Document Title	Specification of LIN Network Management
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	297
Document Classification	Standard

Document Version	2.0.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Change Description
13.10.2011	2.0.0	AUTOSAR Administration	• Added support for NM Coordinator Synchronization • Changed Nm_ReturnType to Std_ReturnType • Updated “Module short name” to “Module Abbreviation”
15.10.2010	1.1.0	AUTOSAR Administration	• Channel ID of the LinNM is harmonized • Added DET check for LinNm_GetVersionInfo API • Requirement on Version Check of module is updated. • Added requirements for Passive Startup to clarify the behavior in sleep mode.
30.11.2009	1.0.0	AUTOSAR Administration	Initial release

Disclaimer

Advice for users

AUTOSAR specifications 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 specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, 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

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	Development 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 Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.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

3.2 Related standards and norms

Not available.

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.

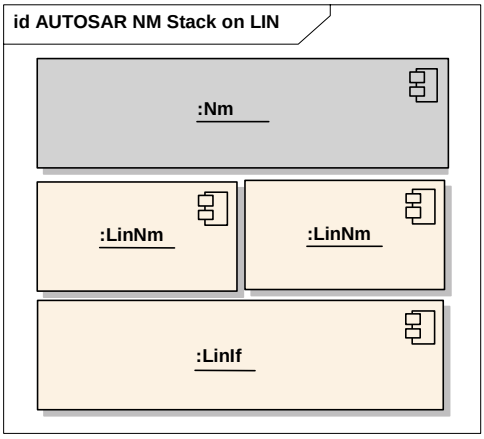

Figure 4-1

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).

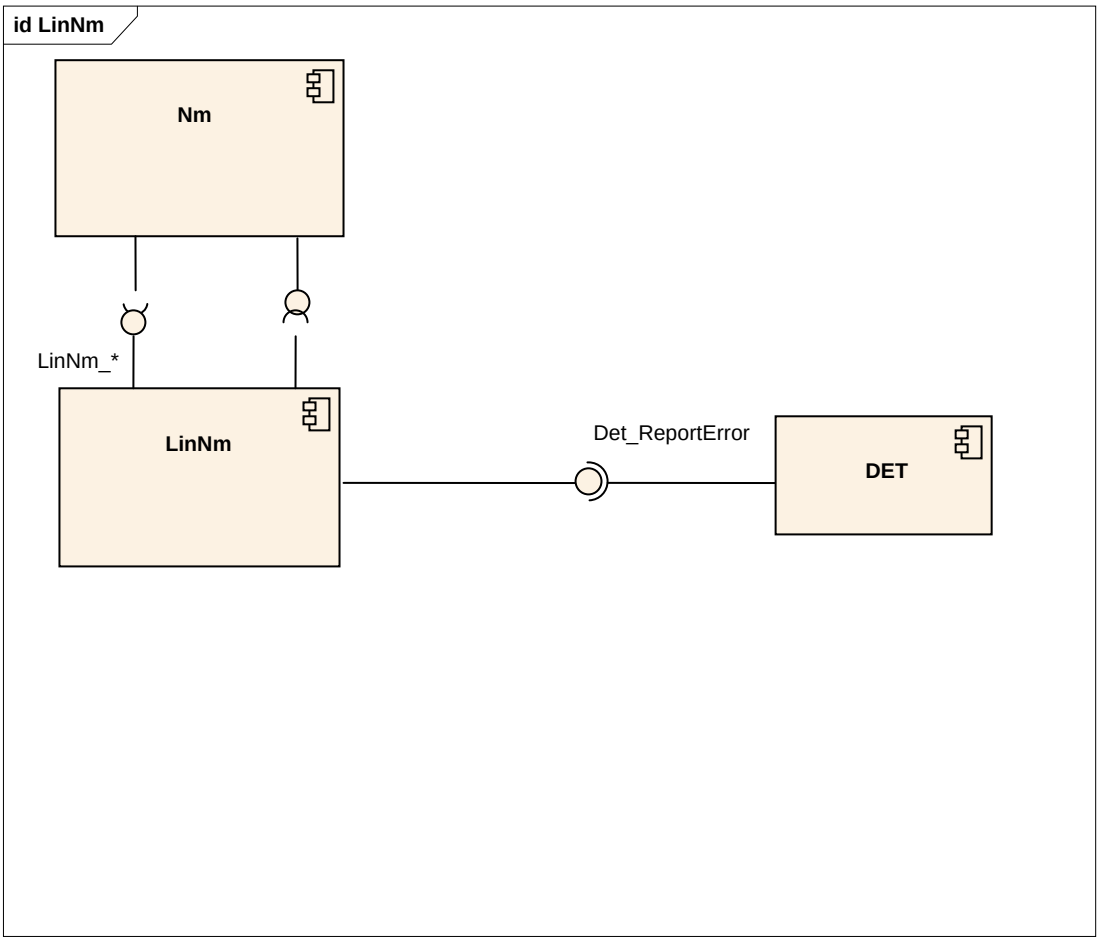

Figure 5-1 Dependencies to other modules

5.1 File Structure

5.1.1 Code File Structure

[LINNM000] ⌈The code file structure shall not be defined within this specification completely. ⌋(BSW00419, BSW00346, BSW158, BSW00308)

[LINNM137] ⌈At this point it shall be pointed out that the code-file structure shall include the following files named:

- LinNm_Cfg.c
- LinNm_Lcfg.c ⌋( )

[LINNM138] ⌈LinNm_Cfg.c shall contain pre-compile time configurable parameters. ⌋( )

[LINNM139] ⌈LinNm_Lcfg.c shall contain link time configurable parameters. ⌋( )

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

5.1.2 Header File Structure

[LINNM001] ⌈The LinNm module shall provide the following H-files.

- `LinNm.h` (for declaration of provided interface functions)
- `LinNm_Cbk.h` (for declaration of provided call-back functions)
- `LinNm_Cfg.h` (for pre-compile time configurable parameters) ⌋(BSW00345, BSW00380, BSW00381, BSW00412, BSW00346, BSW158, BSW00370, BSW00302)

[LINNM002] ⌈The LinNm module shall include the following H-files.

- `ComStack_Types.h`
  Note: The following header files are indirectly included by `ComStack_Types.h`
  - `Std_Types.h` (for AUTOSAR standard types)
  - `Platform_Types.h` (for platform specific types)
  - `Compiler.h` (for compiler specific language extensions)
- `LinNm.h` (for declaration of provided interface functions)
- `Nm_Cbk.h` (for LinNm specific callbacks of Generic Generic Network Management Interface)
- `Det.h` (for interface of DET – included only if DET is configured)
- `NmStack_Types.h` (for common network management types)
- `SchM_LinNm.h` (for services of the Basic Software Scheduler)
- `MemMap.h` (for Memory Mapping) ⌋(BSW00348, BSW00353, BSW00361, BSW00301)

[LINNM003] ⌈The LinNm module shall include the following header files containing configuration data.

