***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*** | EcucIntegerParamDef (Symbolic Name generated for this parameter) | | |
| ***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*** | scope: instance | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0012_Conf :** | | |
| ***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*** | scope: ECU | | |

| |
|---|
| ***No Included Containers*** |

### 10.5.6 FrNmTxPdu

| | |
|---|---|
| ***SWS Item*** | **FrNm0009_Conf :** |
| ***Container Name*** | FrNmTxPdu |
| ***Description*** | This container describes the FlexRay NM TX PDU:s. |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0018_Conf :** | | |
| ***Name*** | FrNmTxConfirmationPduId {FRNM_NM_TX_PDU_ID} | | |
| ***Description*** | Handle Id to be used by the Lower Layer to confirm the transmission of the FrNmTxPdu to the LowerLayer. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucIntegerParamDef (Symbolic Name generated for this parameter) | | |
| ***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*** | scope: instance | | |

| | |
|---|---|
| ***SWS Item*** | **FrNm0024_Conf :** |
| ***Name*** | FrNmTxPduContainsData {FRNM_TX_PDU_CONTAINS_DATA} |

| | | | |
|---|---|---|---|
| ***Description*** | This parameted defines if the PDU contains NM Data. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***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*** | scope: instance | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0023_Conf :** | | |
| ***Name*** | FrNmTxPduContainsVote {FRNM_TX_PDU_CONTAINS_VOTE} | | |
| ***Description*** | This parameted defines if the PDU contains NM Vote information. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***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*** | scope: instance | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0011_Conf :** | | |
| ***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*** | scope: ECU | | |

| |
|---|
| ***No Included Containers*** |

### 10.5.7 FrNmUserDataTxPdu

| | |
|---|---|
| ***SWS Item*** | **FrNm0055_Conf :** |
| ***Container Name*** | FrNmUserDataTxPdu |
| ***Description*** | This optional container is used to configure the UserNm PDU. This container is only available if FrNmComUserDataSupport is enabled. |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0056_Conf :** | | |
| ***Name*** | FrNmTxUserDataPduId {FRNM_TX_USER_DATA_PDU_ID} | | |
| ***Description*** | This parameter defines the Handle ID of the NM User Data I-PDU. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucIntegerParamDef (Symbolic Name generated for this parameter) | | |
| ***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*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **FrNm0057_Conf :** | | |
| ***Name*** | FrNmTxUserDataPduRef | | |
| ***Description*** | Reference to the NM User Data I-PDU in the global PDU collection. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | Reference to [ Pdu ] | | |
| ***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*** |

## 10.6 Published parameters

**[FRNM400]** ⌈The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1].⌋( )

Additional module-specific published parameters are listed below if applicable.

## 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 ([5]) 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).

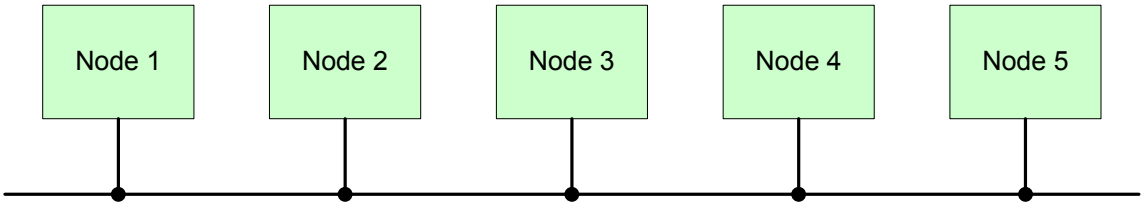



Figure 10-1 Example of five Node Network

The FlexRay Schedule allots 5 slots in the static segment and 6 (mini)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.

