Document Title	**Specification of LIN Network Management**
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	297
Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History

Date	Release	Changed by	Change Description
2018-10-31	4.4.0	AUTOSAR Release Management	• Marked the specification as obsolete
2017-12-08	4.3.1	AUTOSAR Release Management	• Renamed "default error" to "developement error". • Changed DET error name from "LINNM_E_NO_INIT" to "LINNM_E_UNINIT". • The configuration parameters LinNmNodeIdEnabled and LinNmNodeDetectionEnabled are moved from LinNmGlobalConfig container to LinNmChannelConfig. • Editorial changes.
2016-11-30	4.3.0	AUTOSAR Release Management	• Updated requirement ECUC_LinNm_00028 for LinNm_MainFunction calling period (0..INF). • Harmonize descriptions of identical API functions. • Introduced reliable TxConfirmation. • Editorial changes.

Document Change History

Date	Release	Changed by	Change Description
2015-07-31	4.2.2	AUTOSAR Release Management	• Updated the SWS requirements for DET renaming. • Updated the SWS for LinNmComUserDataSupport. • Removed SWS requirement SWS_LinNm_00040. • Removed SWS numbers LINNM170, LINNM171 and updated with SWS_LinNm_00173, SWS_LinNm_00174.
2014-10-31	4.2.1	AUTOSAR Release Management	• Added SWS_LinNm_00172 for LinNm_ConfigType, LINNM170 for LinNm_MainFunction, ECUC_LinNm_00027 for LinNmTimeoutTime and ECUC_LinNm_00028 for LinNmMainFunctionPeriod. • Updated SWS_LinNm_00029 and SWS_LinNm_00054 for LinNm initialization ConfigPtr. • Updated “Figure 7-1”, “Figure 7-2” and “9.2 LinNm_PassiveStartUp” to enter the Lin channel into sleep mode once the LinNmTimeoutTime elapsed in passive startup. • Updated the requirements for const usage in function parameters.
2014-03-31	4.1.3	AUTOSAR Release Management	• Harmonize descriptions of identical API functions • Removed SWS_LinNm_00003
2013-10-31	4.1.2	AUTOSAR Release Management	• Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Corrected Sync / Async value of NM APIs • Renamed MemMap.h to LinNm_MemMap.h

Document Change History

Date	Release	Changed by	Change Description
2011-12-22	4.0.3	AUTOSAR Administration	• Added support for NM Coordinator Synchronization • Changed Nm_ReturnType to Std_ReturnType • Updated “Module short name” to “Module Abbreviation”
2010-02-02	3.1.4	AUTOSAR Administration	• Initial release

Table of Contents

1 Introduction and Functional Overview

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

For a general understanding of the AUTOSAR Network Management functionality please refer to [8].
Moreover, the LIN stack in AUTOSAR supports Master behavior and the protocols LIN2.x and LIN1.x.

2 Acronyms and abbreviations

Acronym/abbreviation:	***Description:***
API	Application Programming Interface
BSW	Basic Software
DET	Default Error Tracer
LinNm	Abbreviation for LIN Network Management
NM	Network Management
PDU	Protocol Data Unit
SDU	Service Data Unit

3 Related documentation

3.1 Input documents

[1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[2] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral

[3] Requirements on Network Management
AUTOSAR_SRS_NetworkManagement.pdf

[4] Requirements on LIN
AUTOSAR_SRS_LIN.pdf

[5] Specification of Communication Stack Types
AUTOSAR_SWS_CommunicationStackTypes.pdf

[6] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[7] Specification of BSW Scheduler
AUTOSAR_SWS_BSW_Scheduler.pdf

[8] Specification of Generic Network Management Interface
AUTOSAR_SWS_NetworkManagementInterface.pdf

[9] Specification of Communication Manager
AUTOSAR_SWS_COMManager.pdf

[10] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

[11] Specification of Operating System
AUTOSAR_SWS_OS.pdf

[12] Specification of Default Error Tracer
AUTOSAR_SWS_DefaultErrorTracer.pdf

[13] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[14] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes.pdf

[15] Specification of Compiler Abstraction
AUTOSAR_SWS_CompilerAbstraction.pdf

[16] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[17] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList

[18] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

Not available.

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [18] (SWS BSW General), which is also valid for LIN Network Management.

Thus, the specification SWS BSW General shall be considered as additional and required specification for LIN Network Management.

4 Constraints and assumptions

4.1 Limitations

1. One instance of LinNm is associated with only one network management cluster in one network. One network management cluster can have multiple instance of LinNm in one node.
2. One instance of LinNm is associated with only one network within the same ECU.
3. LinNm is only applicable for LIN systems.

The Figure 4-1 presents an AUTOSAR Network Management stack within an example ECU belonging to two LinNm clusters.

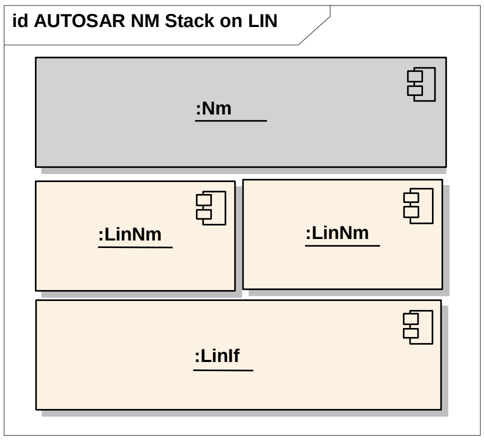

Figure 4-1

The LinNm strategy management does no need of specific coordination algorithm (like CanNm for example). Indeed, the LIN master can send to sleep and wake-up all LIN slaves connected to the bus without waiting their approvals.

4.2 Applicability to car domains

The LinNm module can be applied to any car domain under limitations provided above.

5 Dependencies to other modules

LIN Network Management (LinNm) and provides services to the Generic Network Management Interface (Nm).

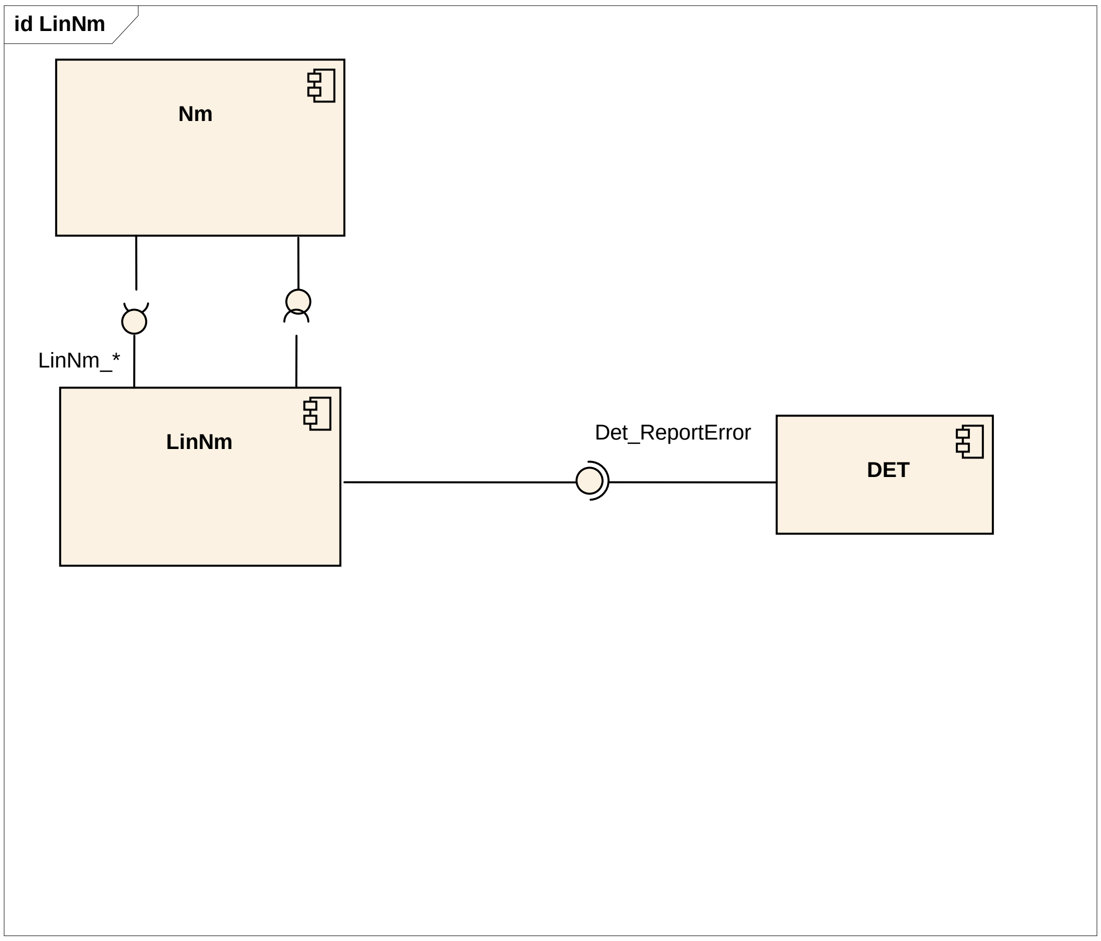

Figure 5-1 Dependencies to other modules

5.1 File Structure

5.1.1 Code File Structure

[SWS_LinNm_00000] ⌈The code file structure shall not be defined within this specification completely. ⌋ (SRS_BSW_00419, SRS_BSW_00346, SRS_BSW_00158, SRS_BSW_00308)

Note: No Post Build time configurable parameters for this Module.

