Document Title	Specification of LIN State Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	255

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R23-11

Document Change History

Date	**Release**	**Changed by**	**Description**
2023-11-23	R23-11	AUTOSAR Release Management	• Updated Chapter 7.1.8 & 8.3.2 • Removed Chapter 7.1.8.1 Wakeup repetitions for slave • Editorial Changes in Chapter 8.5.1 & 8.5.2 • Corrected Figure in Chapter 9.1 • Added new parameter [ECUC_LinSM_00212] in Chapter 10.3.2
2022-11-24	R22-11	AUTOSAR Release Management	• Changed ID from [SWS_LinSM_00211] to [SWS_LinSM_NA_00211] • Renamed the arguments “Schedule” ([SWS_LinSM_00113] LinSM_ScheduleRequest) and “schedule” ([SWS_LinSM_00129] LinSM_ScheduleRequestConfirmation)
2021-11-25	R21-11	AUTOSAR Release Management	• Corrected Figure 7 and [SWS_LinSM_00233]
2020-11-30	R20-11	AUTOSAR Release Management	• Cleanup error sections in chapter 7
2019-11-28	R19-11	AUTOSAR Release Management	• Editorial changes • Changed Document Status from Final to published

▽

△

2018-10-31	4.4.0	AUTOSAR Release Management	• LIN Slave support (CONC 631) • Replaced references to Lin 2.1 by ISO 17987:2016 • Editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• LINSM E CONFIRMATION TIMEOUT changed to Runtime Error
2016-11-30	4.3.0	AUTOSAR Release Management	• Editorial changes
2015-07-31	4.2.2	AUTOSAR Release Management	• Modified header file structure • Debugging support marked as obsolete • Editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• Removed NULL pointer check requirement and moved to BSW General • Corrections in ECU parameter configuration
2014-03-31	4.1.3	AUTOSAR Release Management	• Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Minor bug fixes • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• LIN wakeup and sleep mode handling corrected
2011-12-22	4.0.3	AUTOSAR Administration	• Added post-build configuration support • Added completion of Production error concept in Com Stack • Removed local network index
2010-09-30	3.1.5	AUTOSAR Administration	• Post-build configuration variant added • Module version check changed according SRS General SRS BSW 00004 • TrcvModeType definition moved from LinIf to LinTrcv

▽

△

2010-02-02	3.1.4	AUTOSAR Administration	• Controlling of I-PDU groups has been moved to the BSW Mode Manager module • Interface to the LIN Transceiver module has been introduced since LIN Transceiver driver is a new module in release 4.0 • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• Initial Release

Disclaimer

Contents

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module LIN State Manager (LinSM). The LinSM together with the LIN Interface, LIN driver, LIN Transceiver driver forms the complete LIN protocol.

The LIN State Manager is designed to be hardware independent.

The LinSM is dependent on upper module Communication Manager [1](ComM) and lower module LIN Interface [2] (LinIf).

It is assumed that the reader is familiar with the ISO 17987 specification. This document will not describe functionality already described in the ISO 17987 specification.

Note that figures in this document are not regarded as requirements. All requirements are described in text prefixed with a requirement tag (e.g. LINSM042). Text not prefixed with a requirement shall be seen as informative text.

1.1 Architectural overview

The Layered Software Architecture [3] positions the LinSM within the BSW architecture as shown below.

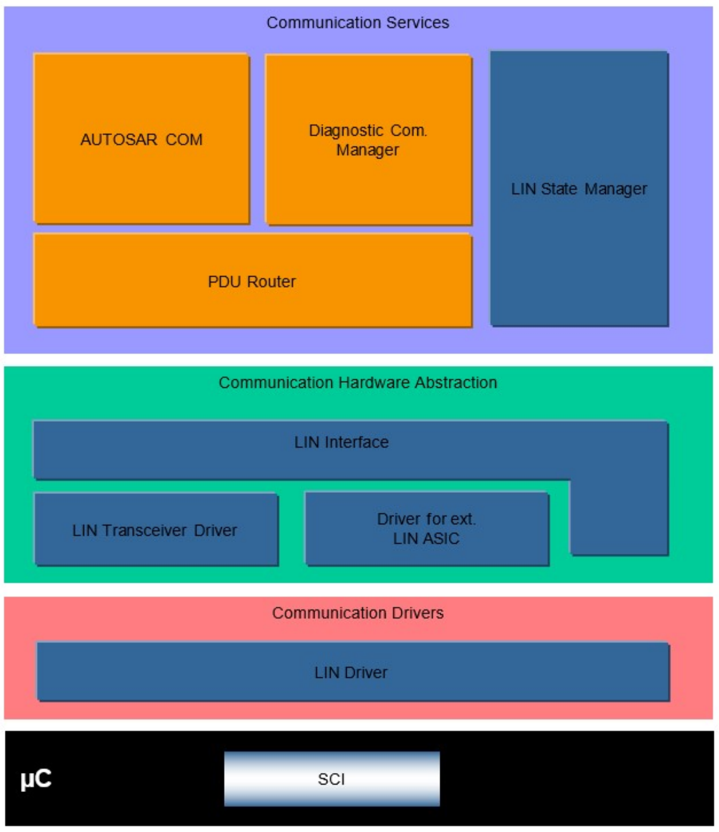

Figure 1.1: AUTOSAR BSW software architecture - LIN stack scope

1.2 Functional overview

The LinSM is responsible for the control flow of the LIN Bus.

This means:

- Switching schedule tables when requested by the upper layer(s) (for LIN master node only).
- Go to sleep and wake up handling, when requested by the upper layer(s) or indicated by the lower layer(s)
- Notification to upper layers when new state is entered.

2 Acronyms and Abbreviations

Acronyms and abbreviations used in this document. Additional abbreviations can be found in the ISO 17987 specification.

Abbreviation / Acronym:	Description:
API	Application Program Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basic Software
BswM	BSW Mode Manager
ComM	Communication Manager
DCM	Diagnostic Communication Manager
DEM	Diagnostic Event Manager
DET	Default Error Tracer
ECU	Electric Control Unit
ID	Identifier
ISR	Interrupt Service Routine
Jitter	Difference between longest delay and shortest delay (e.g. Worst case execution time - Best case execution time)
LIN	Local Interconnect Network
LinIf	LIN Interface
LinSM	LIN State Manager (the subject of this document)
MCAL	Microcontroller Abstraction Layer
PDU	Protocol Data Unit
RAM	Random Access memory
RTE	Run Time Environment
RX	Receive
SPAL	Standard Peripheral Abstraction Layer
SRS	Software Requirement Specification
SW	Software
SWS	Software Design Specification
TP	Transport Protocol
TX	Transmit
UART	Universal Asynchronous Receiver Transmitter
UML	Universal Modelling Language
URL	Uniform Resource Locator
WPII	Work Package in AUTOSAR phase 2
XML	Extensible Markup Language

3 Related documentation

3.1 Input documents & related standards and norms

[1] Specification of Communication Manager
AUTOSAR_CP_SWS_COMManager

[2] Specification of LIN Interface
AUTOSAR_CP_SWS_LINInterface

[3] List of Basic Software Modules
AUTOSAR_CP_TR_BSWModuleList

[4] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[5] Specification of LIN Driver
AUTOSAR_CP_SWS_LINDriver

[6] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer

[7] Specification of Diagnostic Event Manager
AUTOSAR_CP_SWS_DiagnosticEventManager

[8] Specification of Basic Software Mode Manager
AUTOSAR_CP_SWS_BSWModeManager

[9] General Requirements on Basic Software Modules
AUTOSAR_CP_SRS_BSWGeneral

[10] Requirements on LIN
AUTOSAR_CP_SRS_LIN

[11] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

[12] Specification of Standard Types
AUTOSAR_CP_SWS_StandardTypes

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [4, SWS BSW General], which is also valid for Lin State Manager.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Lin State Manager.

4 Constraints and assumptions

4.1 Limitations

There is at most one instance of the LinSM in each ECU. If the underlying LIN Driver [5] supports multiple networks, the LinSM may be LIN master or LIN slave on more than one cluster.

All references to (switching of) schedule tables do only apply to LIN master node; there are no schedule tables for LIN slave node.

4.2 Applicability to car domains

This specification is applicable to all car domains, where LIN is used.

5 Dependencies to other modules

This section describes the relations to other modules within the basic software. It describes the services that are used from these modules. Figure 5.1 shows the modules that are required or optional for the realization of the LinSM module. The figure is complete but is not regarded as requirement.

To be able for the LinSM module to operate the following modules will be interfaced:

[SWS_LinSM_00001] ⌈LIN Interface - LinIf⌋*(SRS_BSW_00384)*

[SWS_LinSM_00085] ⌈Diagnostic Event Manager - DEM⌋*(SRS_BSW_00384)*

[SWS_LinSM_00086] ⌈Default Error Tracer - DET⌋*(SRS_BSW_00384)*

[SWS_LinSM_00105] ⌈Communication Manager - ComM⌋*(SRS_BSW_00384)*

[SWS_LinSM_00196] ⌈BSW Mode Manager - BswM⌋*(SRS_BSW_00384)*

Note that modules that are using the interface (except callbacks) from this module are not listed.

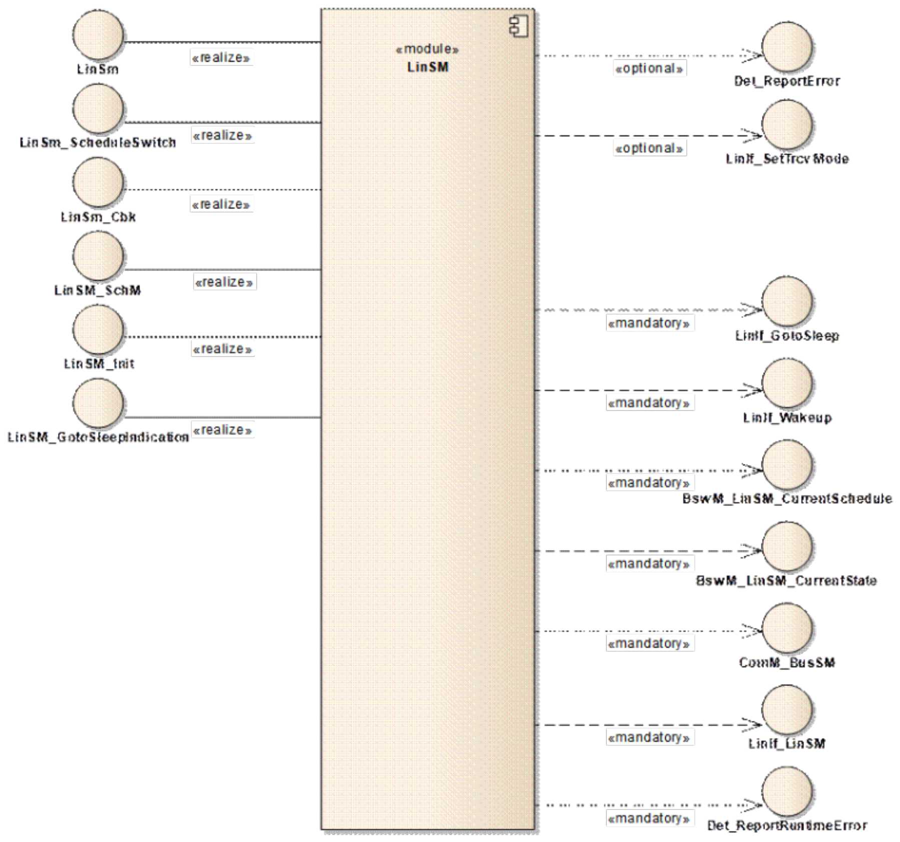

Figure 5.1: Dependencies to other modules

5.1 Relation to Upper layers

In principle, there is no requirement that specific modules shall call the interfaces of the LinSM module. Below, the normal users of LinSM module are listed.