- `Nm_Cfg.h` (for the derived configuration items from Nm) ⌋(BSW00301)

[LINNM144] ⌈The LinNm module shall include `PduR_LinNm.h` if LinNmComUserDataSupport is enabled. ⌋( )

6 Requirements traceability

Requirement	Satisfied by
-	LINNM022
-	LINNM025
-	LINNM113
-	LINNM130
-	LINNM108
-	LINNM102
-	LINNM017
-	LINNM110
-	LINNM096
-	LINNM019
-	LINNM126
-	LINNM091
-	LINNM038
-	LINNM094
-	LINNM156
-	LINNM118
-	LINNM026
-	LINNM124
-	LINNM070
-	LINNM018
-	LINNM006
-	LINNM161
-	LINNM054
-	LINNM104
-	LINNM045
-	LINNM151
-	LINNM071
-	LINNM008
-	LINNM061
-	LINNM120
-	LINNM121
-	LINNM135
-	LINNM055
-	LINNM041
-	LINNM147
-	LINNM158
-	LINNM042
-	LINNM092
-	LINNM069

-	LINNM112
-	LINNM163
-	LINNM127
-	LINNM004
-	LINNM129
-	LINNM139
-	LINNM031
-	LINNM136
-	LINNM140
-	LINNM153
-	LINNM148
-	LINNM123
-	LINNM106
-	LINNM012
-	LINNM160
-	LINNM064
-	LINNM144
-	LINNM044
-	LINNM105
-	LINNM072
-	LINNM159
-	LINNM095
-	LINNM016
-	LINNM029
-	LINNM014
-	LINNM020
-	LINNM116
-	LINNM078
-	LINNM141
-	LINNM058
-	LINNM122
-	LINNM093
-	LINNM063
-	LINNM114
-	LINNM117
-	LINNM089
-	LINNM005
-	LINNM046
-	LINNM119
-	LINNM037
-	LINNM028
-	LINNM128

-	LINNM157
-	LINNM131
-	LINNM150
-	LINNM015
-	LINNM033
-	LINNM138
-	LINNM056
-	LINNM162
-	LINNM115
-	LINNM111
-	LINNM043
-	LINNM040
-	LINNM125
-	LINNM053
-	LINNM065
-	LINNM109
-	LINNM149
-	LINNM154
-	LINNM034
-	LINNM103
-	LINNM030
-	LINNM090
-	LINNM137
BSW00301	LINNM003, LINNM002
BSW00302	LINNM001
BSW00305	LINNM165
BSW00306	LINNM165
BSW00307	LINNM165
BSW00308	LINNM000
BSW00309	LINNM165
BSW00312	LINNM165
BSW00314	LINNM165
BSW00321	LINNM165
BSW00323	LINNM048, LINNM047
BSW00325	LINNM165
BSW00326	LINNM165
BSW00328	LINNM165
BSW00330	LINNM165
BSW00331	LINNM165
BSW00333	LINNM165
BSW00334	LINNM165
BSW00335	LINNM165

BSW00336	LINNM165
BSW00339	LINNM165
BSW00341	LINNM165
BSW00345	LINNM001
BSW00346	LINNM001, LINNM000
BSW00347	LINNM165
BSW00348	LINNM002
BSW00353	LINNM002
BSW00361	LINNM002
BSW00370	LINNM001
BSW00375	LINNM165
BSW00377	LINNM165
BSW00380	LINNM001
BSW00381	LINNM001
BSW00387	LINNM165
BSW004	LINNM073
BSW00409	LINNM165
BSW00410	LINNM165
BSW00412	LINNM001
BSW00413	LINNM165
BSW00415	LINNM165
BSW00416	LINNM165
BSW00417	LINNM165
BSW00419	LINNM000
BSW00423	LINNM165
BSW00424	LINNM165
BSW00425	LINNM165
BSW00426	LINNM165
BSW00427	LINNM165
BSW00429	LINNM165
BSW00432	LINNM165
BSW00434	LINNM165
BSW005	LINNM165
BSW006	LINNM165
BSW010	LINNM165
BSW01515	LINNM165
BSW01523	LINNM165
BSW01564	LINNM165
BSW02503	LINNM165
BSW02504	LINNM165
BSW02505	LINNM165
BSW02506	LINNM165

BSW02508	LINNM165
BSW02509	LINNM165
BSW02510	LINNM165
BSW02511	LINNM165
BSW02512	LINNM165
BSW043	LINNM165
BSW046	LINNM165
BSW048	LINNM165
BSW050	LINNM165
BSW051	LINNM165
BSW052	LINNM165
BSW053	LINNM165
BSW054	LINNM165
BSW136	LINNM165
BSW137	LINNM165
BSW139	LINNM165
BSW140	LINNM165
BSW142	LINNM165
BSW143	LINNM165
BSW144	LINNM165
BSW145	LINNM165
BSW146	LINNM165
BSW147	LINNM165
BSW151	LINNM165
BSW153	LINNM165
BSW154	LINNM165
BSW158	LINNM001, LINNM000
BSW160	LINNM165
BSW161	LINNM165
BSW162	LINNM165
BSW164	LINNM165
BSW168	LINNM165
BSW170	LINNM165
BSW172	LINNM165

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:

[LINNM004] ⌈If LinNm_NetworkRelease is called in the Network mode then mode shall be changed to Bus Sleep mode.⌋( )

[LINNM161] ⌈If LinNm_PassiveStartUp is called in Bus Sleep Mode, then mode shall be changed to Network mode.⌋( )

[LINNM162] ⌈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.