6 Requirements traceability

Requirement	Description	Satisfied by
BSW00434	-	SWS_LinNm_00165
BSW136	-	SWS_LinNm_00165
BSW139	-	SWS_LinNm_00165
BSW140	-	SWS_LinNm_00165
SRS_BSW_00005	Modules of the µC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_LinNm_00165
SRS_BSW_00006	The source code of software modules above the µC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_LinNm_00165
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_LinNm_00165
SRS_BSW_00158	-	SWS_LinNm_00000
SRS_BSW_00160	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	SWS_LinNm_00165
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_LinNm_00165
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_LinNm_00165
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_LinNm_00165
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_LinNm_00165
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_LinNm_00165
SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	SWS_LinNm_00165
SRS_BSW_00305	Data types naming convention	SWS_LinNm_00165
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_LinNm_00165
SRS_BSW_00307	Global variables naming convention	SWS_LinNm_00165
SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_LinNm_00000
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_LinNm_00165
SRS_BSW_00312	Shared code shall be reentrant	SWS_LinNm_00165
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_LinNm_00165
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software	SWS_LinNm_00165

	Modules shall be enumerated according specific rules	
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_LinNm_00047, SWS_LinNm_00048
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_LinNm_00165
SRS_BSW_00326	-	SWS_LinNm_00165
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_LinNm_00165
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_LinNm_00165
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_LinNm_00165
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_LinNm_00165
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_LinNm_00165
SRS_BSW_00335	Status values naming convention	SWS_LinNm_00165
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_LinNm_00165
SRS_BSW_00339	Reporting of production relevant error status	SWS_LinNm_00165
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_LinNm_00165
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_LinNm_00000
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_LinNm_00165
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_LinNm_00165
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_LinNm_00165
SRS_BSW_00387	-	SWS_LinNm_00165
SRS_BSW_00409	All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from Dem configuration	SWS_LinNm_00165
SRS_BSW_00410	Compiler switches shall have defined values	SWS_LinNm_00165
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_LinNm_00165
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_LinNm_00165
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_LinNm_00165
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_LinNm_00165
SRS_BSW_00419	If a pre-compile time configuration parameter is implemented as "const" it should be placed into a	SWS_LinNm_00000

	separate c-file	
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_LinNm_00165
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_LinNm_00165
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_LinNm_00165
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_LinNm_00165
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_LinNm_00165
SRS_BSW_00429	Access to OS is restricted	SWS_LinNm_00165
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_LinNm_00165
SRS_Lin_01515	The LIN Interface shall provide an API to wake-up a LIN channel cluster	SWS_LinNm_00165
SRS_Lin_01523	There shall be an API call to set the LIN bus to sleep-mode.	SWS_LinNm_00165
SRS_Lin_01564	A Schedule Table Manager shall be available for LIN master nodes.	SWS_LinNm_00165
SRS_Nm_00043	NM shall not prohibit bus traffic with NM not being initialized	SWS_LinNm_00165
SRS_Nm_00046	It shall be possible to trigger the startup of all Nodes at any Point in Time.	SWS_LinNm_00165
SRS_Nm_00048	NM shall put the communication controller into sleep mode if there is no bus communication	SWS_LinNm_00165
SRS_Nm_00050	The NM shall provide the current state of NM	SWS_LinNm_00165
SRS_Nm_00051	NM shall inform application when NM state changes occur.	SWS_LinNm_00165
SRS_Nm_00052	The NM interface shall signal to the application that all other ECUs are ready to sleep.	SWS_LinNm_00165
SRS_Nm_00053	NM on a node which is or become bus unavailable shall have a deterministic Behavior	SWS_LinNm_00165
SRS_Nm_00054	There shall be a deterministic time from the point where all nodes agree to go to bus sleep to the point where bus is switched off.	SWS_LinNm_00165
SRS_Nm_00137	NM shall perform communication system error handling for errors that have impact on the NM behavior.	SWS_LinNm_00165
SRS_Nm_00142	NM shall guarantee an upper limit for the bus load generated by NM itself.	SWS_LinNm_00165
SRS_Nm_00143	The bus load caused by NM shall be predictable.	SWS_LinNm_00165
SRS_Nm_00144	NM shall support communication clusters of up to 64 ECUs	SWS_LinNm_00165
SRS_Nm_00145	On a properly configured node, NM shall tolerate a loss of a predefined number of NM messages	SWS_LinNm_00165

SRS_Nm_00146	The NM shall tolerate a time jitter of NM messages in one or more ECUs	SWS_LinNm_00165
SRS_Nm_00147	The NM algorithm shall be processor independent.	SWS_LinNm_00165
SRS_Nm_00151	The Network Management algorithm shall allow any node to integrate into an already running NM cluster	SWS_LinNm_00165
SRS_Nm_00153	The Network Management shall optionally provide a possibility to detect present nodes	SWS_LinNm_00165
SRS_Nm_00154	The Network Management API shall be independent from the communication bus	SWS_LinNm_00165
SRS_Nm_02503	The NM API shall optionally give the possibility to send user data	SWS_LinNm_00165
SRS_Nm_02504	The NM API shall optionally give the possibility to get user data	SWS_LinNm_00165
SRS_Nm_02505	The NM shall optionally set the local node identifier to the NM-message	SWS_LinNm_00165
SRS_Nm_02506	The NM API shall give the possibility to read the source node identifier of the sender	SWS_LinNm_00165
SRS_Nm_02508	Every node shall have associated with it a node identifier that is unique in the NM-cluster	SWS_LinNm_00165
SRS_Nm_02509	The NM interface shall signal to the application that at least one other ECUs is not ready to sleep anymore.	SWS_LinNm_00165
SRS_Nm_02510	For CAN NM it shall be optionally possible to immediately transmit the confirmation	SWS_LinNm_00165
SRS_Nm_02511	It shall be possible to configure the Network Management of a node in Cluster Shutdown	SWS_LinNm_00165
SRS_Nm_02512	The NM shall give the possibility to enable or disable the network management related communication configured for an active NM node	SWS_LinNm_00165

7 Functional specification

7.1 Coordination algorithm

The AUTOSAR LinNm is based on a basic state machine to go to network mode or bus sleep mode.

The main concept of the AUTOSAR LinNm state machine can be defined by the following requirement:

[SWS_LinNm_00004] ⌈If LinNm_NetworkRelease is called in the Network mode then mode shall be changed to Bus Sleep mode. ⌋ ( )

[SWS_LinNm_00161] ⌈If LinNm_PassiveStartUp is called in Bus Sleep Mode, then mode shall be changed to Network mode. ⌋ ( )

[SWS_LinNm_00162] ⌈If LinNm_NetworkRequest is called in Bus Sleep Mode, then mode shall be changed to Network mode. ⌋ ( )

The Figure 7-1 shows an overview of the state diagram for the AUTOSAR LinNm state machine from point of view of one single node in the network management cluster (one state machine per network). All services called by AUTOSAR LinNm module are in italic typeface, the bus-communication state is underlined and the events triggering the state transitions are in normal typeface.

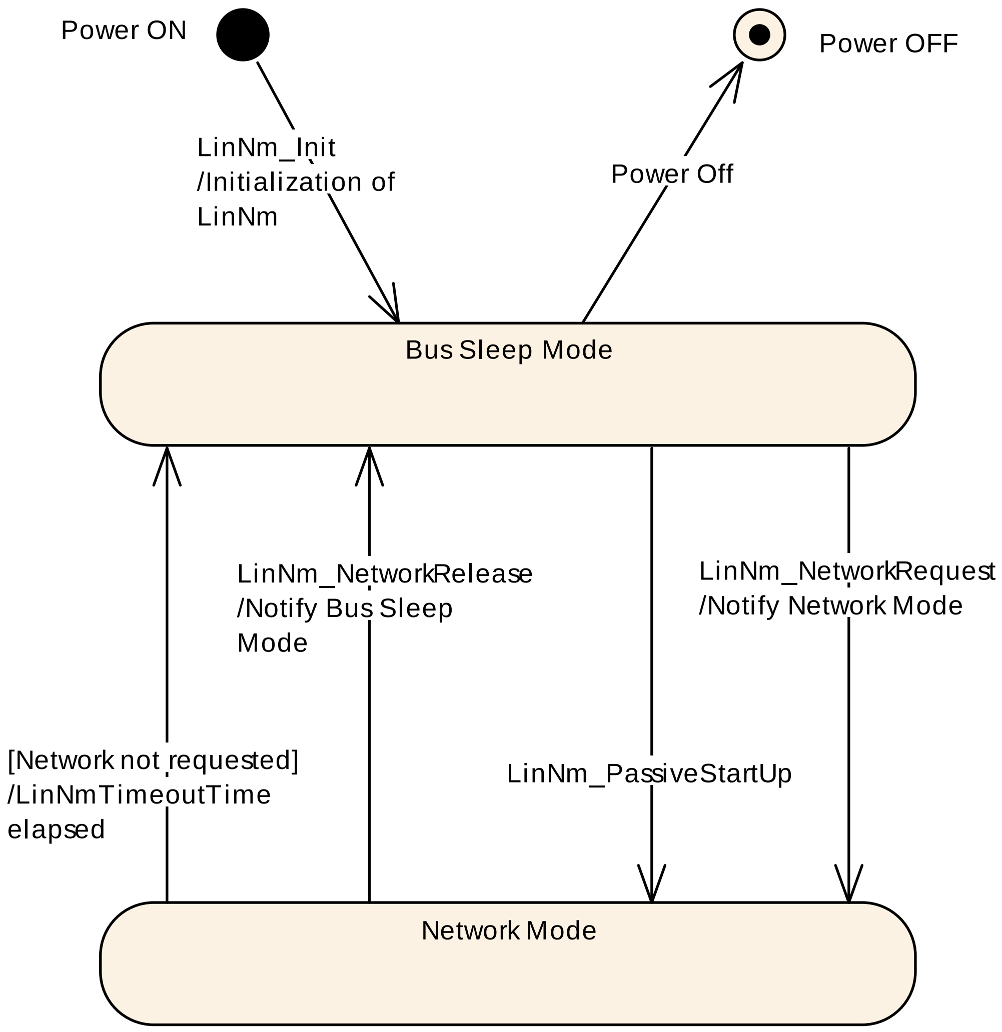
Power ON
Power OFF
LinNm_Init /Initialization of LinNm
Power Off
Bus Sleep Mode
LinNm_NetworkRelease /Notify Bus Sleep Mode
LinNm_NetworkRequest /Notify Network Mode
[Network not requested] /LinNmTimeoutTime elapsed
LinNm_PassiveStartUp
Network Mode