5.1.1 Operating System

The LinSM module does contain access of shared data with above or below modules (using the API). The data that is shared will not need a help of the OS to protect the data for consistency (there are no array accesses, only simple type accesses). However, there may be reentrant functions that access the same data in the LinSM module. It is up to the implementer to solve these accesses.

5.1.2 Module DET (Default Error Tracer)

The Det_ReportError - function of module DET [6] will be called for development and runtime errors.

5.1.3 Module DEM (Diagnostic Event Manager)

Production errors will be reported to the Diagnostic Event Manager [7] module.

5.1.4 ComM

The Com manager module requests the communication via the LIN stack and queries the state of the LinSM module.

5.1.5 BSW Mode Manager

The LinSM module will notify the BSW Mode Manager module [8] when a state is changed. The BSW Mode Manager module will interface the LinSM module when requesting a new schedule table (LIN master node only).

5.2 Relation to Lower layers

Below are the BSW modules that will be interfaced by LinSM module.

5.2.1 LinIf

The LinSM module assumes the following primitives to be provided by the LinIf [2] module:

- Transmission of the goto-sleep command (LIN master node only) and setting the lower layers to sleep mode (LinIf_GotoSleep)
- The wakeup of the Lin bus (LinIf_Wakeup)
- Request to change schedule tables (LinIf_ScheduleRequest). Only applicable to LIN master node.

It is assumed that the LinIf module will call the following callbacks:

- Confirming that the operational mode has been entered, with or without transmission of a wakeup frame (LinSM_WakeupConfirmation)
- Confirming that the sleep mode has been entered, after transmission of a goto-sleep command (LIN master node) or after reception of a goto-sleep command or bus idle detection (LIN slave node) (LinSM_GotoSleepConfirmation)
- Confirming a schedule change (LinSM_ScheduleRequestConfirmation). Only applicable to LIN master node.

[SWS_LinSM_00002] ⌈The LinSM module shall not use or access the LIN driver or assume information about it any way other than what the LinIf module provides through the function calls to the LinIf module listed above.⌋*()*

5.3 File structure

5.3.1 Code file structure

This chapter describes the c-files that implement the LinSM module Configuration. The code file structure is not defined completely within this specification. It is up to each implementer to design the missing structure details.

The pre compile and link time configuration parameters has to be kept in separate files:

5.3.2 Header File structure

This chapter describes the header files that will be included by the LinSM module and possible other modules.

5.3.2.1 LinSM header files

Following header files will exist in a LinSM implementation:

[SWS_LinSM_00005] ⌈A LinSM implementation shall provide a header file LinSM.h that contains all data exported from the LinSM - API declarations (except callbacks), extern types, and global data.⌋*()*

5.3.2.2 Included header files

Following external header files shall be included:

[SWS_LinSM_00013] ⌈The LinSM module shall include the ComM.h file⌋*()*

[SWS_LinSM_00201] ⌈The LinSM module shall include the BswM_LinSM.h⌋*()*

[SWS_LinSM_00305] ⌈The LinSM module shall include the ComM_BusSM.h⌋*()*

[SWS_LinSM_00208] ⌈The LinSM module shall perform a consistency check between code files and header files based on pre-process-checking the version numbers of related code files and header files.⌋*(SRS_BSW_00004)*

6 Requirements Tracing

The following tables reference the requirements specified in [9] and [10] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00004]	All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files	[SWS_LinSM_00208]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_LinSM_00155]
[SRS_BSW_00167]	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	[SWS_LinSM_00073]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_LinSM_00155]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_LinSM_00113] [SWS_LinSM_00122] [SWS_LinSM_00126]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[SWS_LinSM_00156]
[SRS_BSW_00384]	The Basic Software Module specifications shall specify at least in the description which other modules they require	[SWS_LinSM_00001] [SWS_LinSM_00085] [SWS_LinSM_00086] [SWS_LinSM_00105] [SWS_LinSM_00196]
[SRS_BSW_00406]	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	[SWS_LinSM_00116] [SWS_LinSM_00125] [SWS_LinSM_00128] [SWS_LinSM_00131] [SWS_LinSM_00134] [SWS_LinSM_00137]
[SRS_BSW_00407]	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	[SWS_LinSM_00117]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_LinSM_00155]

Table 6.1: RequirementsTracing

7 Functional specification

This chapter specifies the requirements on the module LinSM module. See the Basic Software Modules document [11] for an overview of the responsibilities of the LinSM.

The main responsibilities for the LinSM are:

- Control the communication status (no communication or full communication) of all LIN networks
- Handle schedule change requests (Only applicable to LIN master node)
- Handle communication mode requests
- Notify of state changes to upper layers

The LinSM module will not directly implement functionality in the LIN specification. The LinSM module will support the behavior defined in the ISO 17987 specification. The LIN behavior provided by the LinSM module will allow the reuse of existing LIN nodes conforming to the LIN 1.3, 2.0, 2.1, 2.2 and ISO 17987 specifications.

[SWS_LinSM_00019] ⌈The LinSM module shall be able to handle one or more LIN networks.⌋*()*

Number of LIN networks are restricted by the LinIf specification. All networks are handled via the NetworkHandleType specified by the ComM module.

The identification of the LIN networks is made using reference in the configuration to the ComM network handles.

7.1 States and transitions of the LinSM state machine

The LinSM module will operate in a state-machine. Each network connected will operate in an independent sub-state-machine. Figure 7.1 and Figure 7.2 show a simplified version of the requirements below, the intention of the figures is not to be complete, rather give an overview.

[SWS_LinSM_00020] ⌈The LinSM module shall have one state-machine containing the states LINSM_UNINIT and LINSM_INIT.⌋*()*

[SWS_LinSM_00173] ⌈In the LINSM_INIT there shall be a sub-state-machine for each network with the states LINSM_NO_COM and LINSM_FULL_COM.⌋*()*

[SWS_LinSM_00021] ⌈In LINSM_INIT each network may be in the sub-states LINSM_NO_COM or LINSM_FULL_COM independently.⌋*()*

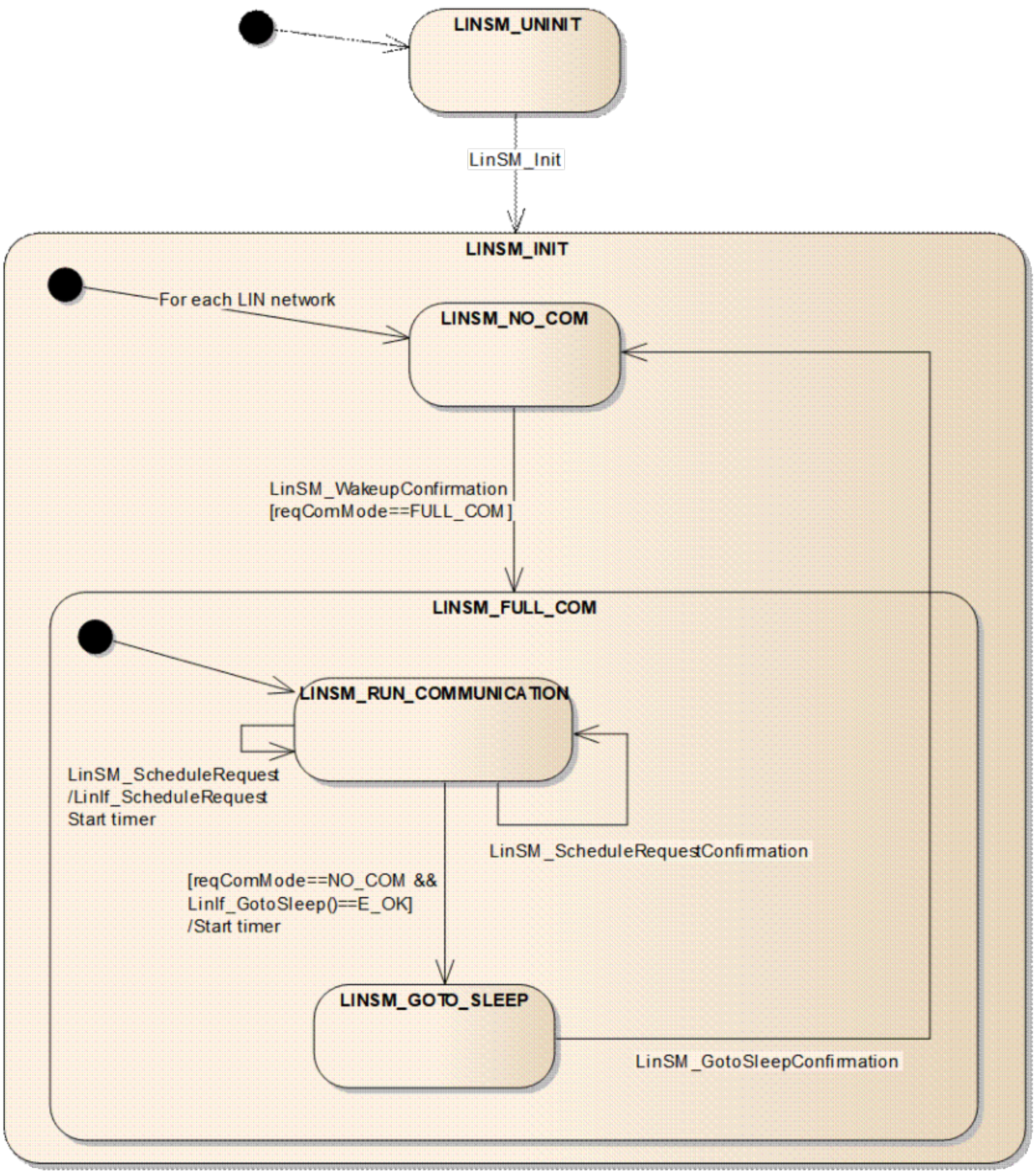

Figure 7.1: LinSM State Machine (master node)