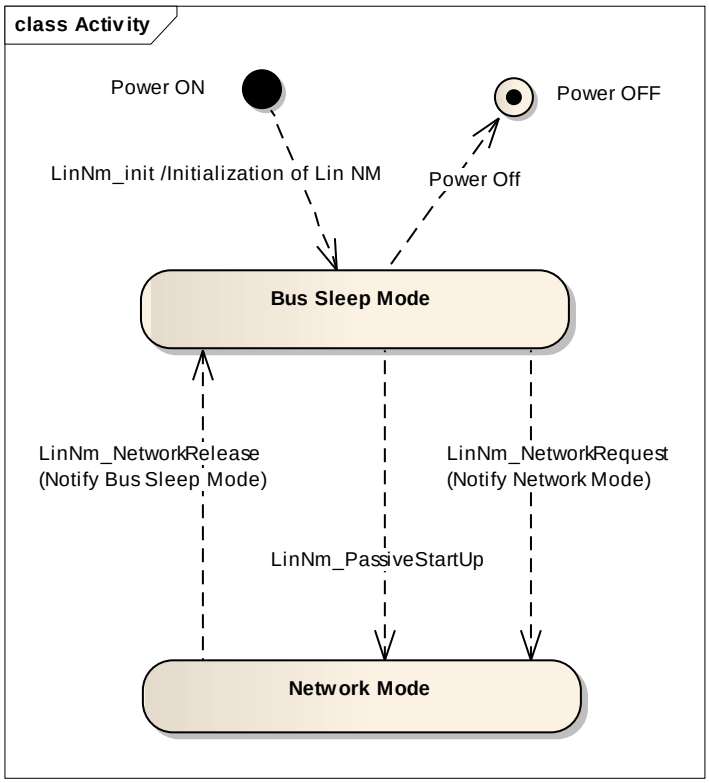

Figure 7-1

7.2 Operational Modes

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

[LINNM005] ⌈The AUTOSAR LinNm shall contain two operational modes visible at the module's interface:

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

[LINNM006] ⌈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

[LINNM008] ⌈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.`** ⌋ ( )

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.

[LINNM012] ⌈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.

[LINNM014] ⌈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').

[LINNM015] ⌈The function call **LinNm_NetworkRequest** shall request the network. I.e. the LinNm module shall change network state to 'NM_STATE_NORMAL_OPERATION'.⌋( )

[LINNM016] ⌈The function call **LinNm_NetworkRelease** shall release the network. I.e. the LinNm module shall change network state to 'NM_STATE_BUS_SLEEP'.⌋( )

[LINNM160] ⌈If **LinNm_PassiveStartUp** is called in Bus Sleep Mode, then LinNm shall change network state to NM_STATE_NORMAL_OPERATION.⌋( )

[LINNM103] ⌈The Modes and States shall be available for debugging.⌋( )

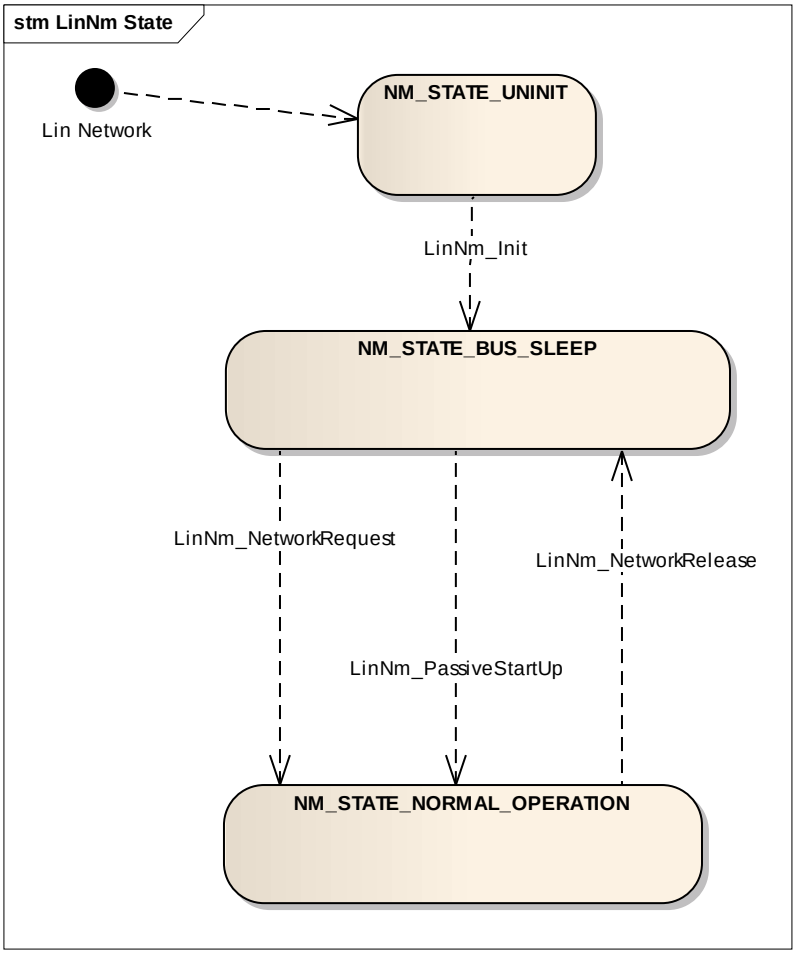

Figure 7-2

7.4 Initialization

[LINNM017] ⌈During initialization of the LinNm module (**LinNm_Init**) the LinNm module shall set the Network Management State to NM_STATE_UNINIT. ⌋ ( )

[LINNM018] ⌈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. ⌋ ( )

[LINNM102] ⌈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.

[LINNM019] ⌈If initialized, by default, the LinNm module shall set the network state to NM_STATE_BUS_SLEEP. ⌋ ( )

[LINNM020] ⌈If initialized, by default, the LinNm module shall enter the Bus-Sleep Mode. ⌋ ( )

[LINNM022] ⌈If `LinNm_PassiveStartUp` is called in the Network Mode, the LinNm module shall not execute this service and shall return E_NOT_OK. ⌋ ( )

[LINNM156] ⌈If `LinNm_NetworkRequest` is called in the Network Mode, the LinNM module shall not execute this service and shall return E_NOT_OK. ⌋ ( )

[LINNM157] ⌈If `LinNm_NetworkRelease` is called in the Bus Sleep Mode, the LinNM module shall not execute this service and shall return E_NOT_OK. ⌋ ( )

[LINNM025] ⌈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

[LINNM026] ⌈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

[LINNM061]「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

[LINNM028]「Development error values are of type uint8」( )

[LINNM029]「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_NO_INIT	0x01
API service called with wrong channel handle	Development	LINNM_E_INVALID_CHANNEL	0x02
Null pointer has been passed as an argument (Does not apply to function LinNm_Init)	Development	LINNM_E_PARAM_POINTER	0x12

」( )

7.8 Error detection

[LINNM030]「The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch **`LINNM_DEV_ERROR_DETECT`** (see chapter 10) shall activate or deactivate the detection of all development errors.」( )

[LINNM031] ⌈If the **LINNM_DEV_ERROR_DETECT** switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.7 and chapter 8. ⌋( )

7.9 Error notification

[LINNM033] ⌈Detected development errors shall be reported to the **Det_ReportError** service of the Development Error Tracer (DET) if the pre-processor switch **LINNM_DEV_ERROR_DETECT** is set (see chapter 10). ⌋( )

[LINNM034] ⌈The LinNm module shall not return development errors by API functions; in case of a development error, the respective API function shall return **E_NOT_OK**, if applicable. ⌋( )

[LINNM037] ⌈Each LinNm function that is not executed due to missing initialization of LinNm shall report **LINNM_E_NO_INIT** to the Development Error Tracer and it shall return **E_NOT_OK** to the calling function. ⌋( )

[LINNM038] ⌈When LinNm service with an invalid network handle is called, the called function shall not be executed, but instead of that it shall report **LINNM_E_INVALID_CHANNEL** to the Development Error Tracer (the value of the invalid network handle shall be passed to DET as instance ID) and it shall return **E_NOT_OK** to the calling function.