Figure 7-1

7.2 Operational Modes

In the following chapter operational modes of the AUTOSAR LinNm coordination algorithm are described in detail.

[SWS_LinNm_00005] ⌈The AUTOSAR LinNm shall contain two operational modes visible at the module's interface:

- `Network Mode`
- `Bus-Sleep Mode`⌋ ( )

[SWS_LinNm_00006] ⌈Changes of the AUTOSAR LinNm operational modes shall be notified to the upper layer (NM) by means of callback functions (`Nm_NetworkMode`, `Nm_BusSleepMode`). ⌋ ( )

7.2.1 Network Mode

[SWS_LinNm_00008] ⌈When the Network Mode is entered; LinNm shall notify the upper layer (NM) of the new current operational mode by calling the callback function **`Nm_NetworkMode`**. ⌋ ( )

[SWS_LinNm_00174] ⌈ If Network Mode has been entered due to a call of function **LinNm_PassiveStartUp** and if within the time configured by the parameter LinNmTimeoutTime network has not been requested; then LinNm module shall perform a transition to Bus-Sleep Mode.⌋ ()

Note: If configuration parameter LinNmTimeoutTime is set to 0 LinNm module shall immediately leave Network Mode after entering it; if no network has been requested.

7.2.2 Bus-Sleep Mode

The communication controller is switched into the sleep mode and power consumption is reduced to the adequate level in the Bus-Sleep Mode.

[SWS_LinNm_00012] ⌈When Bus-Sleep Mode is entered, except by default at initialization, the LinNm module shall notify the upper layer by calling the callback function **`Nm_BusSleepMode`**. ⌋ ( )

Note: In the Bus-Sleep Mode is assumed that the network is released, unless bus communication is explicitly requested.

[SWS_LinNm_00014] ⌈When the network is requested in Bus-Sleep Mode, the LinNm module shall enter the Network Mode. ⌋ ( )

7.3 Network states

Network states (i.e. 'NM_STATE_NORMAL_OPERATION' and 'NM_STATE_BUS_SLEEP') are two additional states of the AUTOSAR LinNm state machine that exist in parallel to the state machine. Network states denote, whether the software components need to communicate on the bus (the network state is then 'requested'); or whether the software components don't have to communicate on the bus (the bus network state is then 'released').

[SWS_LinNm_00015] ⌈The function call **LinNm_NetworkRequest** shall request the network. I.e. the LinNm module shall change network state to 'NM_STATE_NORMAL_OPERATION'. ⌋ ( )

[SWS_LinNm_00016] ⌈The function call **LinNm_NetworkRelease** shall release the network. I.e. the LinNm module shall change network state to 'NM_STATE_BUS_SLEEP'. ⌋ ( )

[SWS_LinNm_00160] ⌈If **LinNm_PassiveStartUp** is called in Bus Sleep Mode, then LinNm shall change network state to NM_STATE_NORMAL_OPERATION. ⌋ ( )

[SWS_LinNm_00103] ⌈The Modes and States shall be available for debugging. ⌋ ( )

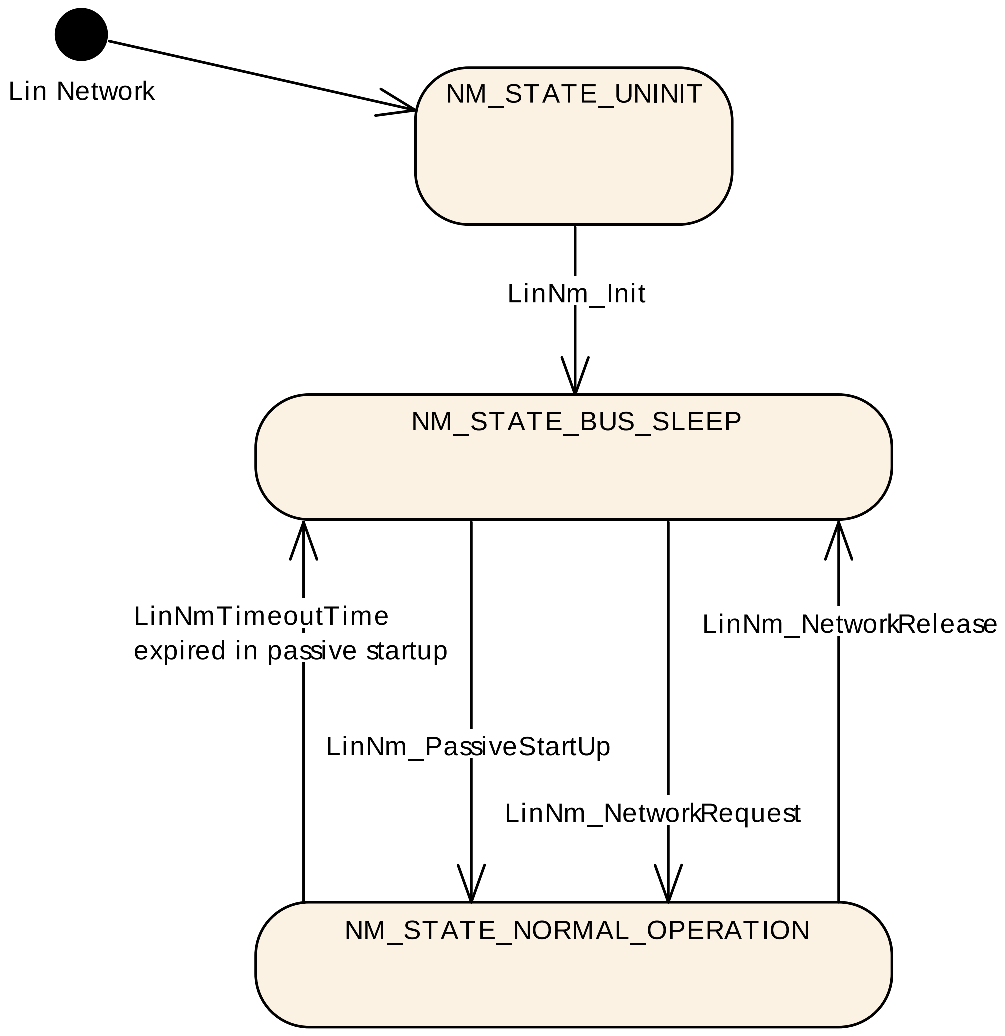

Figure 7-2

7.4 Initialization

[SWS_LinNm_00017] ⌈During initialization of the LinNm module (**LinNm_Init**) the LinNm module shall set the Network Management State to NM_STATE_UNINIT.⌋ ( )

[SWS_LinNm_00018] ⌈If the initialization of the LinNm module (**LinNm_Init**) is successful, the LinNm module shall set the Network Management State to NM_STATE_BUS_SLEEP.⌋ ( )

[SWS_LinNm_00102] ⌈No callouts shall be made from the init function, since it is not known if the other module is initialized.⌋ ( )

Note: The LinNm module should be initialized before any other network management service is called.

[SWS_LinNm_00019] ⌈If initialized, by default, the LinNm module shall set the network state to NM_STATE_BUS_SLEEP. ⌋ ( )

[SWS_LinNm_00020] ⌈If initialized, by default, the LinNm module shall enter the Bus-Sleep Mode. ⌋ ( )

[SWS_LinNm_00022] ⌈If `LinNm_PassiveStartUp` is called in the Network Mode, the LinNm module shall not execute this service and shall return E_NOT_OK. ⌋ ( )

[SWS_LinNm_00156] ⌈If `LinNm_NetworkRequest` is called in the Network Mode, the LinNM module shall not execute this service and shall return E_NOT_OK. ⌋ ( )

[SWS_LinNm_00157] ⌈If `LinNm_NetworkRelease` is called in the Bus Sleep Mode, the LinNM module shall not execute this service and shall return E_NOT_OK. ⌋ ( )

[SWS_LinNm_00025] ⌈If LinNm is not initialized the LinNm module shall reject each call of a LinNm function with the respective error code, except `LinNm_Init` and `LinNm_GetVersionInfo`. ⌋ ( )

7.5 Execution

7.5.1 Processor architecture

[SWS_LinNm_00026] ⌈The AUTOSAR LinNm state machine shall be processor independent, which means it shall not rely on any processor specific hardware support and thus shall be realizable on any processor architecture that is in the scope of AUTOSAR. ⌋ ( )

7.5.2 Timing parameters

There is no configuration parameter.