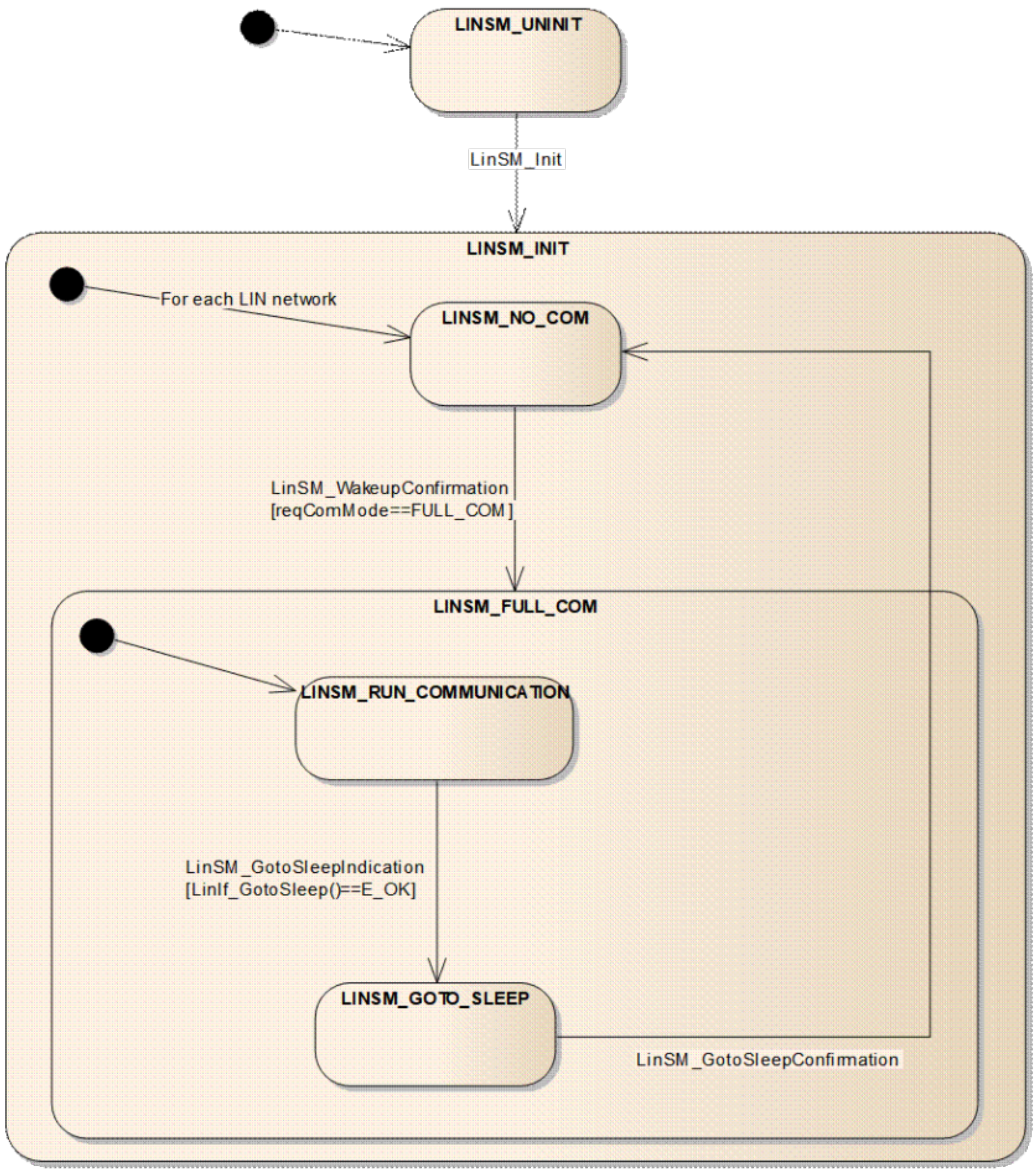

Figure 7.2: LinSM State Machine (slave node)

7.1.1 LINSM_UNINIT

The uninit state is the first state that is active in the LinSM module.

[SWS_LinSM_00022] ⌈There shall be a state called LINSM_UNINIT⌋*()*

[SWS_LinSM_00161] ⌈The state LINSM_UNINIT shall be active at start-up, before any API is called.⌋*()*

7.1.2 LINSM_INIT

After the initialization is made the LinSM module will activate the init state. There are two sub-states in this state. One is the no communication state where no communication is made on the LIN bus, the other is full communication where all communication is made and, for LIN master nodes, schedule tables are active. Each network may be in no communication or full communication independent of each other.

[SWS_LinSM_00024] ⌈There shall be a state called LINSM_INIT⌋*()*

[SWS_LinSM_00025] ⌈The LinSM state-machine shall transit from any state or sub-state to the state LINSM_INIT when LinSM_Init is called.⌋*()*

[SWS_LinSM_00152] ⌈The LinSM state-machine shall transit from any state or sub-state to sub-state LINSM_NO_COM for all networks when LinSM_Init is called.⌋*()*

[SWS_LinSM_00043] ⌈When entering LINSM_INIT the LinSM shall be put in an init state. Init state means that global variables, etc, shall be set to default value (reset value).⌋*()*

[SWS_LinSM_00160] ⌈The sub-state LINSM_NO_COM shall be active when entering the LINSM_INIT state, for all networks when LinSM_Init is called.⌋*()*

[SWS_LinSM_00216] ⌈The LinSM_Init function shall set the schedule type NULL_SCHEDULE for each configured channel. This requirement is only applicable for LIN master node.⌋*()*

To make the LinSM independent from the LinIf module the LinSM module should not call the LinIf module in when the LinSM module is in the init function. The LinSM_Init has therefore additional requirement, see 8.3.1

7.1.3 LINSM_NO_COM

The no communication state is active after initialization and when the ComM module requests no communication (LIN master node) or when the LinIf indicates a bus sleep event (LIN slave node).

[SWS_LinSM_00026] ⌈There shall be a sub-state called LINSM_NO_COM in the state LINSM_INIT.⌋*()*

[SWS_LinSM_00027] ⌈When entering LINSM_NO_COM the LinSM module shall notify (with the exception [SWS_LinSM_00166]) ComM of the state change by calling the ComM_BusSM_ModeIndication with the parameter COMM_NO_COMMUNICATION for the specific network.⌋*()*

[SWS_LinSM_00193] ⌈When entering LINSM_NO_COM the LinSM module shall notify (with the exception [SWS_LinSM_00166]) BswM of the state change by calling the BswM_LinSM_CurrentState with the parameter LINSM_NO_COM for the specific network.⌋*()*

There is one exception to the above two requirements. The rationale is that the ComM may not be initialized when executing the LinSM_Init function.

[SWS_LinSM_00166] ⌈The LinSM module shall not notify the state change to LINSM_NO_COM when the LinSM is executing the LinSM_Init function, i.e. the LinSM_Init function shall neither call ComM_BusSM_ModeIndication nor BswM_LinSM_CurrentState.⌋*()*

[SWS_LinSM_00028] ⌈When LINSM_NO_COM is active, the LinSM module shall not command the LinIf module to communicate for the selected network, i.e. bus shall be silent.

Note: Upon entering or exiting the LINSM_NO_COM state the LinSM module will not set the hardware interface or μ-controller into a new power mode. This is not in the scope of the LinSM⌋*()*

[SWS_LinSM_00203] ⌈When entering LINSM_NO_COM the transceiver shall be set to STANDBY if LinSMTransceiverPassiveMode is true and SLEEP otherwise by using the LinIf_SetTrcvMode. This requirement is applicable only when LinSMTransceiverPassiveMode is configured for the channel.⌋*()*

[SWS_LinSM_00204] ⌈The LinIf_SetTrcvMode shall not be called from the function LinSM_Init.

Note: There is no need to set the mode in the LinSM init function since the Transceiver will set the mode in its init function. The mode is selected in the Transceiver configuration.⌋*()*

7.1.4 LINSM_FULL_COM

The LINSM_FULL_COM is the only state where communication on the LIN bus is allowed. Each network can be in LINSM_FULL_COM independent of each other. All of the following requirements are applicable for each network.

[SWS_LinSM_00032] ⌈There shall be a sub-state called LINSM_FULL_COM for each network in the state LINSM_INIT.⌋*()*

[SWS_LinSM_00033] ⌈When entering LINSM_FULL_COM the ComM shall be notified of the state change by calling the ComM_BusSM_ModeIndication with the parameter COMM_FULL_COMMUNICATION for the specified network.⌋*()*

[SWS_LinSM_00192] ⌈When entering LINSM_FULL_COM the BswM shall be notified of the state change by calling the BswM_LinSM_CurrentState with the parameter LINSM_FULL_COM for the specified network.⌋*()*

[SWS_LinSM_00205] ⌈When entering LINSM_FULL_COM the transceiver shall be set to active by using the LinIf_SetTrcvMode. This requirement is applicable only when Lin SMTransceiverPassiveMode is configured for the channel.⌋*()*

[SWS_LinSM_00301] ⌈When entering LINSM_FULL_COM, the sub-state LINSM_ RUN_COMMUNICATION will be entered.⌋*()*

7.1.5 Goto sleep

The goto-sleep sequence differs between master and slave nodes.

In a master node, when the ComM module requests the no communication mode, the LinSM will request the goto-sleep command to be sent on the LIN bus.

In a slave node, the LIN Interface indicates the bus sleep event to LinSM, either caused by reception of a goto-sleep command or by detection of a bus idle condition. If the ComM module has requested the no communication mode before, the bus sleep event is forwarded to ComM. Otherwise if the full communication mode requested by Com M module is active, the bus sleep event is not forwarded to ComM, but the wakeup process is restarted by LinSM after the goto-sleep sequence is completed.

In all cases, the entering of the no communication mode is notified to BswM and Com M. The callback will always be made, even if there was a problem.