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

[LINNM040] ⌈The LinNm module shall provide mechanisms that support coordination of coupled networks with the following services:

- Bus synchronization on demand
  Bus synchronization on demand allows synchronization of a network management cluster for an arbitrary point of

time; in result NM-Timeout Timers in all nodes of the network management cluster are restarted. ⌋ ( )

[LINNM041] ⌈Support of bus synchronization on demand shall be statically configurable with use of the **LINNM_BUS_SYNCHRONIZATION_ENABLED** switch (configuration parameter). ⌋ ( )

[LINNM042] ⌈If **LinNm_RequestBusSynchronization** is called in Bus-Sleep Mode, the LinNm module shall not execute the service and shall return **E_OK**. ⌋ ( )

[LINNM140] ⌈The parameter LINNM_SYNCHRONIZATIONPOINT_ENABLED shall be always disabled. ⌋ ( )

[LINNM141] ⌈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.

[**LINNM169]** ⌈This feature is optional and only available if `LINNM_COORDINATOR_SYNC_SUPPORT` is set to `TRUE`.⌋().

7.10.4 Debugging Concept

[LINNM043] ⌈Each variable that shall be accessible by AUTOSAR Debugging shall be defined as global variable. ⌋ ( )

[LINNM044] ⌈All type definitions of variables which shall be debugged shall be accessible by the header file LinNm.h. ⌋ ( )

[LINNM045] ⌈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".⌋ ( )

[LINNM046] ⌈Variables available for debugging shall be described in the respective Basic Software Module Description. ⌋ ( )

8 API specification

[LINNM047] ⌈The LinNm module shall provide parameter value check only in "development mode".⌋ (BSW00323)

[LINNM048] ⌈The LinNm module shall reject the execution of a service called with an invalid parameter and shall inform the DET. ⌋ (BSW00323)

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 files are listed:

[LINNM078] ⌈

Module	*Imported Type*
ComStack_Types	PduIdType
	NetworkHandleType
	PduInfoType
Nm	Nm_ModeType
	Nm_StateType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋ ( )

8.2 Type Definitions

8.3 LinNm Functions called by the Nm

8.3.1 LinNm_Init

[LINNM054] ⌈

Service name:	LinNm_Init

Syntax:	`void LinNm_Init(`
	`  void`
	`)`

Service ID[hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None

Return value:	None
Description:	Initialize the complete LinNm module.

⌋( )

8.3.2 LinNm_PassiveStartUp

[LINNM063] ⌈

Service name:	LinNm_PassiveStartUp	
Syntax:	`Std_ReturnType LinNm_PassiveStartUp( const NetworkHandleType nmChannelHandle )`	
Service ID[hex]:	0x01	
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):	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.	

⌋( )

[LINNM064] ⌈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. ⌋( )

[LINNM065] ⌈Caveats of LinNm_PassiveStartUp: The LinNm module is initialized correctly. ⌋( )

8.3.3 LinNm_NetworkRequest

[LINNM055] ⌈

Service name:	LinNm_NetworkRequest	
Syntax:	`Std_ReturnType LinNm_NetworkRequest( const NetworkHandleType nmChannelHandle )`	
Service ID[hex]:	0x02	
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):	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.	

⌋( )

[LINNM053] ⌈Caveats of LinNm_NetworkRequest: The LinNm module is initialized correctly.⌋( )

[LINNM158] ⌈Configuration of LinNm_NetworkRequest: This function is only available if `LINNM_PASSIVE_MODE_ENABLED` is set to FALSE.⌋( )

8.3.4 LinNm_NetworkRelease

[LINNM056] ⌈

Service name:	LinNm_NetworkRelease	
Syntax:	`Std_ReturnType LinNm_NetworkRelease( const NetworkHandleType nmChannelHandle )`	
Service ID[hex]:	0x03	
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):	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.	

⌋( )

[LINNM058] ⌈Caveats of LinNm_NetworkRelease: The LinNm module is initialized correctly.⌋( )

[LINNM159] ⌈Configuration of LinNm_NetworkRelease: This function is only available if `LINNM_PASSIVE_MODE_ENABLED` is set to FALSE.⌋( )

8.3.5 LinNm_GetVersionInfo

[LINNM106] ⌈

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.	

⌋( )

[LINNM104] ⌈The function LinNm_GetVersionInfo shall return the version information of this module. The version information includes:

- ***Module Id***
- ***Vendor Id***
- Vendor specific version numbers. ⌋( )

Note: This function can be called even if LinNm is not initialized.

[LINNM105] ⌈Configuration of LinNm_GetVersionInfo: Optional (only available if LINNM_VERSION_INFO_API is defined). ⌋( )

[LINNM163] ⌈If DET is enabled for the LinNm module, the function LinNm_GetVersionInfo shall raise LINNM_E_PARAM_POINTER, if the argument is a NULL pointer and return without any action. ⌋( )

8.3.6 LinNm_RequestBusSynchronization

[LINNM089] ⌈

Service name:	LinNm_RequestBusSynchronization	
Syntax:	`Std_ReturnType LinNm_RequestBusSynchronization( const NetworkHandleType nmChannelHandle )`	
Service ID[hex]:	0xc0	
Sync/Async:	Synchronous	
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.	

⌋( )

[LINNM095] ⌈Service call **`LinNm_RequestBusSynchronization`** shall provide an empty implementation. ⌋( )

[LINNM090] ⌈Caveats of LinNm_RequestBusSynchronization: The LinNm module is initialized correctly. ⌋( )

[LINNM091] ⌈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

[LINNM092]⌈

Service name:	LinNm_CheckRemoteSleepIndication	
Syntax:	`Std_ReturnType LinNm_CheckRemoteSleepIndication( const NetworkHandleType nmChannelHandle, boolean* const 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.	

⌋( )

[LINNM093] ⌈Service call **`LinNm_CheckRemoteSleepIndication`** shall provide an empty implementation.⌋( )

[LINNM094]⌈Caveats of LinNm_CheckRemoteSleepIndication: The LinNm module and Nm module shall be initialized correctly.⌋( )

[LINNM096]⌈Configuration of LinNm_CheckRemoteSleepIndication: Optional (Only available if `LINNM_REMOTE_SLEEP_INDICATION_ENABLED` is defined)⌋( )

8.3.8 LinNm_SetSleepReadyBit

[LINNM166]

Service name:	LinNm_SetSleepReadyBit	
Syntax:	`Std_ReturnType LinNm_SetSleepReadyBit( const NetworkHandleType nmChannelHandle, const 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.	