7.6 Additional features

7.6.1 State change notification

[SWS_LinNm_00061] ⌈If the configuration parameter **LINNM_STATE_CHANGE_IND_ENABLED** is defined, the LinNm module shall call the callback function **`Nm_StateChangeNotification`** each time the bus state is modified. ⌋ ( )

7.7 Error classification

7.7.1 Development Errors

This chapter shall list all Development Errors that can be detected within this software module. For each error, a value shall be defined.

[SWS_LinNm_00029] ⌈The following errors shall be detectable by the LinNm depending on its build version (development).

Type or error	***Relevance***	***Related error code***	***Error Value***
API service used without module initialization	Development	LINNM_E_UNINIT	0x01
API service called with wrong channel handle	Development	LINNM_E_INVALID_CHANNEL	0x02
Null pointer has been passed as an argument.	Development	LINNM_E_PARAM_POINTER	0x12
LinNm initialization has been failed, e.g. selected configuration set doesn't exist.	Development	LINNM_E_INIT_FAILED	0x13
API call with invalid Parameter	Development	LINNM_E_INVALID_PARAMETER	0x14

⌋ ( )

7.7.2 Runtime Errors

There are no runtime errors.

7.7.3 Transient Faults

There are no transient faults.

7.7.4 Production Errors

There are no production errors.

7.7.5 Extended Production Errors

There are no extended production errors.

7.8 Error detection

For details refer to the chapters 7.3 “Error Detection” in *SWS_BSWGeneral.*

7.9 Error notification

[SWS_LinNm_00034] ⌈If development error detection is enabled and the input arguments to LinNm API services are invalid then the LinNm module shall report respective errors to Default Error Tracer and return without any action.⌋ ( )

[SWS_LinNm_00037] ⌈If development error detection is enabled and the LinNm module is not initialized then all the LinNm API services (except LinNm_Init and LinNm_GetVersionInfo) shall report an error LINNM_E_UNINIT to Default Error Tracer and return without any action.⌋ ( )

[SWS_LinNm_00038] ⌈If development error detection is enabled and the input argument nmChannelHandle has an invalid value then the network handle services shall report an error **LINNM_E_INVALID_CHANNEL** to Default Error Tracer and return without any action.

Note: The network handle is invalid if it is different from allowed configured values.⌋ ( )

7.10 Application notes

7.10.1 Wakeup notification

Wakeup notification is defined in detail in the ECU State Manager specification.

7.10.2 Coordination of coupled networks

[SWS_LinNm_00041] ⌈Support of bus synchronization on demand shall be statically configurable with use of the **LINNM_BUS_SYNCHRONIZATION_ENABLED** switch (configuration parameter). ⌋ ( )

[SWS_LinNm_00042] ⌈If **LinNm_RequestBusSynchronization** is called in Bus-Sleep Mode, the LinNm module shall not execute the service and shall return **E_OK**. ⌋ ( )

[SWS_LinNm_00140] ⌈The parameter LINNM_SYNCHRONIZATIONPOINT_ENABLED shall be always disabled. ⌋ ( )

[SWS_LinNm_00141] ⌈LinNm shall make a callout to Nm_RemoteSleepIndication(channel) after wakeup of Network.(i.e., after entering into Normal Operation Mode). ⌋ ( )

Note: LinNm shall never make callouts to Nm_SynchronizationPoint(channel).

7.10.3 Coordinator Synchronization Support

When having more than one coordinator connected to the same bus a special bit in the Control Bit Vector (CBV), the *NmCoordinatorSleepReady* bit is used to indicate that the main coordinator requests to start shutdown sequence. The main functionality of the algorithm is described in the Nm module.

[**SWS_LinNm_00169]** ⌈ The API LinNm_SetSleepReadyBit() and the feature "Coordinated Bus Shutdown" shall only be available if LinNmCoordinatorSyncSupport is set to TRUE.⌋ ().

7.10.4 Debugging Concept

For details refer to the chapter 7.1.17 “Debugging support” in *SWS_BSWGeneral.*

8 API specification

[SWS_LinNm_00047] ⌈The LinNm module shall provide parameter value check only in "development mode".⌋ (SRS_BSW_00323)

[SWS_LinNm_00048] ⌈The LinNm module shall reject the execution of a service called with an invalid parameter and shall inform the DET. ⌋ (SRS_BSW_00323)

AUTOSAR LinNm API consists of services, which are LIN specific and can be called whenever they are required; each service apart from **LinNm_Init** refers to one NM channel only.

8.1 Imported Types

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

[SWS_LinNm_00078] ⌈

Module	***Header File***	***Imported Type***
ComStack_Types		NetworkHandleType
		PduIdType
		PduInfoType
	ComStackTypes.h	PduIdType
Nm		Nm_ModeType
		Nm_StateType
Std_Types		Std_ReturnType
		Std_VersionInfoType
	StandardTypes.h	Std_ReturnType

⌋ ()

8.2 Type Definitions

8.2.1 LinNm_ConfigType

[SWS_LinNm_00172] ⌈

Name:	`LinNm_ConfigType`	
Type:	`Structure`	
Range:	`implementation specific`	--
Description:	A pointer to an instance of this structure will be used in the initialization of LinNm module. The outline of the structure is defined in chapter 10 Configuration Specification.	
Available via:	`LinNm.h`	

⌋ ()

8.3 Function Definitions

8.3.1 LinNm_Init

[SWS_LinNm_00054] ⌈

Service name:	LinNm_Init	

Syntax:	`void LinNm_Init(`
	`    const LinNm_ConfigType* ConfigPtr`
	`)`	

Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfigPtr	Pointer to a selected configuration structure
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initialize the complete LinNm module.	
Available via:	`LinNm.h`	

⌋ ()

8.3.2 LinNm_PassiveStartUp

[SWS_LinNm_00063] ⌈

Service name:	LinNm_PassiveStartUp	

Syntax:	`Std_ReturnType LinNm_PassiveStartUp(`
	`    NetworkHandleType nmChannelHandle`
	`)`	

Service ID[hex]:	0x01	
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 E_NOT_OK: Passive startup of network management has failed
Description:	Passive startup of the AUTOSAR LIN NM.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00064] ⌈If the current state is not equal to Bus-Sleep Mode, then the function LinNm_PassiveStartUp shall have no effect except that `E_NOT_OK` is returned. ⌋ ( )

[SWS_LinNm_00065] ⌈Caveats of LinNm_PassiveStartUp: The LinNm module is initialized correctly. ⌋ ( )

8.3.3 LinNm_NetworkRequest

[SWS_LinNm_00055] ⌈

Service name:	LinNm_NetworkRequest	
Syntax:	`Std_ReturnType LinNm_NetworkRequest( NetworkHandleType nmChannelHandle )`	
Service ID[hex]:	0x02	
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 E_NOT_OK: Requesting of network has failed
Description:	Request the network, since ECU needs to communicate on the bus.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00053] ⌈Caveats of LinNm_NetworkRequest: The LinNm module is initialized correctly. ⌋ ( )

[SWS_LinNm_00158] ⌈Configuration of LinNm_NetworkRequest: This function is only available if `LINNM_PASSIVE_MODE_ENABLED` is set to FALSE. ⌋ ( )

8.3.4 LinNm_NetworkRelease

[SWS_LinNm_00056] ⌈

Service name:	LinNm_NetworkRelease	
Syntax:	`Std_ReturnType LinNm_NetworkRelease( NetworkHandleType nmChannelHandle )`	
Service ID[hex]:	0x03	
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 E_NOT_OK: Releasing of network has failed
Description:	Release the network, since ECU doesn't have to communicate on the bus.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00058] ⌈Caveats of LinNm_NetworkRelease: The LinNm module is initialized correctly. ⌋ ( )

[SWS_LinNm_00159] ⌈Configuration of LinNm_NetworkRelease: This function is only available if `LINNM_PASSIVE_MODE_ENABLED` is set to FALSE. ⌋ ( )

8.3.5 LinNm_GetVersionInfo

[SWS_LinNm_00106] ⌈

Service name:	LinNm_GetVersionInfo	
Syntax:	`void LinNm_GetVersionInfo( Std_VersionInfoType* versioninfo )`	
Service ID[hex]:	0xf1	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module
Return value:	None	
Description:	This service returns the version information of this module.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00163] ⌈If development error detection is enabled and the input argument versioninfo has null pointer then the service LinNm_GetVersionInfo() shall report an error LINNM_E_PARAM_POINTER to Default Error Tracer and return without any action.⌋ ( )

8.3.6 LinNm_RequestBusSynchronization

[SWS_LinNm_00089] ⌈

Service name:	LinNm_RequestBusSynchronization	
Syntax:	`Std_ReturnType LinNm_RequestBusSynchronization( NetworkHandleType nmChannelHandle )`	
Service ID[hex]:	0xc0	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK : No error
Description:	Empty function to be complaint with NM specifications. Request bus synchronization.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00095] ⌈Service call **`LinNm_RequestBusSynchronization`** shall provide an empty implementation. ⌋ ( )

[SWS_LinNm_00090] ⌈Caveats of LinNm_RequestBusSynchronization: The LinNm module is initialized correctly. ⌋ ( )

[SWS_LinNm_00091] ⌈Configuration of LinNm_RequestBusSynchronization: Optional (Only available if `LINNM_BUS_SYNCHRONIZATION_ENABLED` is defined) and `LINNM_PASSIVE_MODE_ENABLED` is not defined. ⌋ ( )

Rationale: This service is typically used for supporting the NM gateway extensions.