[SWS_LinSM_00035] ⌈The LinSM module may only call LinIf_GotoSleep API in Lin If when the state LINSM_FULL_COM and the sub-state LINSM_RUN_COMMUNICATION is active.⌋*()*

[SWS_LinSM_00046] ⌈When LinSM_GotoSleepConfirmation is called, and the current state/substate is LINSM_FULL_COM/LINSM_GOTOSLEEP, the LinSM shall set the state to LINSM_NO_COM, regardless of the "success" parameter. In any other state, the LinSM_GotoSleepConfirmation shall be ignored.⌋*()*

[SWS_LinSM_00302] ⌈If the LinIf_GotoSleep returns E_OK the LinSM sets the substate LINSM_GOTOSLEEP.⌋*()*

7.1.5.1 Goto sleep specific for master node

This chapter is only applicable for LIN master nodes.

[SWS_LinSM_10208] ⌈If the state is LINSM_FULL_COM, the ComM requests COMM_NO_COMMUNICATION; the LinSM shall call LinIf_GotoSleep to transmit a goto sleep command on the requested network.⌋*()*

[SWS_LinSM_10209] ⌈In all other cases from [SWS_LinSM_10208] the LinIf_Goto Sleep shall not be called.⌋*()*

[SWS_LinSM_00036] ⌈If the ComM module calls LinSM_RequestComMode requesting COMM_NO_COMMUNICATION the LinSM module shall directly call (and not wait for next main function call) the LinIf module function LinIf_GotoSleep on the specified network.⌋*()*

[SWS_LinSM_00177] ⌈If the LinIf_GotoSleep returns E_NOT_OK the LinSM_Request ComMode shall return E_NOT_OK.

If the LinSM module returns LinSM_RequestComMode with E_NOT_OK, the same state shall be set (so that a ComM_BusSM_ModeIndication and BswM_LinSM_Current State are called).⌋*()*

7.1.5.2 Goto sleep specific for slave node

This chapter is only applicable for LIN slave nodes.

[SWS_LinSM_00230] ⌈If the state is LINSM_FULL_COM, the ComM requests COMM_NO_COMMUNICATION; the LinSM shall store the requested communication mode and return E_OK without further action.⌋*()*

[SWS_LinSM_00231] ⌈When LinSM_GotoSleepIndication is called, and the current state is LINSM_FULL_COM, the LinSM shall directly call LinIf_GotoSleep (and not wait for next main function call) to enter sleep mode on the requested network.⌋*()*

[SWS_LinSM_00232] ⌈In all other cases from SWS_LinSM_00231 the LinIf_Goto Sleep shall not be called.⌋*()*

[SWS_LinSM_00233] ⌈When the current state is LINSM_FULL_COM, and the requested communication mode by ComM module is COMM_NO_COMMUNICATION, LinIf shall call LinIf_GotoSleep and afterwards notifiy ComM of the bus sleep event by calling ComM_BusSM_BusSleepMode for the specified network.⌋*()*

[SWS_LinSM_00234] ⌈In the case of [SWS_LinSM_00046] and the requested communication mode by ComM module is COMM_FULL_COMMUNICATION, the LinSM shall restart the wakeup up process.⌋*()*

7.1.6 Changing schedule table (Master only)

This chapter is only applicable for LIN master nodes.

[SWS_LinSM_00079] ⌈If the function LinSM_ScheduleRequest is called, the LinSM module shall forward (and not wait for the next main function call) the request to the LinIf module using the function call LinIf_ScheduleRequest.⌋*()*

[SWS_LinSM_00168] ⌈When the LinSM called LinIf_ScheduleRequest from a call to LinSM_ScheduleRequest, it shall forward the return value to its caller.⌋*()*

[SWS_LinSM_00213] ⌈If LinIf_ScheduleRequest returns with E_NOT_OK the LinSM module shall call BswM_LinSM_CurrentSchedule with the old schedule table in the next main function call.⌋*()*

[SWS_LinSM_00206] ⌈When the LinSM module gets the confirmation of setting a schedule table from the LinIf module the BswM_LinSM_CurrentSchedule shall be called, if not timer has elapsed.⌋*()*

[SWS_LinSM_00214] ⌈If timer has elapsed, the LinSM module shall call BswM_LinSM_CurrentSchedule with unchanged schedule table.

Be aware of that the LinIf will switch to a NULL schedule when entering sleep, then it may make a schedule switch callback.⌋*()*

[SWS_LinSM_00207] ⌈If the LinIf confirms a schedule switch without a preceding call to request new schedule table the BswM_LinSM_CurrentSchedule shall be called⌋*()*

7.1.7 Wake up process

A LIN network will be woken up if ComM module requests a wake up through the LinSM_RequestComMode call or if a LIN node transmits the wakeup signal on the network. The wakeup by cluster is not handled by the LinSM module, it is handled by the EcuM module and will lead to that the ComM requests the network. In both cases the ComM will request full communication to the LinSM module for the specific network.

In case the LinIf is already awake (because of a LIN node waking up the bus) the LinIf will just ignore the wakeup call.

[SWS_LinSM_00047] ⌈If the ComM requests COMM_FULL_COMMUNICATION the LinSM shall call LinIf_Wakeup directly (and not wait for next main function call) to transmit a wake up signal on the requested network, except in the case of SWS_LinSM_00237. Additionally LinSM shall reset the counter for maximum number of retries.⌋*()*

[SWS_LinSM_00178] ⌈In all other cases from [SWS_LinSM_00047] the LinSM module shall not call LinIf_Wakeup.⌋*()*

[SWS_LinSM_00049] ⌈When the LinIf notifies that the WakeUp is successfully sent (success = true), the state shall be set to LINSM_FULL_COM.⌋*()*

[SWS_LinSM_00202] ⌈In all other cases from [SWS_LinSM_00049] the state shall be set same state as previous to the request (so that a mode indication callback is made to BswM and ComM).⌋*()*

[SWS_LinSM_00176] ⌈If the LinIf_Wakeup returns E_NOT_OK the LinSM_Request ComMode shall return E_NOT_OK directly with no further action⌋*()*

7.1.8 Timeout of requests

After calling LinIf_Wakeup the LinSM module is waiting for the LinIf module to confirm the wake up on the bus. There is a possibility that the confirmation is not received, and therefore the LinSM module will wait forever.

The cause for a missing wakeup confirmation could be a problem in software, but also a bus failure or a problem in the master node. A slave node confirms a bus wakeup not after wakeup transmission like a master node, but with reception of the first LIN header from the master node. A LIN slave node shall repeat the wakeup frame transmission up to three times if the communication does not start. After three (failing) wakeup requests the node shall wait a minimum time before restarting the wake up process.

[SWS_LinSM_00175] ⌈There shall be independent wakeup confirmation timers for each network.⌋*()*

[SWS_LinSM_00162] ⌈The (countdown and expiration) handling of wakeup confirmation timers shall be done in the LinSM_MainFunction.⌋*()*

[SWS_LinSM_00159] ⌈The wakeup confirmation timeout parameter LinSMConfirmationTimeout shall have a time that is either 0 or divisible by the LinSM_MainFunction (i.e. LinSM_MainFunction period * m; m integer >0).⌋*()*

[SWS_LinSM_00100] ⌈Before the LinSM calls LinIf_Wakeup on a channel it shall start a wakeup confirmation timer for the according channel with the configured timeout parameter LinSMConfirmationTimeout if LinSMConfirmationTimeout > 0.⌋*()*

[SWS_LinSM_00154] ⌈If LinSM_WakeupConfirmation is called on a channel before the wakeup confirmation timer has expired on the according channel LinSM shall stop the timer.⌋*()*

[SWS_LinSM_00102] ⌈If a wakeup confirmation timer elapses (i.e. no wakeup confirmation occurred) and the maximum number of retries (LinSMModeRequestRepetitionMax) has been reached and LinSMDevErrorDetect (ECUC_LinSM_00206) is set to TRUE, the LinSM module shall report LINSM_E_CONFIRMATION_TIMEOUT to DET.⌋*()*

Making the timeout optional enhances implementation size, if the timeout is not required:

Note: If the configuration parameter LinSMConfirmationTimeout is set to zero the wakeup confirmation timer is not used, and hence a timeout cannot occur. This means that requirement [SWS_LinSM_00102] will not happen.

[SWS_LinSM_00304] ⌈If a wakeup confirmation timer elapses (i.e. no wakeup confirmation occurred), LinSMNodeType is set to MASTER and ComM request is still COMM_FULL_COMMUNICATION then LinIf_Wakeup shall be called.⌋*()*

Note: LinSM stops to retry when confirmation callback LinSM_WakeupConfirmation (SWS_LinSM_00132) occurs or timeout parameter LinSMConfirmationTimeout (ECUC_LinSM_00144) is expired and ComM requests COMM_NO_COMMUNICATION.

[SWS_LinSM_00235] ⌈If a wakeup confirmation timer elapses (i.e. no wakeup confirmation occurred), the maximum number of retries (LinSMModeRequestRepetition Max) has not been reached, LinSMNodeType is set to Slave and ComM request is still COMM_FULL_COMMUNICATION then LinIf_Wakeup shall be called.⌋*()*

[SWS_LinSM_00236] ⌈If a wakeup confirmation timer elapses (i.e. no wakeup confirmation occurred), the maximum number of retries (LinSMModeRequestRepetition Max) has been reached, LinSMNodeType is set to Slave then LinSM shall wait the silence-after-wakeup time (LinSMSilenceAfterWakeupTimeout) and afterwards restart the wakeup process by

- calling LinIf_Wakeup if ComM request is still COMM_FULL_COMMUNICATION.
- resetting the counter for maximum number of retries.

⌋*()*

[SWS_LinSM_00237] ⌈If a wakeup confirmation timer elapses (i.e. no wakeup confirmation occurred), the maximum number of retries (LinSMModeRequestRepetitionMax) has been reached, LinSMNodeType is set to Slave, the silence-after-wakeup time (Lin SMSilenceAfterWakeupTimeout) has not expired and ComM requests COMM_FULL_COMMUNICATION then the call of LinIf_Wakeup shall be delayed to the expiration of the silence-after-wakeup time.⌋*()*

7.2 Handling multiple networks and drivers

Usually only one LIN driver module (supporting multiple networks) is needed in an ECU to handle all LIN networks. However, rarely, some hardware configurations the ECU contain different LIN hardware (e.g. an advanced LIN controller and a UART). In such case, more than one different LIN drivers are required. This will not affect the Lin SM module since the LIN driver only interfaces the LinIf module and not the LIN driver module directly.

The LinSM will only handle networks, and is not concerned to which driver the network maps to, this will be handled by the LinIf.

7.2.1 Multiple networks

Each network has a unique network index (LinSMComMNetworkHandleRef) in the Lin SM configuration.

The configuration parameter LinSMComMNetworkHandleRef is referencing the ComM module configuration directly. This means that no mapping between networks has to be made in the LinSM module when interfacing to the LinIf module. The network index may be used directly to the LinIf module APIs.