[LINNM167] Configuration of LinNm_SetSleepReadyBit: This function is only available if LINNM_COORDINATOR_SYNC_SUPPORT is set to TRUE. ⌋( )

[LINNM168] 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

[LINNM108] ⌈

Service name:	LinNm_DisableCommunication	
Syntax:	`Std_ReturnType LinNm_DisableCommunication( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x04	
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):	None	
Return value:	Std_ReturnType	E_OK: No error
Description:	Empty function to be complaint with NM specifications.	

⌋( )

[LINNM109] ⌈Caveats of LinNm_DisableCommunication: The **LinNm** and the **Nm** itself are initialized correctly. ⌋( )

[LINNM110] ⌈Configuration of LinNm_DisableCommunication: This function is only available if LINNM_COM_CONTROL_ENABLED is set to TRUE. ⌋( )

8.3.9.2 LinNm_EnableCommunication

[LINNM111] ⌈

Service name:	LinNm_EnableCommunication
Syntax:	`Std_ReturnType LinNm_EnableCommunication( const NetworkHandleType NetworkHandle )`
Service ID[hex]:	0x05
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):	None	
Return value:	Std_ReturnType	E_OK: No error
Description:	Empty function to be complaint with NM specifications.	

⌋( )

[LINNM112] ⌈Caveats of LinNm_EnableCommunication: The **LinNm** and the **Nm** itself are initialized correctly. ⌋( )

[LINNM113] ⌈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

[LINNM114] ⌈

Service name:	LinNm_SetUserData	

Syntax:	`Std_ReturnType LinNm_SetUserData(`
	`  const NetworkHandleType NetworkHandle,`
	`  const uint8 * const 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.	

⌋( )

[LINNM115] ⌈Caveats of LinNm_SetUserData: The **LinNm** and the **Nm** itself are initialized correctly. ⌋( )

[LINNM116] ⌈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. ⌋( )

[LINNM147] ⌈If LinNmComUserDataSupport is enabled the API LinNm_SetUserData shall not be available.⌋ ( )

8.3.10.2 LinNm_GetUserData

[LINNM117] ⌈

Service name:	LinNm_GetUserData	
Syntax:	`Std_ReturnType LinNm_GetUserData( const NetworkHandleType NetworkHandle, uint8 * const nmUserDataPtr )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	NetworkHandle	Identification of the NM-channel
Parameters (inout):	None	
Parameters (out):	nmUserDataPtr	Pointer 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.	

⌋ ( )

[LINNM118] ⌈Caveats of LinNm_GetUserData: The **LinNm** and the **Nm** itself are initialized correctly.⌋ ( )

[LINNM119] ⌈Configuration of LinNm_GetUserData: This function is only available if LINNM_USER_DATA_ENABLED is set to TRUE.⌋ ( )

8.3.10.3 LinNm_GetPduData

[LINNM120] ⌈

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

⌋ ( )

[LINNM121] ⌈Caveats of LinNm_GetPduData: The **LinNm** and the **Nm** itself are initialized correctly.⌋( )

[LINNM122] ⌈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

[LINNM123] ⌈

Service name:	LinNm_RepeatMessageRequest	
Syntax:	`Std_ReturnType LinNm_RepeatMessageRequest( const NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x09	
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):	None	
Return value:	Std_ReturnType	E_OK: No error
Description:	Empty function to be complaint with NM specifications.	

⌋( )

[LINNM124] ⌈Caveats of LinNm_RepeatMessageRequest: **LinNm** and **Nm** itself are initialized correctly.⌋( )

[LINNM125] ⌈Configuration of LinNm_RepeatMessageRequest: This function is only available if LINNM_NODE_DETECTION_ENABLED is TRUE.⌋( )

8.3.10.5 LinNm_GetNodeIdentifier

[LINNM126] ⌈

Service name:	LinNm_GetNodeIdentifier	
Syntax:	`Std_ReturnType LinNm_GetNodeIdentifier( const NetworkHandleType NetworkHandle, uint8 * const 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.	

⌋( )

[LINNM127] ⌈Caveats of LinNm_GetNodeIdentifier: The **LinNm** and the **Nm** itself are initialized correctly.⌋( )

[LINNM128] ⌈Configuration of LinNm_GetNodeIdentifier: This function is only available if LINNM_NODE_ID_ENABLED is set to TRUE.⌋( )

8.3.10.6 LinNm_GetLocalNodeIdentifier

[LINNM129] ⌈

Service name:	LinNm_GetLocalNodeIdentifier	
Syntax:	`Std_ReturnType LinNm_GetLocalNodeIdentifier( const NetworkHandleType NetworkHandle, uint8 * const 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.	

⌋( )

[LINNM130] ⌈Caveats of LinNm_GetLocalNodeIdentifier: The **LinNm** and the **Nm** itself are initialized correctly.⌋( )

[LINNM131] ⌈Configuration of LinNm_GetLocalNodeIdentifier: This function is only available if LINNM_NODE_ID_ENABLED is set to TRUE.⌋( )

8.3.10.7 LinNm_GetState

[LINNM135] ⌈

Service name:	LinNm_GetState
Syntax:	`Std_ReturnType LinNm_GetState( const NetworkHandleType nmNetworkHandle, Nm_StateType* const nmStatePtr, Nm_ModeType* const 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).	

」( )

[LINNM136] 「Caveats of LinNm_GetState: The **LinNm** and the **Nm** itself are initialized correctly. 」( )

8.3.10.8 LinNm_Transmit

[LINNM148] 「

Service name:	LinNm_Transmit	
Syntax:	`Std_ReturnType LinNm_Transmit( PduIdType LinTxPduId, const PduInfoType* PduInfoPtr )`	
Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	LinTxPduId	Upper layer identification of the LIN frame to be transmitted (not the LIN protected ID). This parameter is used to determine the corresponding LIN protected ID (PID) and implicitly the LIN Driver instance as well as the corresponding LIN Controller device.
	PduInfoPtr	Pointer to a structure with frame related data: DLC and pointer to frame data buffer. This parameter is not used by this call.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_NOT_OK: returns always
Description:	Empty function to be complaint with NM specifications. Always return E_NOT_OK	

」( )

[LINNM149] 「Service call **`LinNm_Transmit`** shall provide an empty implementation」( )

[LINNM150] 「Caveats of LinNm_Transmit: The **LinNm** and the **Nm** itself are initialized correctly. 」( )

[LINNM151] 「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

[LINNM153]⌈

Service name:	LinNm_TxConfirmation	

Syntax:	`void LinNm_TxConfirmation(`
	`  PduIdType TxPduId`
	`)`	

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.	