8.3.7 LinNm_CheckRemoteSleepIndication

[SWS_LinNm_00092] ⌈

Service name:	LinNm_CheckRemoteSleepIndication	
Syntax:	`Std_ReturnType LinNm_CheckRemoteSleepIndication( NetworkHandleType nmChannelHandle, boolean* nmRemoteSleepIndPtr )`	
Service ID[hex]:	0xd0	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (but not for the same NM-channel)	
Parameters (in):	nmChannelHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmRemoteSleepIndPtr	Pointer where check result of remote sleep indication shall be copied to
Return value:	Std_ReturnType	E_OK: No error
Description:	Empty function to be complaint with NM specifications.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00093] ⌈Service call **`LinNm_CheckRemoteSleepIndication`** shall provide an empty implementation. ⌋ ( )

[SWS_LinNm_00094] ⌈Caveats of LinNm_CheckRemoteSleepIndication: The LinNm module and Nm module shall be initialized correctly. ⌋ ( )

[SWS_LinNm_00096] ⌈Configuration of LinNm_CheckRemoteSleepIndication: Optional (Only available if `LINNM_REMOTE_SLEEP_INDICATION_ENABLED` is defined) ⌋ ( )

8.3.8 LinNm_SetSleepReadyBit

[SWS_LinNm_00175] ⌈

Service name:	LinNm_SetSleepReadyBit
Syntax:	`Std_ReturnType LinNm_SetSleepReadyBit( NetworkHandleType nmChannelHandle, boolean nmSleepReadyBit )`
Service ID[hex]:	0x10
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
Description:	Empty function to be compliant with NM specifications.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00176] ⌈ Configuration of LinNm_SetSleepReadyBit: This function is only available if LinNmCoordinatorSyncSupport is set to TRUE.⌋ ()

[SWS_LinNm_00177] ⌈ Service call LinNm_SetSleepReadyBit shall provide an empty implementation.⌋ ()

8.3.9 Communication control services provided by NM Interface

The following services are provided by NM Interface to allow the **Diagnostic Communication Manager** (**DCM**) to control the transmission of NM Messages.

8.3.9.1 LinNm_DisableCommunication

[SWS_LinNm_00108] ⌈

Service name:	LinNm_DisableCommunication	

Syntax:	`Std_ReturnType LinNm_DisableCommunication(`
	`    NetworkHandleType NetworkHandle`
	`)`	

Service ID[hex]:	0x04	
Sync/Async:	Asynchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error
Description:	Empty function to be complaint with NM specifications.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00109] ⌈Caveats of LinNm_DisableCommunication: The **LinNm** and the **Nm** itself are initialized correctly. ⌋ ( )

[SWS_LinNm_00110] ⌈Configuration of LinNm_DisableCommunication: This function is only available if LINNM_COM_CONTROL_ENABLED is set to TRUE. ⌋ ( )

8.3.9.2 LinNm_EnableCommunication

[SWS_LinNm_00111] ⌈

Service name:	LinNm_EnableCommunication	
Syntax:	`Std_ReturnType LinNm_EnableCommunication( NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x05	
Sync/Async:	Asynchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error
Description:	Empty function to be complaint with NM specifications.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00112] ⌈Caveats of LinNm_EnableCommunication: The **LinNm** and the **Nm** itself are initialized correctly. ⌋ ( )

[SWS_LinNm_00113] ⌈Configuration of LinNm_EnableCommunication: This function is only available if LINNM_COM_CONTROL_ENABLED is set to TRUE. ⌋ ( )

8.3.10 Extra services provided by NM Interface

The following services are provided by NM Interface for OEM specific extensions of the NM stack and are not required by any AUTOSAR module.

8.3.10.1 LinNm_SetUserData

[SWS_LinNm_00114] ⌈

Service name:	LinNm_SetUserData	
Syntax:	`Std_ReturnType LinNm_SetUserData( NetworkHandleType NetworkHandle, const uint8* nmUserDataPtr )`	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
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
Description:	Empty function to be complaint with NM specifications.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00115] ⌈Caveats of LinNm_SetUserData: The **LinNm** and the **Nm** itself are initialized correctly. ⌋ ( )

[SWS_LinNm_00116] ⌈Configuration of LinNm_SetUserData: This function is only available if LINNM_USER_DATA_ENABLED is set to TRUE and LINNM_PASSIVE_MODE_ENABLED is set to FALSE. ⌋ ( )

8.3.10.2 LinNm_GetUserData

[SWS_LinNm_00117] ⌈

Service name:	LinNm_GetUserData	
Syntax:	`Std_ReturnType LinNm_GetUserData( NetworkHandleType NetworkHandle, uint8* nmUserDataPtr )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmUserDataPtr	Pointer where user data out of the last successfully received NM message shall be copied to
Return value:	Std_ReturnType	E_OK: No error
Description:	Empty function to be complaint with NM specifications.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00118] ⌈Caveats of LinNm_GetUserData: The **LinNm** and the **Nm** itself are initialized correctly. ⌋ ( )

[SWS_LinNm_00119] ⌈Configuration of LinNm_GetUserData: This function is only available if LINNM_USER_DATA_ENABLED is set to TRUE. ⌋ ( )

8.3.10.3 LinNm_GetPduData

[SWS_LinNm_00120] ⌈

Service name:	LinNm_GetPduData	
Syntax:	`Std_ReturnType LinNm_GetPduData( NetworkHandleType NetworkHandle, uint8* nmPduData )`	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmPduData	Pointer where NM PDU shall be copied to.
Return value:	Std_ReturnType	E_OK: No error
Description:	Empty function to be complaint with NM specifications.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00121] ⌈Caveats of LinNm_GetPduData: The **LinNm** and the **Nm** itself are initialized correctly. ⌋ ( )

[SWS_LinNm_00122] ⌈Configuration of LinNm_GetPduData: This function is only available if LINNM_NODE_ID_ENABLED or LINNM_USER_DATA_ENABLED is set to TRUE. ⌋ ( )

8.3.10.4 LinNm_RepeatMessageRequest

[SWS_LinNm_00123] ⌈

Service name:	LinNm_RepeatMessageRequest	
Syntax:	`Std_ReturnType LinNm_RepeatMessageRequest( NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x09	
Sync/Async:	Asynchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error
Description:	Empty function to be complaint with NM specifications.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00124] ⌈Caveats of LinNm_RepeatMessageRequest: **LinNm** and **Nm** itself are initialized correctly. ⌋ ( )

[SWS_LinNm_00125] ⌈Configuration of LinNm_RepeatMessageRequest: This function is only available if LINNM_NODE_DETECTION_ENABLED is TRUE. ⌋ ( )

8.3.10.5 LinNm_GetNodeIdentifier

[SWS_LinNm_00126] ⌈

Service name:	LinNm_GetNodeIdentifier	
Syntax:	`Std_ReturnType LinNm_GetNodeIdentifier( NetworkHandleType NetworkHandle, uint8* nmNodeIdPtr )`	
Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmNodeIdPtr	Pointer where node identifier out of the last successfully received NM-message shall be copied to
Return value:	Std_ReturnType	E_OK: No error
Description:	Empty function to be complaint with NM specifications.	

Available via:	`LinNm.h`

⌋ ()

[SWS_LinNm_00127] ⌈Caveats of LinNm_GetNodeIdentifier: The **LinNm** and the **Nm** itself are initialized correctly. ⌋ ( )

[SWS_LinNm_00128] ⌈Configuration of LinNm_GetNodeIdentifier: This function is only available if LINNM_NODE_ID_ENABLED is set to TRUE. ⌋ ( )

8.3.10.6 LinNm_GetLocalNodeIdentifier

[SWS_LinNm_00129] ⌈

Service name:	LinNm_GetLocalNodeIdentifier	

Syntax:	`Std_ReturnType LinNm_GetLocalNodeIdentifier(`
	`    NetworkHandleType NetworkHandle,`
	`    uint8* nmNodeIdPtr`
	`)`	

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant for the same NetworkHandle, reentrant otherwise	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmNodeIdPtr	Pointer where node identifier of the local node shall be copied to
Return value:	Std_ReturnType	E_OK: No error
Description:	Empty function to be complaint with NM specifications.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00130] ⌈Caveats of LinNm_GetLocalNodeIdentifier: The **LinNm** and the **Nm** itself are initialized correctly. ⌋ ( )

[SWS_LinNm_00131] ⌈Configuration of LinNm_GetLocalNodeIdentifier: This function is only available if LINNM_NODE_ID_ENABLED is set to TRUE. ⌋ ( )

8.3.10.7 LinNm_GetState

[SWS_LinNm_00135] ⌈

Service name:	LinNm_GetState	

Syntax:	`Std_ReturnType LinNm_GetState(`
	`    NetworkHandleType nmNetworkHandle,`
	`    Nm_StateType* nmStatePtr,`
	`    Nm_ModeType* nmModePtr`
	`)`	