[SWS_LinSM_00164] ⌈The LinSM module shall use the same NetworkHandle value, received through an API, when interfacing to the LinIf module (when LIN network is required as a parameter).⌋*()*

7.3 Error Classification

Section "Error Handling" of the document "General Specification of Basic Software Modules" describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.3.1 Development Errors

[SWS_LinSM_00053] Definiton of development errors in module LinSM ⌈

Type of error	*Related error code*	*Error value*
API called without initialization of LinSM	LINSM_E_UNINIT	0x00
Referenced network does not exist (identification is out of range)	LINSM_E_NONEXISTENT_NETWORK	0x20
API service called with wrong parameter	LINSM_E_PARAMETER	0x30
API service called with invalid pointer	LINSM_E_PARAM_POINTER	0x40
Init function failed	LINSM_E_INIT_FAILED	0x50

⌋*()*

7.3.2 Runtime Errors

[SWS_LinSM_00224] Definiton of runtime errors in module LinSM ⌈

Type of error	*Related error code*	*Error value*
Timeout of the callbacks from LinIf	LINSM_E_CONFIRMATION_TIMEOUT	0x00

⌋()

7.3.3 Transient Faults

There are no transient faults.

7.3.4 Production Errors

There are no production errors.

7.3.5 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 Imported types

8.1.1 Standard types

In this chapter all types included from the following modules are listed. The standard AUTOSAR types are defined in the AUTOSAR Specification of Standard Types document [12].

Following types are used by the LinSM module:

[SWS_LinSM_00219] Definition of imported datatypes of module LinSM ⌈

Module	***Header File***	***Imported Type***
ComM	Rte_ComM_Type.h	ComM_ModeType
ComStack_Types	ComStack_Types.h	NetworkHandleType
LinIf	LinIf.h	LinIf_SchHandleType
LinTrcv	Lin_GeneralTypes.h	LinTrcv_TrcvModeType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋*()*

8.2 Type definitions

Following types are defined by the LinSM module:

8.2.1 LinSM_ModeType

[SWS_LinSM_00220] Definition of datatype LinSM_ModeType ⌈

Name	LinSM_ModeType		
Kind	Type		
Derived from	uint8		
Range	LINSM_FULL_COM	0x01	Full communication
	LINSM_NO_COM	0x02	No communication
Description	Type used to report the current mode to the BswM		
Available via	LinSM.h		

⌋*()*

8.2.2 LinSM_ConfigType

[SWS_LinSM_00221] Definition of datatype LinSM_ConfigType ⌈

Name	LinSM_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	–
Description	Data structure type for the post-build configuration parameters.	
Available via	LinSM.h	

⌋*()*

8.3 LinSM API

This is a list of API calls provided for upper layer modules.

8.3.1 LinSM_Init

[SWS_LinSM_00155] Definition of API function LinSM_Init ⌈

Service Name	LinSM_Init	
Syntax	`void LinSM_Init ( const LinSM_ConfigType* ConfigPtr )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Parameters (in)	ConfigPtr	Pointer to the LinSM post-build configuration data.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function initializes the LinSM.	
Available via	LinSM.h	

⌋*(SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00414)*

[SWS_LinSM_00151] ⌈No other LinSM API or other module's (e.g. LinIf) API shall be called from the LinSM_Init function. Other modules may not be initialized.⌋*()*

8.3.2 LinSM_ScheduleRequest

The service LinSM_ScheduleRequest is only applicable for LIN master node.

[SWS_LinSM_00113] Definition of API function LinSM_ScheduleRequest ⌈

Service Name	LinSM_ScheduleRequest	
Syntax	`Std_ReturnType LinSM_ScheduleRequest ( NetworkHandleType network, LinIf_SchHandleType ScheduleTableIdx )`	
Service ID [hex]	0x10	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	network	Identification of the LIN channel
	ScheduleTableIdx	Index of the scheduled table
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK` - Schedule table request has been accepted. `E_NOT_OK` - Not possible to perform the request, e.g. not initialized.
Description	The upper layer requests a schedule table to be changed on one LIN network.	
Available via	LinSM.h	

⌋*(SRS_BSW_00369)*

[SWS_LinSM_00114] ⌈If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value then the error-code LINSM_E_NONEXISTENT_NETWORK shall be reported to the DET module and E_NOT_OK shall be returned.⌋*()*

[SWS_LinSM_00115] ⌈If LinSMDevErrorDetect is enabled: If the schedule parameter has an invalid value, then the error-code LINSM_E_PARAMETER shall be reported to the DET module and E_NOT_OK shall be returned.⌋*()*

[SWS_LinSM_00116] ⌈If LinSMDevErrorDetect is enabled: If the state LINSM_UNINIT is active then the error-code LINSM_E_UNINIT shall be reported to the DET module and E_NOT_OK shall be returned.⌋*(SRS_BSW_00406)*

[SWS_LinSM_10211] ⌈If the function LinSM_ScheduleRequest is called and the state is not LINSM_FULL_COM, the LinSM_ScheduleRequest shall return directly with E_NOT_OK.⌋*()*

[SWS_LinSM_00163] ⌈If the function LinSM_ScheduleRequest is called, LinSMOverwritePendingScheduleRequestProcessing is set to FALSE and another schedule request on the same network is pending, the LinSM_ScheduleRequest shall return directly with E_NOT_OK.⌋*()*

[SWS_LinSM_00165] ⌈If the function LinSM_ScheduleRequest is called, LinSMOverwritePendingScheduleRequestProcessing is set to TRUE and another schedule request on the same network is pending, the pending schedule request shall be overwritten with the new schedule request and return with E_OK.⌋*()*

[SWS_LinSM_00241] ⌈The function LinSM_ScheduleRequest shall be available if the LinSM module is configured as LIN master node on at least one channel.⌋*()*

Note: In a pure LIN slave configuration, this function is not available. This depends on the configuration parameters LinSMNodeType.

8.3.3 LinSM_GetVersionInfo

[SWS_LinSM_00117] Definition of API function LinSM_GetVersionInfo ⌈

Service Name	LinSM_GetVersionInfo	
Syntax	`void LinSM_GetVersionInfo ( Std_VersionInfoType* versioninfo )`	
Service ID [hex]	0x02	
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	–	
Available via	LinSM.h	

⌋*(SRS_BSW_00407)*

[SWS_LinSM_00119] ⌈If LinSMDevErrorDetect is enabled: If the versioninfo pointer parameter is invalid (e.g. NULL), the error-code LINSM_E_PARAM_POINTER shall be reported to the DET module and E_NOT_OK shall be returned.⌋*()*

8.3.4 LinSM_GetCurrentComMode

[SWS_LinSM_00122] Definition of API function LinSM_GetCurrentComMode ⌈

Service Name	LinSM_GetCurrentComMode	
Syntax	`Std_ReturnType LinSM_GetCurrentComMode ( NetworkHandleType network, ComM_ModeType* mode )`	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	network	Identification of the LIN channel
Parameters (inout)	None	
Parameters (out)	mode	Returns the active mode, see ComM_ModeType for descriptions of the modes
Return value	Std_ReturnType	`E_OK` - Ok `E_NOT_OK` - Not possible to perform the request, e.g. not initialized.
Description	Function to query the current communication mode.	
Available via	LinSM.h	

⌋*(SRS_BSW_00369)*

[SWS_LinSM_00123] ⌈If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value, then the error-code LINSM_E_NONEXISTENT_NETWORK shall be reported to the DET module and E_NOT_OK shall be returned.⌋*()*

[SWS_LinSM_00124] ⌈If LinSMDevErrorDetect is enabled: If the mode pointer parameter is invalid (e.g. NULL), then the error-code LINSM_E_PARAM_POINTER shall be reported to the DET module and E_NOT_OK shall be returned.⌋*()*

[SWS_LinSM_00125] ⌈If LinSMDevErrorDetect is enabled: If the state LINSM_UNINIT is active, then the error-code LINSM_E_UNINIT shall be reported to the DET module and E_NOT_OK shall be returned⌋*(SRS_BSW_00406)*

[SWS_LinSM_00180] ⌈If active state is LINSM_NO_COM the state COMM_NO_COMMUNICATION shall be returned.⌋*()*

[SWS_LinSM_00181] ⌈If active state is LINSM_FULL_COM the state COMM_FULL_COMMUNICATION shall be returned.⌋*()*

[SWS_LinSM_00182] ⌈If active state is LINSM_UNINIT the state COMM_NO_COMMUNICATION shall be returned. This is also captured above when the DET is enabled. This is to be defensive.⌋*()*

Note that COMM_SILENT_COMMUNICATION is not used by the LinSM module.

8.3.5 LinSM_RequestComMode

[SWS_LinSM_00126] Definition of API function LinSM_RequestComMode ⌈

Service Name	LinSM_RequestComMode	
Syntax	`Std_ReturnType LinSM_RequestComMode ( NetworkHandleType network, ComM_ModeType mode )`	
Service ID [hex]	0x12	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	network	Identification of the LIN channel
	mode	Request mode
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK` - Request accepted `E_NOT_OK` - Not possible to perform the request, e.g. not initialized.
Description	Requesting of a communication mode. The mode switch will not be made instant. The LinSM will notify the caller when mode transition is made.	
Available via	LinSM.h	

⌋*(SRS_BSW_00369)*

[SWS_LinSM_00127] ⌈If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value, then the error-code LINSM_E_NONEXISTENT_NETWORK shall be reported to the DET module and E_NOT_OK shall be returned.⌋*()*

[SWS_LinSM_00191] ⌈If LinSMDevErrorDetect is enabled: If the mode parameter has an invalid value, then the error-code LINSM_E_PARAMETER shall be reported to the DET module and E_NOT_OK shall be returned.⌋*()*

[SWS_LinSM_00128] ⌈If LinSMDevErrorDetect is enabled: If the state LINSM_UNINIT is active, then the error-code LINSM_E_UNINIT shall be reported to the DET module and E_NOT_OK shall be returned.⌋*(SRS_BSW_00406)*

[SWS_LinSM_00183] ⌈If COMM_SILENT_COMMUNICATION is requested the function shall return E_NOT_OK directly without action⌋*()*

[SWS_LinSM_00223] ⌈LinSM_RequestComMode shall store the requested mode, if the return value is E_OK.

The next activation of the LinSM_MainFunction will then process this request when processing the state machine.

Note, that the state machine definition in section refers to this stored request as req ComMode.⌋*()*

8.4 Scheduled functions

This chapter lists the functions that are called with a fixed period.

8.4.1 LinSM_MainFunction

This function is directly called by the Basic Software Scheduler module. The following function has no return value, no parameter and is non-reentrant.