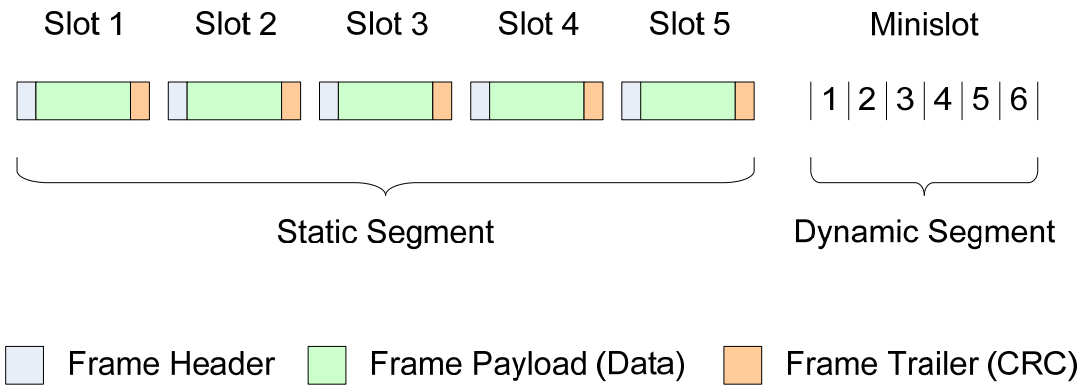



Figure 10-2 Example of Bus Schedule

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 119).