Service ID[hex]:	0x0e	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	nmNetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmStatePtr	Pointer where state of the network management shall be

		copied to
	nmModePtr	Pointer to the location where the mode of the network management shall be copied to
Return value:	Std_ReturnType	E_OK: No error
Description:	Returns the state of the network management.The function LinNm_GetState shall be called (e.g. LinNm_GetState function is called if channel is configured as LIN).	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00136] ⌈Caveats of LinNm_GetState: The **LinNm** and the **Nm** itself are initialized correctly. ⌋ ( )

8.3.10.8 LinNm_Transmit

[SWS_LinNm_00148] ⌈

Service name:	LinNm_Transmit	

Syntax:	`Std_ReturnType LinNm_Transmit(`
	`    PduIdType TxPduId,`
	`    const PduInfoType* PduInfoPtr`
	`)`	

Service ID[hex]:	0x49	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Transmit request has been accepted. E_NOT_OK: Transmit request has not been accepted.
Description:	Requests transmission of a PDU.	
Available via:	`LinNm.h`	

⌋ ()

[SWS_LinNm_00149] ⌈Service call LinNm_Transmit shall provide an empty implementation⌋ ( )
[SWS_LinNm_00178] ⌈ LinNm_Transmit shall always return E_NOT_OK. ⌋ ()

[SWS_LinNm_00150] ⌈Caveats of LinNm_Transmit: The **LinNm** and the **Nm** itself are initialized correctly. ⌋ ( )

[SWS_LinNm_00151] ⌈Configuration of LinNm_Transmit: This function is only available if LINNM_COM_USER_DATA_SUPPORT is set to TRUE. ⌋ ( )

8.3.10.9 LinNm_TxConfirmation

[SWS_LinNm_00153] ⌈

Service name:	LinNm_TxConfirmation

Syntax:	`void LinNm_TxConfirmation(`
	`    PduIdType TxPduId,`
	`    Std_ReturnType result`

	)	
Service ID[hex]:	0x40	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the PDU that has been transmitted.
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	
Available via:	LinNm.h	

⌋ ()

[SWS_LinNm_00154] ⌈Caveats of LinNm_TxConfirmation: The **LinNm** and the **Nm** itself are initialized correctly. ⌋ ( )

[SWS_LinNm_00179] ⌈ If development error detection is enabled:
The function LinNm_TxConfirmation shall check the parameter Result for being valid. If the check for Result fails, the function LinNm_TxConfirmation shall raise the development error LINNM_E_INVALID_PARAMETER.⌋ ()

8.4 Call-back Notifications

Not applicable

8.5 Scheduled Functions

8.5.1 LinNm_MainFunction

[SWS_LinNm_00173] ⌈

Service name:	LinNm_MainFunction

Syntax:	`void LinNm_MainFunction(`
	`    void`
	`)`

Service ID[hex]:	0x11
Description:	Main function of the LinNm which processes the algorithm described in document SWS LinNm.
Available via:	`LinNm_SchM.h`

⌋ ()

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

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

API function	***Header File***	***Description***
Nm_BusSleepMode	Nm.h	Notification that the network management has entered Bus-Sleep Mode.
Nm_NetworkMode	Nm.h	Notification that the network management has entered Network Mode.

8.6.2 Optional Interfaces

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

API function	***Header File***	***Description***
Det_ReportError	Det.h	Service to report development errors.
Nm_CoordReadyToSleepCancellation	Nm.h	Cancels an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set back to 0.

Nm_CoordReadyToSleepIndication	Nm.h	Sets an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set
Nm_RemoteSleepIndication	Nm.h	Notification that the network management has detected that all other nodes on the network are ready to enter Bus-Sleep Mode.
Nm_StateChangeNotification	Nm.h	Notification that the state of the lower layer <BusNm> has changed.

8.6.3 Configurable interfaces

Not applicable

8.6.4 Job End Notification

Not applicable

8.7 Service Interfaces

Not applicable

8.8 Parameter check

[SWS_LinNm_00069] ⌈If development error detection is enabled by `LINNM_DEV_ERROR_DETECT` (configuration parameter), then for all LinNm API services validity check of input parameters shall be made.⌋ ( )

[SWS_LinNm_00070] ⌈Parameter type checking shall be made at compile time;
if types do not fit the compilation process shall be stopped and respective compilation warnings or errors shall be returned as far as supported by the compiler.⌋ ( )

[SWS_LinNm_00071] ⌈Parameter value check (for parameters of the constant value) shall be made at configuration time; if the value is invalid, the configuration process shall be stopped and respective configuration error shall be reported.⌋ ( )

[SWS_LinNm_00072] ⌈Parameter value check (for parameters of the variable value) shall be made at execution time; if the value is invalid, execution of a service shall be rejected and the LinNm module shall report respective errors to Default Error Tracer.⌋ ( )