There is no dependency to other main functions. This main function may be executed without considering other main functions. But scheduling the different main functions intelligent will minimize execution time and jitter.

[SWS_LinSM_00156] Definition of scheduled function LinSM_MainFunction ⌈

Service Name	LinSM_MainFunction

Syntax	`void LinSM_MainFunction (`
	`  void`
	`)`

Service ID [hex]	0x30
Description	Periodic function that runs the timers of different request timeouts
Available via	SchM_LinSm.h

⌋*(SRS_BSW_00373)* Design hint: The function LinSM_MainFunction may be interrupted by other functions. It should be assured that the timers operated by this function are protected so that they behave correctly (e.g. by using critical sections if necessary).

[SWS_LinSM_00157] ⌈The LinSM_MainFunction shall handle the timers that are attached to the functions LinIf_GotoSleep, LinIf_Wakeup or LinIf_ScheduleRequest (see paragraph)⌋*()*

8.5 LinSM callbacks

8.5.1 LinSM_ScheduleRequestConfirmation

The callback LinSM_ScheduleRequestConfirmation is only applicable for LIN master node.

[SWS_LinSM_00129] Definition of callback function LinSM_ScheduleRequest Confirmation ⌈

Service Name	LinSM_ScheduleRequestConfirmation	
Syntax	`void LinSM_ScheduleRequestConfirmation ( NetworkHandleType network, LinIf_SchHandleType ScheduleTableIdx )`	
Service ID [hex]	0x20	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	network	Identification of the LIN channel
	ScheduleTableIdx	Index of the scheduled table
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The LinIf module will call this callback when the new requested schedule table is active.	
Available via	LinSM.h	

⌋*()*

[SWS_LinSM_00130] ⌈If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value, then the error-code LINSM_E_NONEXISTENT_NETWORK shall be reported to the DET module.⌋*()*

[SWS_LinSM_00131] ⌈If LinSMDevErrorDetect is enabled: If the state LINSM_UNINIT is active, then the error-code LINSM_E_UNINIT shall be reported to DET module.⌋*(SRS_BSW_00406)*

[SWS_LinSM_00242] ⌈The callback function LinSM_ScheduleRequestConfirmation shall be available if the LinSM module is configured as LIN master node on at least one channel.⌋*()*

Note: In a pure LIN slave configuration, this function is not available. This depends on the configuration parameters LinSMNodeType.

8.5.2 LinSM_GotoSleepIndication

The callback LinSM_GotoSleepIndication is only applicable for LIN slave node.

[SWS_LinSM_91000] Definition of callback function LinSM_GotoSleepIndication
⌈

Service Name	LinSM_GotoSleepIndication	

Syntax	`void LinSM_GotoSleepIndication (`
	`  NetworkHandleType Channel`
	`)`	

Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different Channels	
Parameters (in)	Channel	Identification of the LIN channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The LinIf will call this callback when the go to sleep command is received on the network or a bus idle timeout occurs. Only applicable for LIN slave nodes.	
Available via	LinSM.h	

⌋*()*

[SWS_LinSM_00239] ⌈If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value, then the error-code LINSM_E_NONEXISTENT_NETWORK shall be reported to the DET module.⌋*()*

[SWS_LinSM_00240] ⌈If LinSMDevErrorDetect is enabled: If the state LINSM_UNINIT is active, then the error-code LINSM_E_UNINIT shall be reported to DET module.⌋*()*

[SWS_LinSM_00243] ⌈The callback function LinSM_GotoSleepIndication shall be available if the LinSM module is configured as LIN slave node on at least one channel.⌋*()*

Note: In a pure LIN master configuration, this function is not available. This depends on the configuration parameters LinSMNodeType.

8.5.3 LinSM_GotoSleepConfirmation

[SWS_LinSM_00135] Definition of callback function LinSM_GotoSleepConfirmation ⌈

Service Name	LinSM_GotoSleepConfirmation	
Syntax	`void LinSM_GotoSleepConfirmation ( NetworkHandleType network, boolean success )`	
Service ID [hex]	0x22	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	network	Identification of the LIN channel
	success	True if goto sleep was successfully sent, false otherwise
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The LinIf will call this callback when the go to sleep command is sent successfully or not sent successfully on the network.	
Available via	LinSM.h	

⌋*()*

[SWS_LinSM_00136] ⌈If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value, then the error-code LINSM_E_NONEXISTENT_NETWORK shall be reported to the DET module.⌋*()*

[SWS_LinSM_00137] ⌈If LinSMDevErrorDetect is enabled: If the state LINSM_UNINIT is active, then the error-code LINSM_E_UNINIT shall be reported to the DET module.⌋*(SRS_BSW_00406)*

8.5.4 LinSM_WakeupConfirmation

[SWS_LinSM_00132] Definition of callback function LinSM_WakeupConfirmation ⌈

Service Name	LinSM_WakeupConfirmation	
Syntax	`void LinSM_WakeupConfirmation ( NetworkHandleType network, boolean success )`	
Service ID [hex]	0x21	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	network	Identification of the LIN channel
	success	True if wakeup was successfully sent, false otherwise
Parameters (inout)	None	

▽

△

Parameters (out)	None
Return value	None
Description	The LinIf will call this callback when the wake up signal command is sent not successfully/ successfully on the network.
Available via	LinSM.h

⌋*()*

[SWS_LinSM_00133] ⌈If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value, then the error-code LINSM_E_NONEXISTENT_NETWORK shall be reported to the DET module.⌋*()*

[SWS_LinSM_00134] ⌈If LinSMDevErrorDetect is enabled: If the state LINSM_UNINIT is active, then the error-code LINSM_E_UNINIT shall be reported to the DET module.⌋ *(SRS_BSW_00406)*

8.6 Mandatory Interfaces

This chapter defines all interfaces that are required to fulfill the core functionality.

[SWS_LinSM_00229] Definition of mandatory interfaces in module LinSM ⌈

API Function	***Header File***	***Description***
BswM_LinSM_CurrentSchedule	BswM_LinSM.h	Function called by LinSM to indicate the currently active schedule table for a specific LIN channel.
BswM_LinSM_CurrentState	BswM_LinSM.h	Function called by LinSM to indicate its current state.
ComM_BusSM_BusSleepMode	ComM.h	Notification of the corresponding Bus State Manager that the actual bus mode is Bus-Sleep. Only applicable for ComM channels with ComMNm Variant set to SLAVE_ACTIVE or SLAVE_PASSIVE. E.g. LIN slaves (ComMNMVariant = SLAVE_ ACTIVE) or Ethernet channels with OA TC10 compliant Ethernet hardware which act as passive communication slave (ComMNMVariant = SLAVE_ PASSIVE and EthTrcvActAsSlavePassiveEnabled set to TRUE)
ComM_BusSM_ModeIndication	ComM.h	Indication of the actual bus mode by the corresponding Bus State Manager. ComM shall propagate the indicated state to the users with means of the RTE and BswM.
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
LinIf_GotoSleep	LinIf.h	Initiates a transition into the Sleep Mode on the selected channel.
LinIf_ScheduleRequest	LinIf.h	Requests a schedule table to be executed. Only used for LIN master nodes.
LinIf_Wakeup	LinIf.h	Initiates the wake up process.

⌋*()*

8.7 Optional Interfaces

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

[SWS_LinSM_00138] Definition of optional interfaces in module LinSM ⌈

API Function	***Header File***	***Description***
Det_ReportError	Det.h	Service to report development errors.
LinIf_SetTrcvMode	LinIf.h	Set the given LIN transceiver to the given mode.

⌋*()*

8.8 Configurable Interfaces

No configurable interfaces.

9 Sequence diagrams

This chapter will show use-cases for LIN communication and API usage. As the communication is in real-time it is not easy to show the real-time behavior in the UML dynamic diagrams. It is advisable to read the corresponding descriptive text to each UML diagram.

To show the behavior of the modules in the different use-cases, there are local function calls made to show what is done and when to get information. It is not mandatory to use these local functions; they are here just to make the use-cases more understandable.

Note that all parameters and return types are left out to make the diagrams easier to read and understand. If needed for clarification the parameter value or return value are shown.

9.1 Goto-sleep process

9.1.1 Master

This chapter is only applicable for LIN master nodes.

This use-case shows the transition into the LINSM_NO_COM state when the goto-sleep command has been sent successfully on the bus.

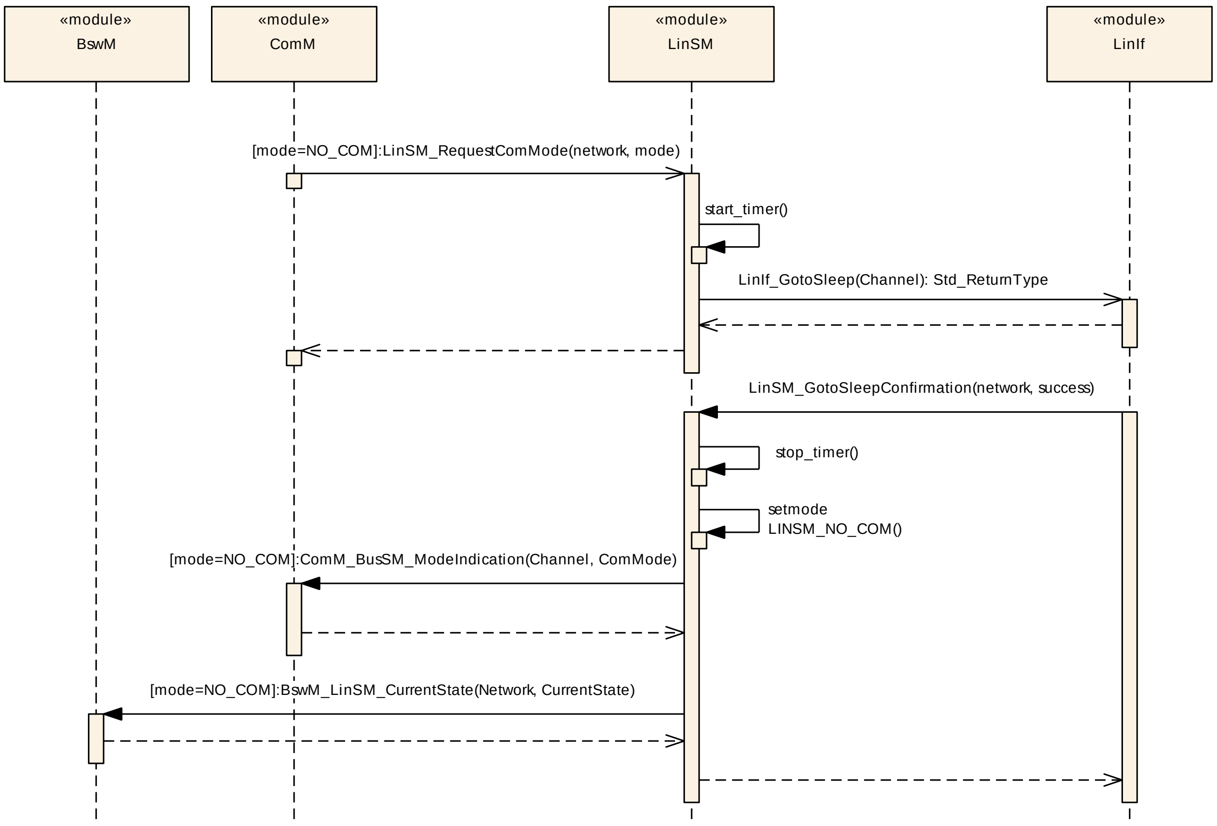
«module» BswM
«module» ComM
«module» LinSM
«module» LinIf
[mode=NO_COM]:LinSM_RequestComMode(network, mode)
start_timer()
LinIf_GotoSleep(Channel): Std_ReturnType
LinSM_GotoSleepConfirmation(network, success)
stop_timer()
setmode LINSM_NO_COM()
[mode=NO_COM]:ComM_BusSM_ModeIndication(Channel, ComMode)
[mode=NO_COM]:BswM_LinSM_CurrentState(Network, CurrentState)

Figure 9.1: Goto-sleep command process (Master)

9.1.2 Slave

This chapter is only applicable for LIN slave nodes.