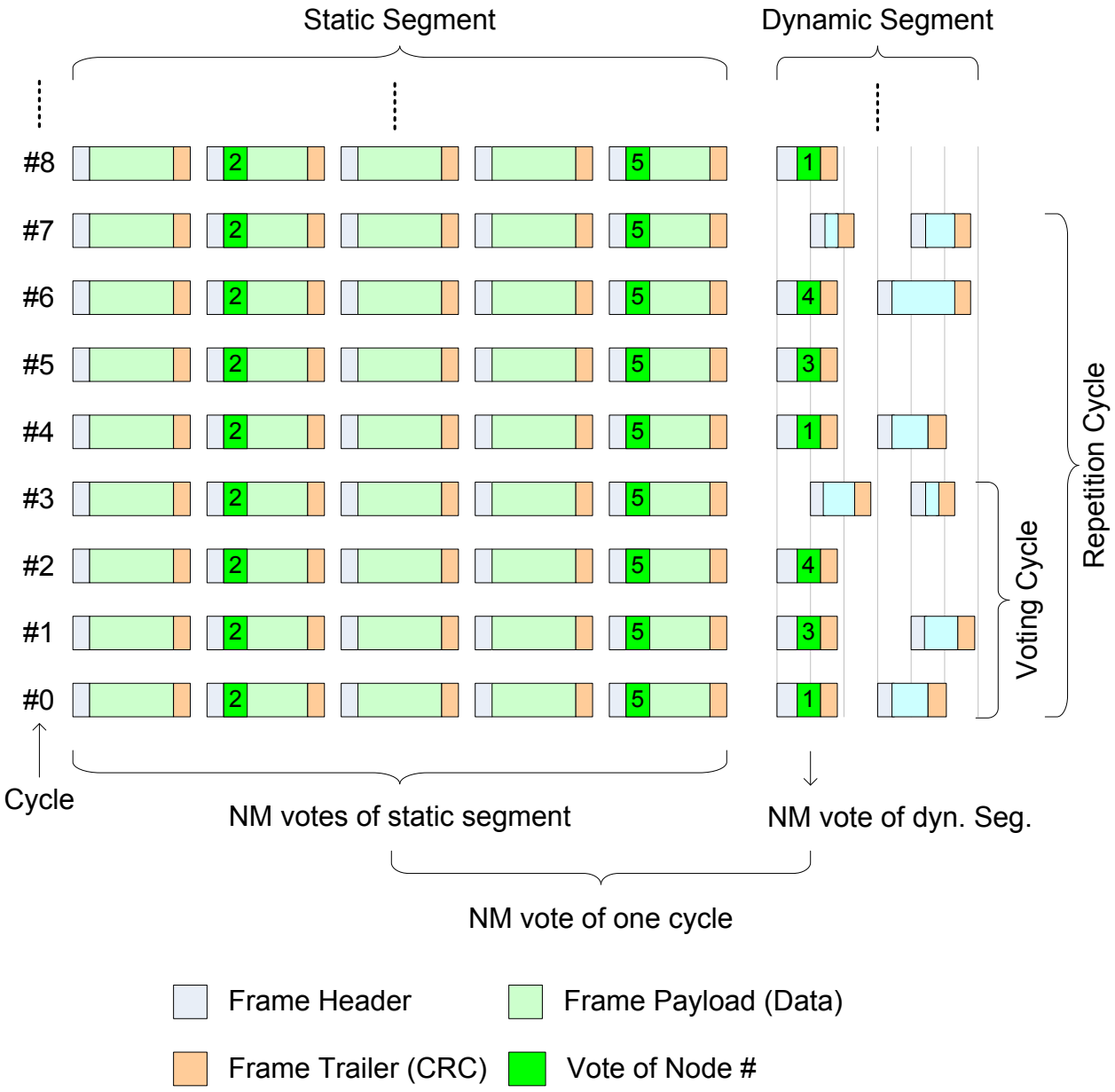



Figure 10-3 Example of NM-Vote in dynamic and static segment

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 118), 5 slot in the static segment and 6 slots in the dynamic segment (Figure 10-2 on page 118).

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 3.x

## 11.1 Deleted SWS Items

| *SWS Item* | *Rationale* |
|---|---|
| FRNM370 | |
| FRNM332 | If FrNmDualChannelPduEnable is set to true, then FrNm module would not generate Nm_RemoteSleepIndication as long as any node on either of the FlexRay channels is voting to keep the cluster awake. |
| FRNM377 | Bus Sleep Mode should be notified on this transition. (Already in FRNM134) |
| FRNM339 | If the configuration parameter FRNM_CYCLE_COUNTER_EMULATION is set to TRUE, then on reception of FrNm_StartupError, FlexRay NM shall remain in the Ready Sleep State until a local timer called FrNm_SyncLossTimer expires. |
| FRNM341 | Split into FRNM383..386<br>If FlexRay NM is in Repeat Message state and FlexRay NM receives the indication FrNm_StartupError or when global time could not be retrieved, then FlexRay NM transits to Synchronize state. The transition to Synchronize State shall only be executed if FRNM_CYCLE_COUNTER_EMULATION is set to FALSE. |

## 11.2 Replaced SWS Items

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

## 11.3 Changed SWS Items

| *SWS Item* | *Rationale* |
|---|---|
| FRNM065 | |
| | |

## 11.4 Added SWS Items

| *SWS Item* | *Rationale* |
|---|---|
| FRNM364 | |
| FRNM365 | |
| FRNM366 | |
| FRNM367 | |
| FRNM368 | |
| FRNM369 | |
| FRNM370 | |
| FRNM371 | |
| FRNM372 | |
| FRNM373 | |
| FRNM374 | |
| FRNM375 | The FrNm module shall access the FlexRay bus communication cycle via |

| | |
|---|---|
| | the API FrIf_getGlobalTime. |
| FRNM376 | |
| FRNM377 | |
| FRNM378 | |
| FRNM379 | |
| FRNM380 | |
| FRNM381 | |
| FRNM382 | |
| FRNM383 | |
| FRNM384 | |
| FRNM385 | |
| FRNM386 | |
| FRNM400 | Rework of Published Information |

# 12 Not applicable requirements

**[FRNM401]** ⌈These requirements are not applicable to this specification.⌋ (BSW00375, BSW168, BSW00423, BSW00426, BSW00427, BSW00431, BSW00433, BSW00434, BSW00336, BSW00417, BSW161, BSW162, BSW005, BSW164, BSW00325, BSW00326, BSW00347, BSW00314, BSW009, BSW00401, BSW172, BSW010, BSW00333, BSW00341, BSW00334, BSW044, BSW136, BSW140, BSW145, BSW146, BSW148, BSW139, BSW02510)