⌋( )

[LINNM154]⌈Caveats of LinNm_TxConfirmation: The **LinNm** and the **Nm** itself are initialized correctly.⌋( )

8.4 Scheduled Functions

Not available

8.5 Expected Interfaces

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

8.5.1 Mandatory Interfaces

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

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

8.5.2 Optional Interfaces

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

API function	*Description*
Det_ReportError	Service to report development errors.
Nm_CoordReadyToSleepIndication	Sets an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set
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.

8.5.3 Configurable interfaces

Not applicable

8.5.4 Job End Notification

Not applicable

8.6 Parameter check

[LINNM069] ⌈If detection of development errors is enabled by `LINNM_DEV_ERROR_DETECT` (configuration parameter), then for all LinNm API services

validity check of input parameters shall be made. ⌋( )

[LINNM070] ⌈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. ⌋( )

[LINNM071] ⌈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. ⌋( )

[LINNM072] ⌈Parameter value check (for parameters of the variable value) shall be made at execution time; if the value is invalid, execution of a service shall be rejected and respective development error shall be reported. ⌋( )

8.7 Version check

[LINNM073] ⌈The LinNm module shall perform Inter Module Checks to avoid integration of incompatible files. ⌋(BSW004)

The imported included files shall be checked by preprocessing directives.

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

Where <MODULENAME> is the module abbreviation of the other (external) modules, which provide header files, included by the LinNm module.