This use-case shows the transition into the LINSM_NO_COM state when the goto-sleep command has been received on the bus when no communication is requested.

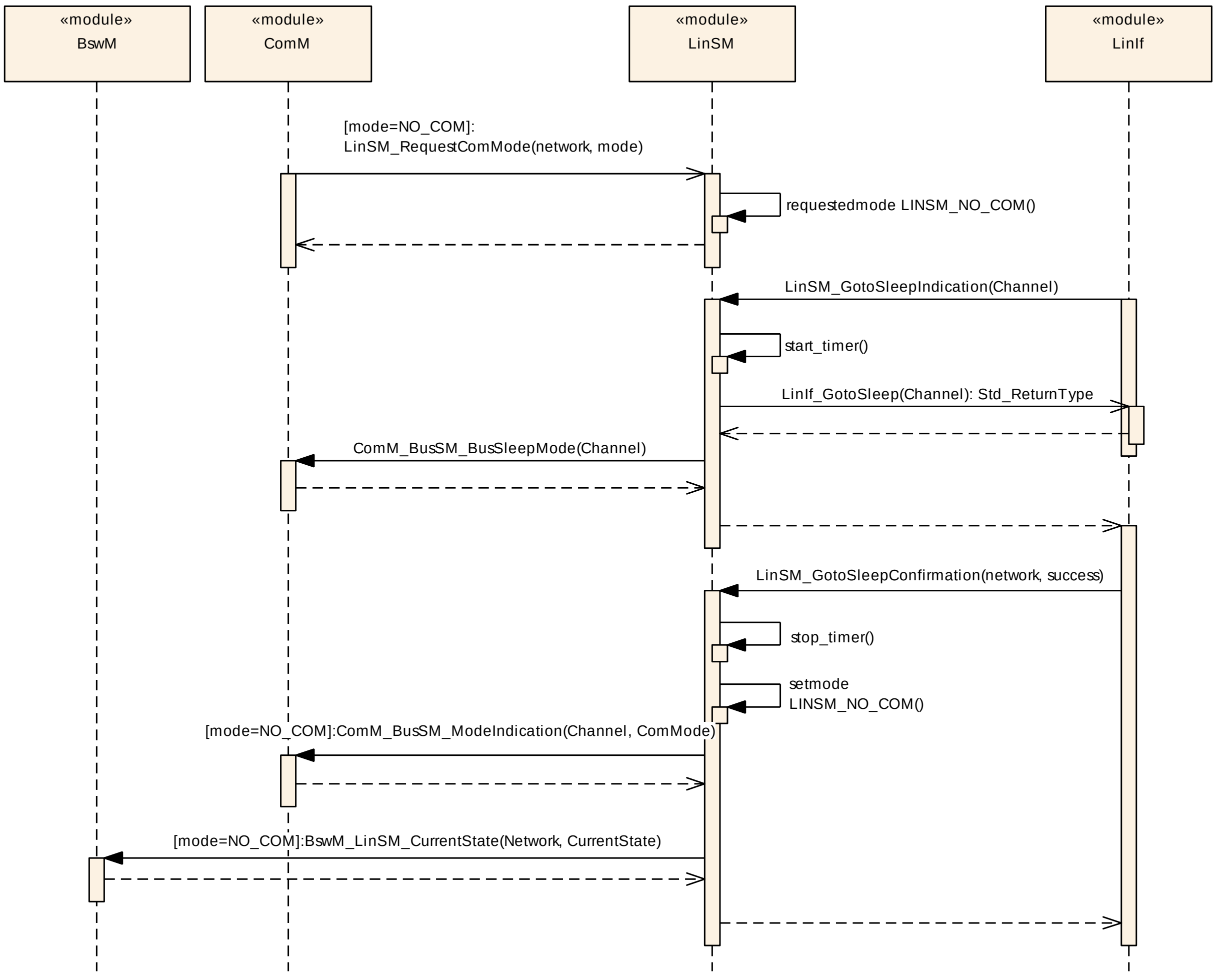
«module» BswM
«module» ComM
«module» LinSM
«module» LinIf
[mode=NO_COM]: LinSM_RequestComMode(network, mode)
requestedmode LINSM_NO_COM()
LinSM_GotoSleepIndication(Channel)
start_timer()
LinIf_GotoSleep(Channel): Std_ReturnType
ComM_BusSM_BusSleepMode(Channel)
LinSM_GotoSleepConfirmation(network, success)
stop_timer()
setmode LINSM_NO_COM()
[mode=NO_COM]:ComM_BusSM_ModeIndication(Channel, ComMode)
[mode=NO_COM]:BswM_LinSM_CurrentState(Network, CurrentState)

Figure 9.2: - Goto-sleep-command process (Slave)

9.2 Internal wake up

The Figure 9.3 shows the internal wakeup. A wakeup is requested from module above the LinSM.

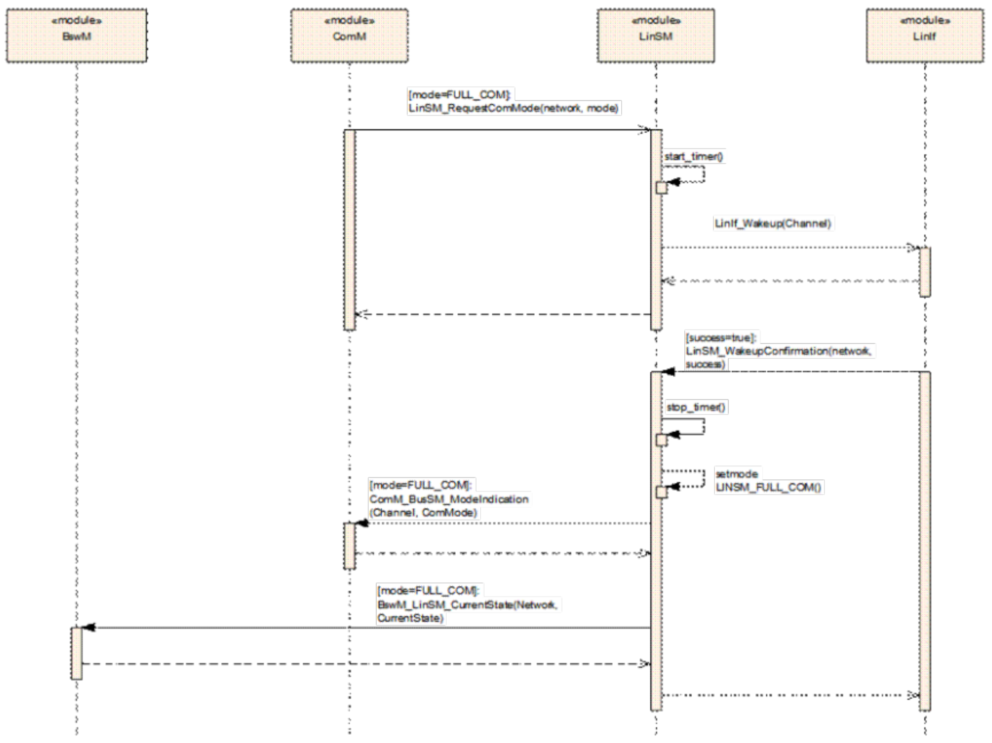

Figure 9.3: - Internal wake up

9.3 Schedule switch (Master only)

This chapter is only applicable for LIN master nodes.

Figure 9.4 shows the use-cases of switching the schedule table when the LinSM accepts the request.

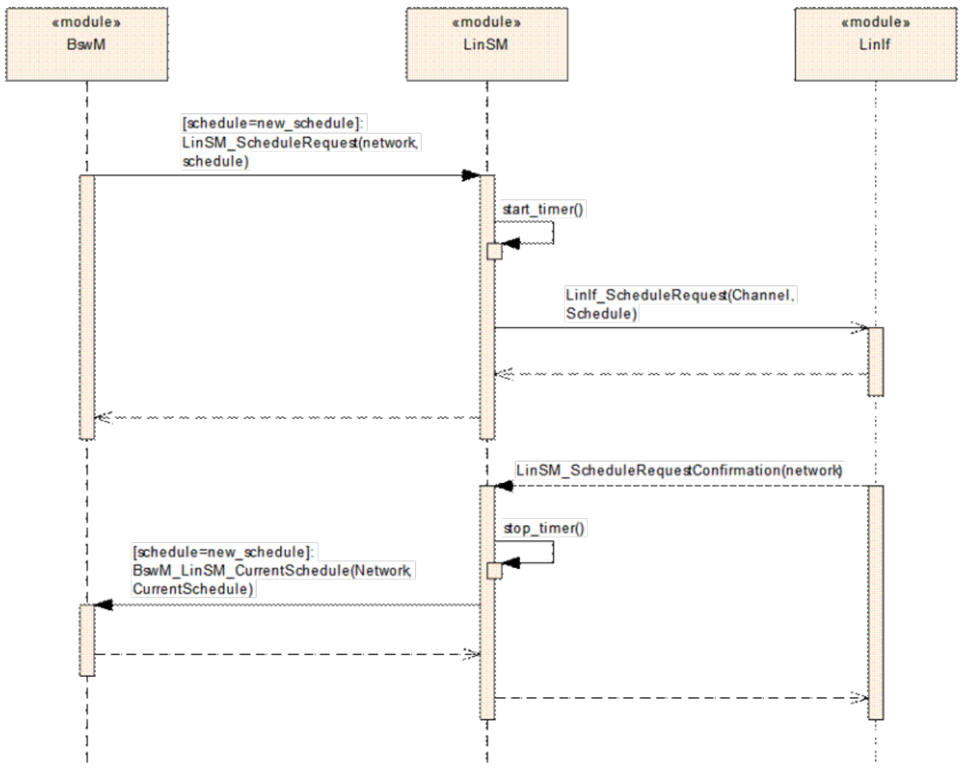

Figure 9.4: Schedule Table switch

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers.