8.9 Version check

For details refer to the chapter 5.1.8 “Version Check” in *SWS_BSWGeneral.*

9 Sequence diagrams

9.1 LinNm_Init

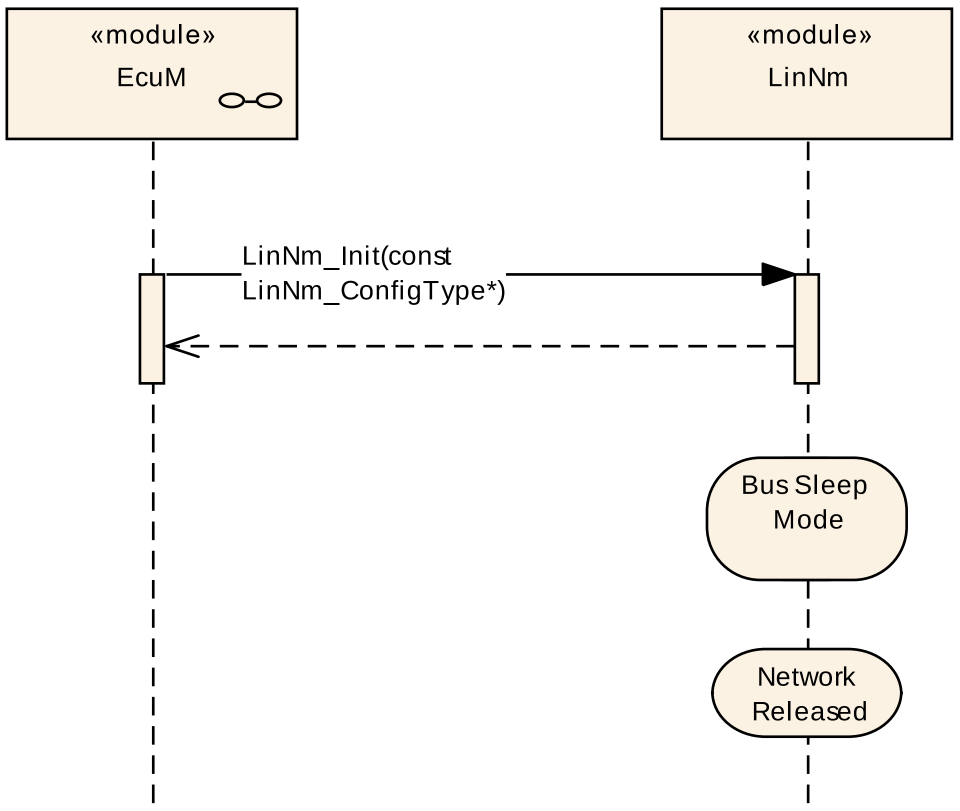

Figure 9-1 LinNm_init

9.2 LinNm_PassiveStartUp

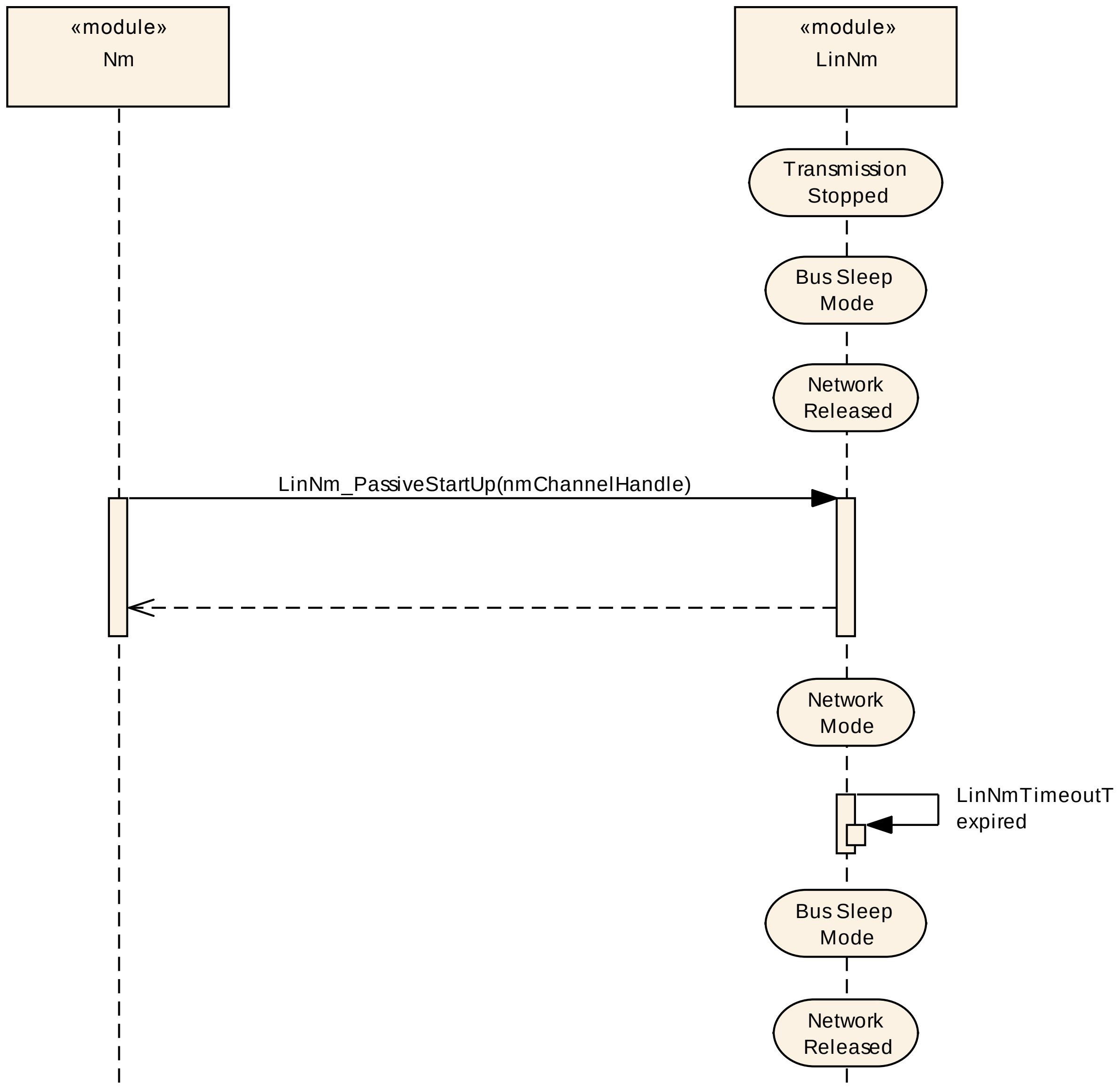

Figure 9-2 LinNm Passive Startup

9.3 LinNm_NormalOperation

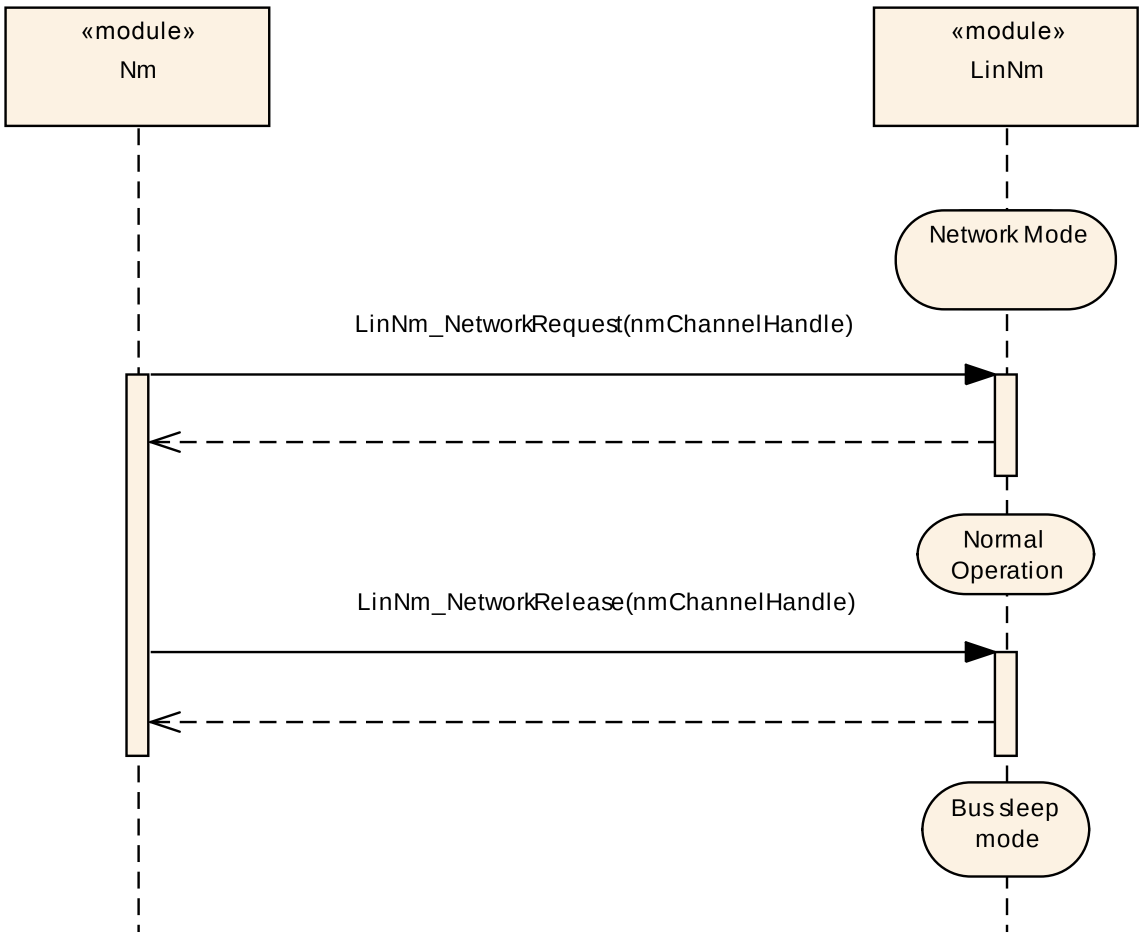
«module»
Nm
«module»
LinNm
Network Mode
LinNm_NetworkRequest(nmChannelHandle)
Normal Operation
LinNm_NetworkRelease(nmChannelHandle)
Bus sleep mode

Figure 9-3 LinNm Normal Operation

10 Configuration specification

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) you shall use for the parameter specification. We intend to leave chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module LinNm.

Chapter 10.3 specifies published information of the module LinNm.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS_BSWGeneral.*

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and 8.

The configuration parameters as defined in this chapter are used to create a data model for an AUTOSAR tool chain. The realization in the code is implementation specific.

The configuration parameters as defined in this chapter are used to create a data model for an AUTOSAR tool chain. The realization in the code is implementation specific.

The configuration parameters are divided in parameters which are used to enable features, parameters which affect all instances of the LinNm and parameters which affect the respective instances of the LinNm.

[SWS_LinNm_00074] ⌈All configuration items shall be located outside the kernel of the module. ⌋ ( )

10.3 Containers and configuration parameters

This chapter describes the configuration container and parameters used for LinNm configuration.

10.3.1 LinNm

SWS Item	**ECUC_LinNm_00029 :**
Module Name	*LinNm*
Module Description	Configuration Parameters for the Lin Nm module.
Post-Build Variant Support	false
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-PRE-COMPILE

Included Containers		
Container Name	*Multiplicity*	*Scope / Dependency*
LinNmGlobalConfig	1	This container contains the global configuration parameter of the LinNm.

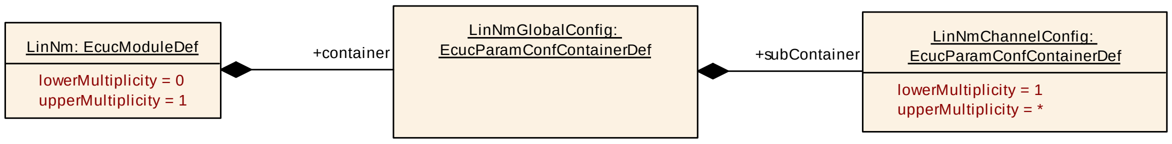

Figure 10-1 LinNm top level configuration overview

10.3.2 LinNmGlobalConfig

SWS Item	**ECUC_LinNm_00001 :**
Container Name	LinNmGlobalConfig
Description	This container contains the global configuration parameter of the LinNm.
Configuration Parameters	

SWS Item	**ECUC_LinNm_00015 :**		
Name	LinNmBusSynchronizationEnabled		
Parent Container	LinNmGlobalConfig		
Description	Pre-processor switch for enabling bus synchronization support of the LinNm. This feature is required for NM Coordinator nodes only.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: It must not be enabled if LINNM_PASSIVE_MODE_ENABLED is enabled.		

SWS Item	**ECUC_LinNm_00019 :**
Name	LinNmComControlEnabled
Parent Container	LinNmGlobalConfig

Description	Pre-processor switch for enabling the Communication Control support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinNm_00026 :**		
Name	LinNmCoordinatorSyncSupport		
Parent Container	LinNmGlobalConfig		
Description	Enables/disables the coordinator synchronization support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: LinNmCoordinatorSyncSupport has to be set to FALSE if LinNmPassiveModeEnabled is set to TRUE.		