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

9 Sequence diagrams

9.1 LinNm_Init

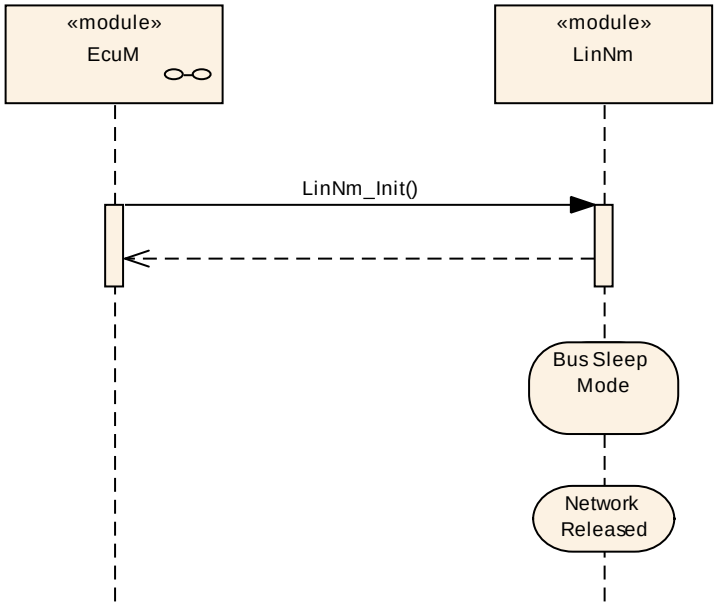

Figure 9-1 LinNm_init

9.2 LinNm_PassiveStartUp

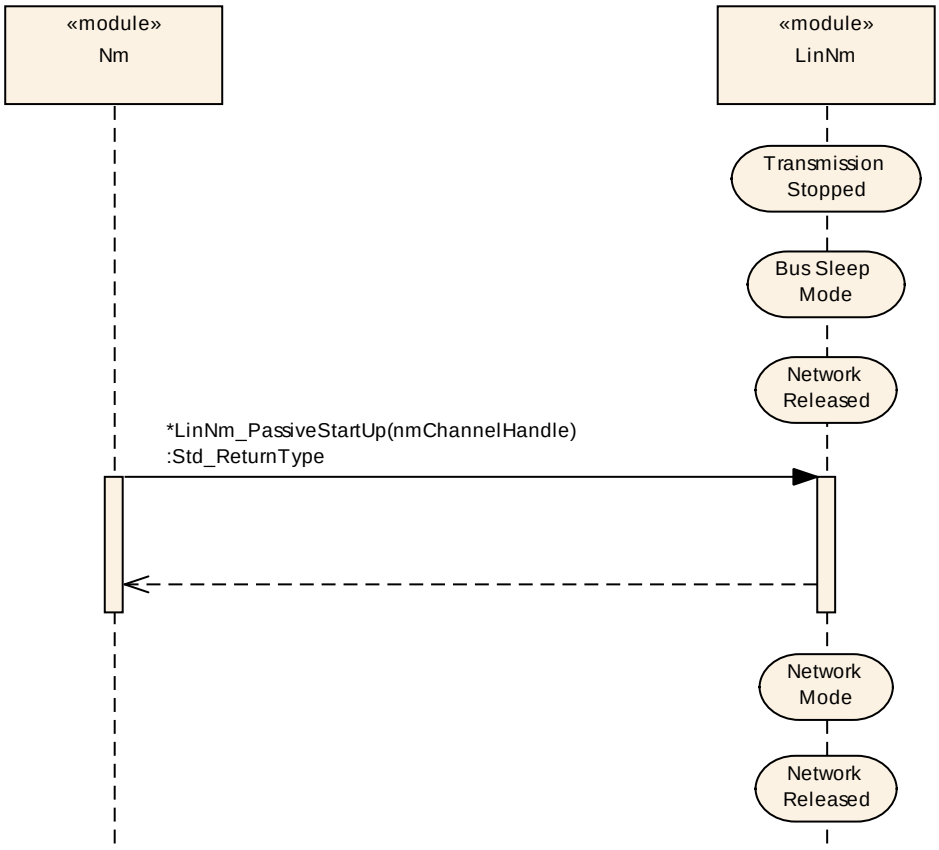

Figure 9-2 LinNm Passive Startup

9.3 LinNm_NormalOperation

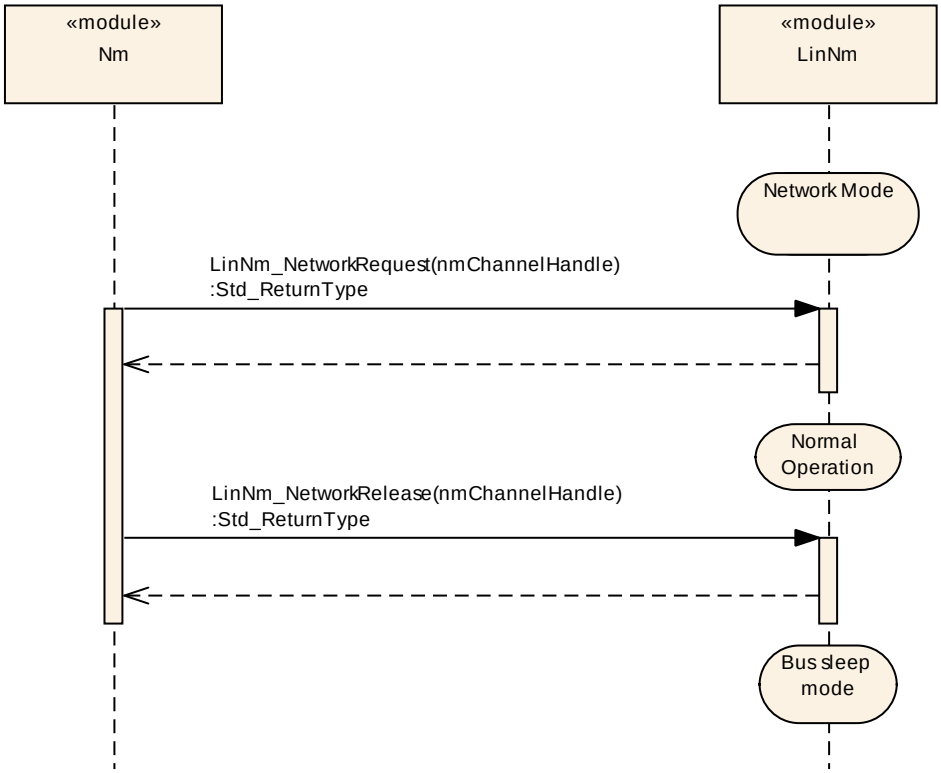

Figure 9-3 LinNm Normal Operation

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

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

- AUTOSAR Layered Software Architecture [1]
- AUTOSAR ECU Configuration Specification [6]
  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: The pre-compile 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 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.

[LINNM074] ⌈All configuration items shall be located outside the kernel of the module.⌋ ( )

10.2.1 Variants

[LINNM075] ⌈Variant 1: All configuration parameters shall be configurable at pre-compile time.

Use case: Source code optimizations⌋ ( )

[LINNM076] ⌈Variant 2: All configuration parameters of the container **`LinNm_GlobalConfig`** related to enable or disable a configurable feature shall be configurable at pre-compile time; the remaining configuration parameters shall be configurable at link time.

Use case: Object code.⌋ ( )

[LINNM077] ⌈Variant 3: The parameters contained in **`LinNm_GlobalConfig`** are configurable at pre-compile time

Use case: ECU configuration can be flashed (L) and selected during initialization phase (M).⌋ ( )

Note:
The possibility to select a configuration (post-build time type L) is only explicitly mentioned for Variant 3, but from a technical perspective it is also possible to provide this configuration variant for variant 1 and 2.

10.3 Containers and configuration parameters

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

10.3.1 LinNm

Module Name	*LinNm*
Module Description	Configuration Parameters for the Lin Nm module.

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

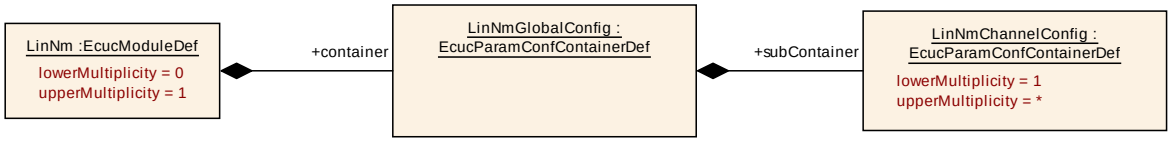

Figure 10-1 LinNm top level configuration overview

10.3.2 LinNmGlobalConfig

SWS Item	**LinNm001_Conf :**
Container Name	LinNmGlobalConfig{LINNM_GLOBAL_CONFIG}
Description	This container contains the global configuration parameter of the LinNm.
Configuration Parameters	

SWS Item	**LinNm015_Conf :**		
Name	LinNmBusSynchronizationEnabled {LINNM_BUS_SYNCHRONIZATION_ENABLED}		
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: It must not be enabled if LINNM_PASSIVE_MODE_ENABLED is enabled.		

SWS Item	**LinNm019_Conf :**
Name	LinNmComControlEnabled {LINNM_COM_CONTROL_ENABLED}
Description	Pre-processor switch for enabling the Communication Control 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	**LinNm025_Conf :**		
Name	LinNmComUserDataSupport {LINNM_COM_USER_DATA_SUPPORT}		
Description	Pre-processor switch for enabling the NM COM 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	**LinNm026_Conf :**		
Name	LinNmCoordinatorSyncSupport {LINNM_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	**LinNm003_Conf :**		
Name	LinNmDevErrorDetect {LINNM_DEV_ERROR_DETECT}		
Description	Pre-processor switch for enabling development error detection 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	**LinNm020_Conf :**		
Name	LinNmNodeDetectionEnabled {LINNM_NODE_DETECTION_ENABLED}		
Description	Pre-processor switch for enabling the Node Detection feature.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**LinNm021_Conf :**
Name	LinNmNodeIdEnabled {LINNM_NODE_ID_ENABLED}
Description	Pre-processor switch for enabling transmission of the source

	node identifier in 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	**LinNm005_Conf :**		
Name	LinNmPassiveModeEnabled {LINNM_PASSIVE_MODE_ENABLED}		
Description	Pre-processor switch for enabling support of the Passive Mode of the LinNm.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**LinNm016_Conf :**		
Name	LinNmRemoteSleepIndicationEnabled {LINNM_REMOTE_SLEEP_INDICATION_ENABLED}		
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: It must not be enabled if LINNM_PASSIVE_MODE_ENABLED is enabled.		

SWS Item	**LinNm018_Conf :**		
Name	LinNmStateChangeIndEnabled {LINNM_STATE_CHANGE_IND_ENABLED}		
Description	Pre-processor switch for enabling the Network Management state change notification.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**LinNm022_Conf :**
Name	LinNmSynchronizationPointEnabled {LINNM_SYNCHRONIZATIONPOINT_ENABLED}
Description	Pre-processor switch for enabling the Synchronize NM feature.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false

ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: Pre-processor switch for enabling the Synchronize NM feature.		