SWS Item	**ECUC_LinNm_00003 :**		
Name	LinNmDevErrorDetect		
Parent Container	LinNmGlobalConfig		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinNm_00028 :**		
Name	LinNmMainFunctionPeriod		
Parent Container	LinNmGlobalConfig		
Description	Call cycle in seconds of LinNm_MainFunction.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinNm_00005 :**		
Name	LinNmPassiveModeEnabled		
Parent Container	LinNmGlobalConfig		
Description	Pre-processor switch for enabling support of the Passive Mode of the LinNm.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinNm_00016 :**		
Name	LinNmRemoteSleepIndicationEnabled		
Parent Container	LinNmGlobalConfig		
Description	Pre-processor switch for enabling Remote Sleep Indication support. This feature is required for NM Coordinator nodes only.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: It must not be enabled if LINNM_PASSIVE_MODE_ENABLED is enabled.		

SWS Item	**ECUC_LinNm_00018 :**		
Name	LinNmStateChangeIndEnabled		
Parent Container	LinNmGlobalConfig		
Description	Pre-processor switch for enabling the Network Management state change notification.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinNm_00022 :**		
Name	LinNmSynchronizationPointEnabled		
Parent Container	LinNmGlobalConfig		
Description	Pre-processor switch for enabling the Synchronize NM feature.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local dependency: Pre-processor switch for enabling the Synchronize NM feature.		

SWS Item	**ECUC_LinNm_00017 :**		
Name	LinNmUserDataEnabled		
Parent Container	LinNmGlobalConfig		
Description	Pre-processor switch for enabling User Data support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinNm_00004 :**		
Name	LinNmVersionInfoApi		
Parent Container	LinNmGlobalConfig		
Description	Pre-processor switch for enabling version info API support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
LinNmChannelConfig	1..*	This container contains the channel specific configuration parameter of the LinNm.

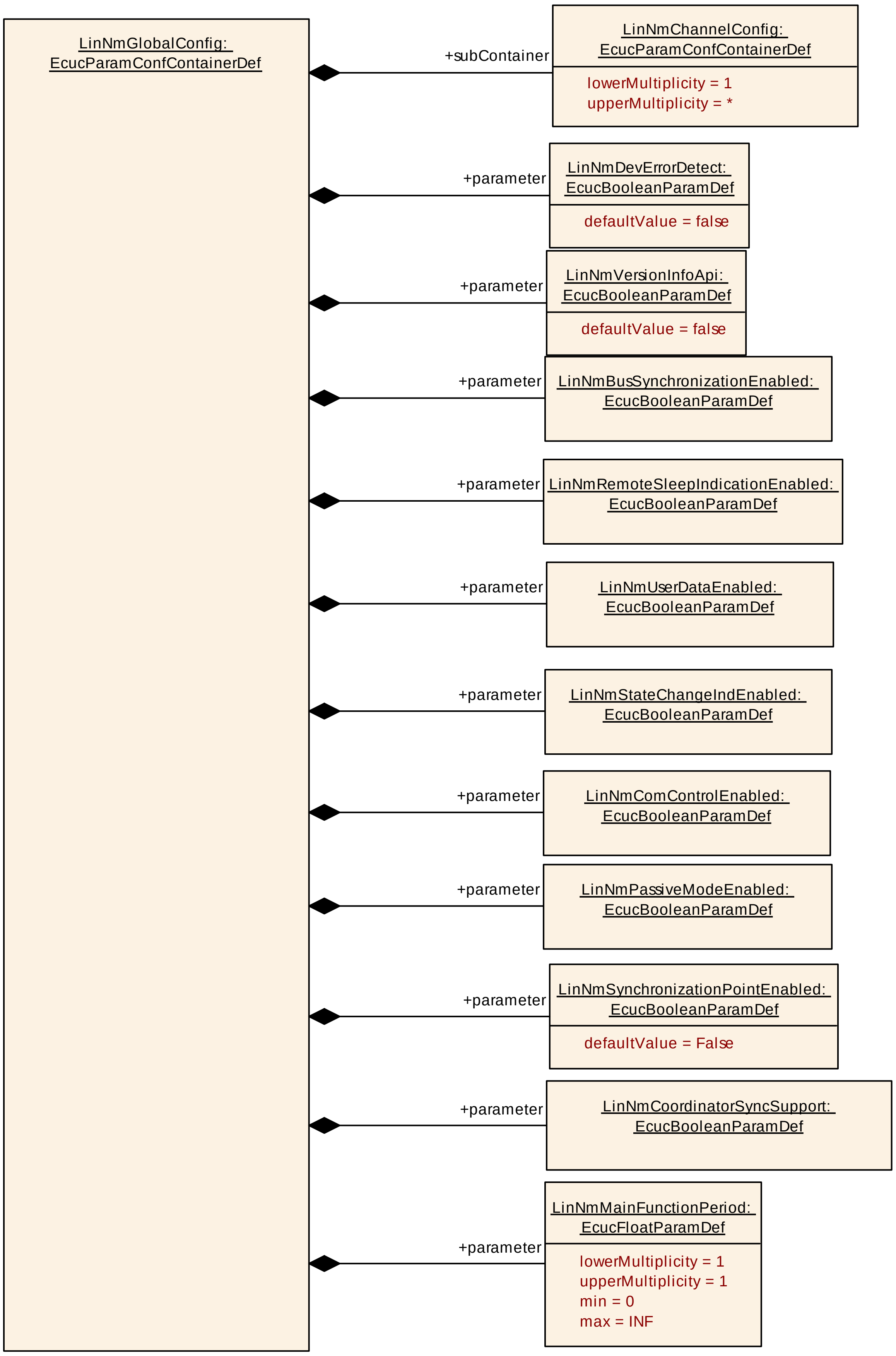
LinNmGlobalConfig: EcucParamConfContainerDef
+subContainer
LinNmChannelConfig: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
LinNmDevErrorDetect: EcucBooleanParamDef
defaultValue = false
+parameter
LinNmVersionInfoApi: EcucBooleanParamDef
defaultValue = false
+parameter
LinNmBusSynchronizationEnabled: EcucBooleanParamDef
+parameter
LinNmRemoteSleepIndicationEnabled: EcucBooleanParamDef
+parameter
LinNmUserDataEnabled: EcucBooleanParamDef
+parameter
LinNmStateChangeIndEnabled: EcucBooleanParamDef
+parameter
LinNmComControlEnabled: EcucBooleanParamDef
+parameter
LinNmPassiveModeEnabled: EcucBooleanParamDef
+parameter
LinNmSynchronizationPointEnabled: EcucBooleanParamDef
defaultValue = False
+parameter
LinNmCoordinatorSyncSupport: EcucBooleanParamDef
+parameter
LinNmMainFunctionPeriod: EcucFloatParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
min = 0
max = INF

Figure 10-2 Parameters of LinNm global configuration

10.3.3 LinNmChannelConfig

SWS Item	**ECUC_LinNm_00002 :**
Container Name	LinNmChannelConfig
Description	This container contains the channel specific configuration parameter of the LinNm.
Configuration Parameters	

SWS Item	**ECUC_LinNm_00030 :**		
Name	LinNmNodeDetectionEnabled		
Parent Container	LinNmChannelConfig		
Description	Pre-processor switch for enabling the Node Detection feature.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinNm_00031 :**		
Name	LinNmNodeIdEnabled		
Parent Container	LinNmChannelConfig		
Description	Pre-processor switch for enabling transmission of the source node identifier in NM messages.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinNm_00027 :**		
Name	LinNmTimeoutTime		
Parent Container	LinNmChannelConfig		
Description	Network Timeout after passive start-up. It denotes the time in seconds how long the NM shall stay in Network Mode in case of passive start-up before transition into Bus-Sleep Mode is initiated.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 65.535]		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_LinNm_00014 :**
Name	LinNmComMNetworkHandleRef

Parent Container	LinNmChannelConfig		
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	Symbolic name reference to [ComMChannel]		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

[SWS_LinNm_00099] ⌈The container LinNmChannelConfig specifies configuration parameter that shall be located in a data structure. ⌋ ( )

10.4 Published parameters

For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*

11 Not applicable requirements

[SWS_LinNm_00165] ⌈ These requirements are not applicable to this specification.⌋ (SRS_Lin_01564, SRS_Lin_01515, SRS_Lin_01523, SRS_BSW_00170, SRS_BSW_00387, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00429, SRS_BSW_00432, BSW00434, SRS_BSW_00336, SRS_BSW_00339, SRS_BSW_00417, SRS_BSW_00409, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00415, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00160, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00305, SRS_BSW_00307, SRS_BSW_00335, SRS_BSW_00410, SRS_BSW_00314, SRS_BSW_00328, SRS_BSW_00312, SRS_BSW_00006, SRS_BSW_00377, SRS_BSW_00306, SRS_BSW_00309, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00172, SRS_BSW_00010, SRS_BSW_00333, SRS_BSW_00321, SRS_BSW_00341, SRS_BSW_00334, SRS_Nm_00151, SRS_Nm_00043, SRS_Nm_00046, SRS_Nm_00048, SRS_Nm_00050, SRS_Nm_00051, SRS_Nm_00052, SRS_Nm_02509, SRS_Nm_02503, SRS_Nm_02504, SRS_Nm_00153, SRS_Nm_02508, SRS_Nm_02505, SRS_Nm_02506, SRS_Nm_02511, SRS_Nm_00053, SRS_Nm_00137, BSW136, BSW140, SRS_Nm_00054, SRS_Nm_00142, SRS_Nm_00143, SRS_Nm_00144, SRS_Nm_00145, SRS_Nm_00146, SRS_Nm_00147, SRS_Nm_00154, BSW139, SRS_Nm_02510, SRS_Nm_02512)