SWS Item	**LinNm017_Conf :**		
Name	LinNmUserDataEnabled {LINNM_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	**LinNm004_Conf :**		
Name	LinNmVersionInfoApi {LINNM_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		

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

[LINNM098] ⌈The Global Scope specifies configuration parameter that shall be defined in the module's configuration header file **LinNm_Cfg.h.** ⌋ ( )

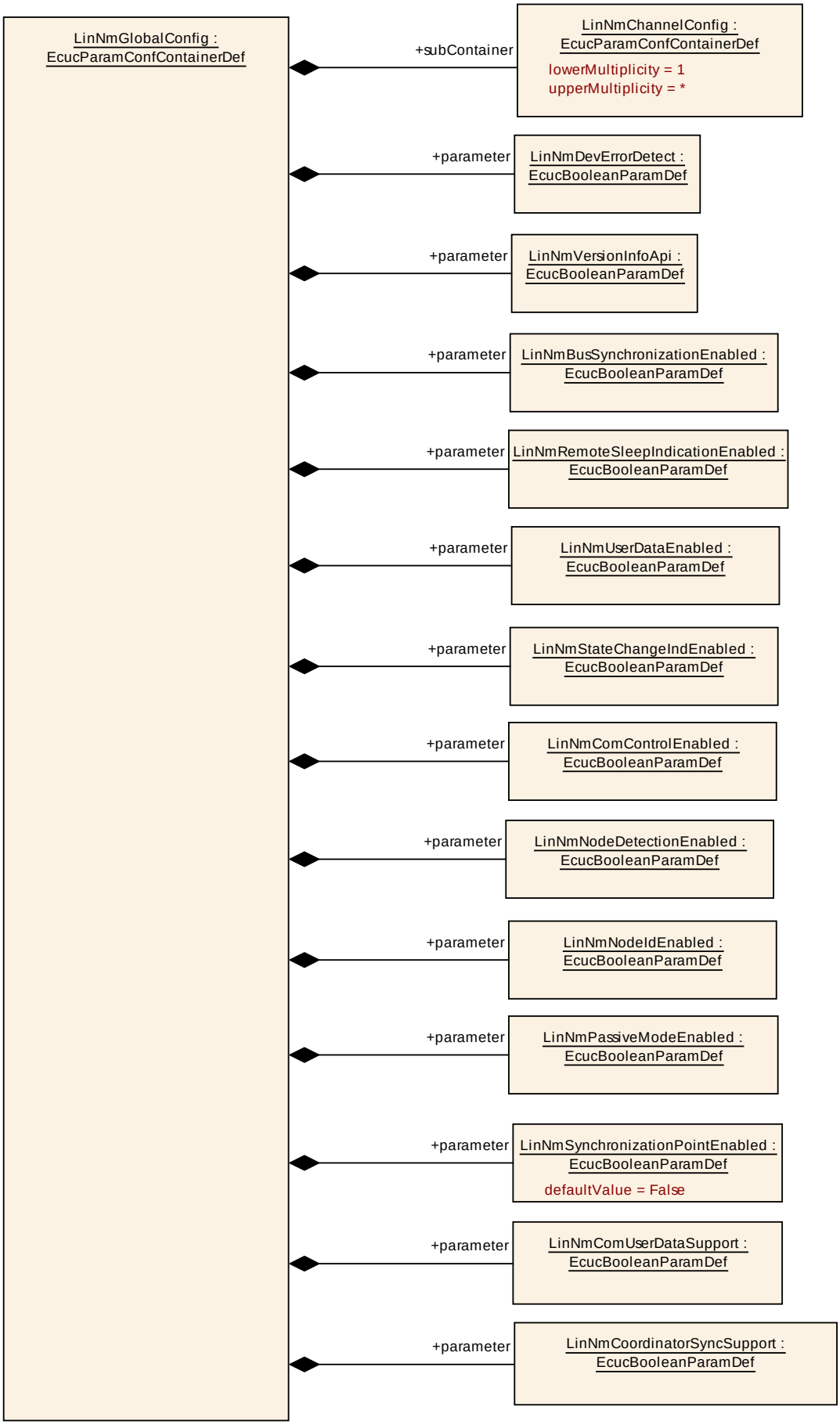
LinNmGlobalConfig :
EcucParamConfContainerDef
+subContainer
LinNmChannelConfig :
EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
LinNmDevErrorDetect :
EcucBooleanParamDef
+parameter
LinNmVersionInfoApi :
EcucBooleanParamDef
+parameter
LinNmBusSynchronizationEnabled :
EcucBooleanParamDef
+parameter
LinNmRemoteSleepIndicationEnabled :
EcucBooleanParamDef
+parameter
LinNmUserDataEnabled :
EcucBooleanParamDef
+parameter
LinNmStateChangeIndEnabled :
EcucBooleanParamDef
+parameter
LinNmComControlEnabled :
EcucBooleanParamDef
+parameter
LinNmNodeDetectionEnabled :
EcucBooleanParamDef
+parameter
LinNmNodeIdEnabled :
EcucBooleanParamDef
+parameter
LinNmPassiveModeEnabled :
EcucBooleanParamDef
+parameter
LinNmSynchronizationPointEnabled :
EcucBooleanParamDef
defaultValue = False
+parameter
LinNmComUserDataSupport :
EcucBooleanParamDef
+parameter
LinNmCoordinatorSyncSupport :
EcucBooleanParamDef

Figure 10-2 Parameters of LinNm global configuration

10.3.3 LinNmChannelConfig

SWS Item	**LinNm002_Conf :**
Container Name	LinNmChannelConfig{LINNM_CHANNEL_CONFIG}
Description	This container contains the channel specific configuration parameter of the LinNm.
Configuration Parameters	

SWS Item	**LinNm014_Conf :**		
Name	LinNmComMNetworkHandleRef {LINNM_COMM_NETWORK_HANDLE_REF}		
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	--	
Scope / Dependency			

No Included Containers

[LINNM099] ⌈The container LinNmChannelConfig specifies configuration parameter that shall be located in a data structure. ⌋ ( )

10.4 Published parameters

[LINNM164] ⌈The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [2]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 [17]. ⌋ ( )

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

11 Not applicable requirements

[LINNM165] ⌈These requirements are not applicable to this specification.⌋

(BSW01564, BSW01515, BSW01523, BSW170, BSW00387, BSW00375, BSW00416, BSW168, BSW00423, BSW00424, BSW00425, BSW00426, BSW00427, BSW00429, BSW00432, BSW00434, BSW00336, BSW00339, BSW00417, BSW00409, BSW161, BSW162, BSW005, BSW00415, BSW164, BSW00325, BSW00326, BSW160, BSW00413, BSW00347, BSW00305, BSW00307, BSW00335, BSW00410, BSW00314, BSW00328, BSW00312, BSW006, BSW00377, BSW00306, BSW00309, BSW00330, BSW00331, BSW172, BSW010, BSW00333, BSW00321, BSW00341, BSW00334, BSW151, BSW043, BSW046, BSW048, BSW050, BSW051, BSW052, BSW02509, BSW02503, BSW02504, BSW153, BSW02508, BSW02505, BSW02506, BSW02511, BSW053, BSW137, BSW136, BSW140, BSW054, BSW142, BSW143, BSW144, BSW145, BSW146, BSW147, BSW154, BSW139, BSW02510, BSW02512)