The chapter 10.3 specifies the structure (containers) and the parameters of the LinSM module.

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

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

Example: The LinSM_Configuration is placed in a specific Flash sector. This flash sector may be reflashed after the ECU is placed in the vehicle.

10.2.1 Configuration Tool

A configuration tool will create a configuration structure that is understood by the Lin SM.

[SWS_LinSM_00073] ⌈The LinSM module shall not make any consistency check of the configuration in run-time in production software. It may, however, be done if the Development Error Detection is enabled.⌋*(SRS_BSW_00167)*

10.3 LinSM_Configuration

The paragraph defines the LinSM configuration.

10.3.1 LinSM

SWS Item	[ECUC_LinSM_00209]
Module Name	LinSM
Description	Configuration of the Lin State Manager module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
LinSMConfigSet	1	This container contains the configuration parameters and sub containers of the AUTOSAR LinSm module.
LinSMGeneral	1	This container contains general parameters of LIN State Manager module.

10.3.2 LinSMConfigSet

SWS Item	[ECUC_LinSM_00207]
Container Name	LinSMConfigSet
Parent Container	LinSM
Description	This container contains the configuration parameters and sub containers of the AUTOSAR LinSm module.
Configuration Parameters	

SWS Item	[ECUC_LinSM_00208]		
Parameter Name	LinSMModeRequestRepetitionMax		
Parent Container	LinSMConfigSet		
Description	Specifies the maximum amount of wakeup repetitions without a respective wakeup confirmation from the LinIf module until the LinSM module reports a runtime Error to DET. In case the channel is configured as slave it also specifies after how many retries the silence-after-wakeup is started.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_LinSM_00212]
Parameter Name	LinSMOverwritePendingScheduleRequest
Parent Container	LinSMConfigSet

▽

△

Description	Select the handling of a pending schedule requests processing, if a new schedule request is requested. • FALSE: Reject a new schedule request, if a schedule request processing is pending. • TRUE: Overwrite a pending request processing with the new schedule request.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
LinSMChannel	1..*	Describes each LIN channel the LinSM is connected to.

10.3.3 LinSMChannel

SWS Item	**[ECUC_LinSM_00142]**
Container Name	LinSMChannel
Parent Container	LinSMConfigSet
Description	Describes each LIN channel the LinSM is connected to.
Configuration Parameters	

SWS Item	**[ECUC_LinSM_00144]**		
Parameter Name	LinSMConfirmationTimeout		
Parent Container	LinSMChannel		
Description	Wakeup Confirmation Timeout in seconds. The timeout must be longer than a wakeup command on the bus (i.e. it is bit rate dependent).		
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, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_LinSM_00211]**		
Parameter Name	LinSMNodeType		
Parent Container	LinSMChannel		
Description	Specifies the LIN node type of this channel.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	MASTER	Master node	
	SLAVE	Slave node	
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_LinSM_00210]**		
Parameter Name	LinSMSilenceAfterWakeupTimeout		
Parent Container	LinSMChannel		
Description	Timeout in seconds after a failed wakeup sequence until a new wakeup process is started.		
Multiplicity	0..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, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local dependency: This parameter is only applicable for LIN slave nodes, depending on parameter LinSMNodeType.		

SWS Item	**[ECUC_LinSM_00202]**		
Parameter Name	LinSMTransceiverPassiveMode		
Parent Container	LinSMChannel		
Description	Selects STANDBY (true) or SLEEP (false) transceiver mode when entering LINSM_NO_COM.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD

▽

△

	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_LinSM_00145]**		
Parameter Name	LinSMComMNetworkHandleRef		
Parent Container	LinSMChannel		
Description	Unique handle to identify one certain LIN network. Reference to one of the network handles configured in the ComM.		
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, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
LinSMSchedule	0..*	The schedule references to a schedule that is located in the LinIf configuration. Moreover, the PDU groups are located in the COM configuration. Note that there are two references to PDU groups. The simple reason is that a PDU group is only allowed to contain one direction (TX or RX). Only applicable to LIN master nodes.

10.3.4 LinSMGeneral

SWS Item	**[ECUC_LinSM_00139]**
Container Name	LinSMGeneral
Parent Container	LinSM
Description	This container contains general parameters of LIN State Manager module.
Configuration Parameters	

SWS Item	**[ECUC_LinSM_00206]**		
Parameter Name	LinSMDevErrorDetect		
Parent Container	LinSMGeneral		
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_LinSM_00141]**		
Parameter Name	LinSMMainProcessingPeriod		
Parent Container	LinSMGeneral		
Description	Fixed period that the MainFunction shall be called.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
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_LinSM_00140]**		
Parameter Name	LinSMVersionInfoApi		
Parent Container	LinSMGeneral		
Description	Switches the LinSM_GetVersionInfo function ON or OFF.		
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		

No Included Containers

10.3.5 LinSMSchedule

SWS Item	**[ECUC_LinSM_00146]**
Container Name	LinSMSchedule
Parent Container	LinSMChannel
Description	The schedule references to a schedule that is located in the LinIf configuration. Moreover, the PDU groups are located in the COM configuration. Note that there are two references to PDU groups. The simple reason is that a PDU group is only allowed to contain one direction (TX or RX). Only applicable to LIN master nodes.
Configuration Parameters	

SWS Item	**[ECUC_LinSM_00001]**
Parameter Name	LinSMScheduleIndex
Parent Container	LinSMSchedule

▽

△

Description	This index parameter can be used by the BswM as a SymbolicNameReference target. The LinSM just forwards the request from the BswM to LinIf. Note that the value of the LinSMScheduleIndex shall be the same as the value from the LinIf.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
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 withAuto = true		

SWS Item	**[ECUC_LinSM_00149]**		
Parameter Name	LinSMScheduleIndexRef		
Parent Container	LinSMSchedule		
Description	Reference to a schedule table in the LinIf configuration		
Multiplicity	1		
Type	Symbolic name reference to LinIfScheduleTable		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.4 Published Information

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

A Not applicable requirements

[SWS_LinSM_NA_00211] ⌈These requirements are not applicable to this specification.⌋*(SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00170, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00437, SRS_BSW_00168, SRS_BSW_00425, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00422, SRS_BSW_00417, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00415, SRS_BSW_00343, SRS_BSW_00439, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00331, SRS_BSW_00010, SRS_BSW_00333, SRS_BSW_00321, SRS_BSW_00341, SRS_BSW_00334, SRS_Lin_01590, SRS_Lin_01560, SRS_Lin_01577, SRS_BSW_00438)*

A.1 Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

A.2 Traceable item history of this document according to AUTOSAR Release R22-11

A.2.1 Added Specification Items in R22-11

[SWS_LinSM_00001] [SWS_LinSM_00002] [SWS_LinSM_00005] [SWS_LinSM_00013] [SWS_LinSM_00019] [SWS_LinSM_00020] [SWS_LinSM_00021] [SWS_LinSM_00022] [SWS_LinSM_00024] [SWS_LinSM_00025] [SWS_LinSM_00026] [SWS_LinSM_00027] [SWS_LinSM_00028] [SWS_LinSM_00032] [SWS_LinSM_00033] [SWS_LinSM_00035] [SWS_LinSM_00036] [SWS_LinSM_00043] [SWS_LinSM_00046] [SWS_LinSM_00047] [SWS_LinSM_00049] [SWS_LinSM_00053] [SWS_LinSM_00073] [SWS_LinSM_00079] [SWS_LinSM_00085] [SWS_LinSM_00086] [SWS_LinSM_00100] [SWS_LinSM_00101] [SWS_LinSM_00102] [SWS_LinSM_00103] [SWS_LinSM_00105] [SWS_LinSM_00113] [SWS_LinSM_00114] [SWS_LinSM_00115] [SWS_LinSM_00116] [SWS_LinSM_00117] [SWS_LinSM_00119] [SWS_LinSM_00122] [SWS_LinSM_00123] [SWS_LinSM_00124] [SWS_LinSM_00125] [SWS_LinSM_00126] [SWS_LinSM_00127] [SWS_LinSM_00128] [SWS_LinSM_00129] [SWS_LinSM_00130] [SWS_LinSM_00131] [SWS_LinSM_00132] [SWS_LinSM_00133] [SWS_LinSM_00134] [SWS_LinSM_00135] [SWS_LinSM_00136] [SWS_LinSM_00137] [SWS_LinSM_00138] [SWS_LinSM_00151] [SWS_LinSM_00152] [SWS_LinSM_00154] [SWS_LinSM_00155] [SWS_LinSM_00156] [SWS_LinSM_00157] [SWS_LinSM_00159] [SWS_LinSM_00160] [SWS_LinSM_00161] [SWS_LinSM_00162] [SWS_LinSM_00163] [SWS_LinSM_00164] [SWS_LinSM_00166] [SWS_LinSM_00168] [SWS_LinSM_00170] [SWS_LinSM_00172] [SWS_LinSM_00173] [SWS_LinSM_00175] [SWS_LinSM_00176] [SWS_-

LinSM_00177] [SWS_LinSM_00178] [SWS_LinSM_00180] [SWS_LinSM_00181] [SWS_LinSM_00182] [SWS_LinSM_00183] [SWS_LinSM_00191] [SWS_LinSM_00192] [SWS_LinSM_00193] [SWS_LinSM_00196] [SWS_LinSM_00201] [SWS_LinSM_00202] [SWS_LinSM_00203] [SWS_LinSM_00204] [SWS_LinSM_00205] [SWS_LinSM_00206] [SWS_LinSM_00207] [SWS_LinSM_00208] [SWS_LinSM_00213] [SWS_LinSM_00214] [SWS_LinSM_00215] [SWS_LinSM_00216] [SWS_LinSM_00219] [SWS_LinSM_00220] [SWS_LinSM_00221] [SWS_LinSM_00223] [SWS_LinSM_00224] [SWS_LinSM_00229] [SWS_LinSM_00230] [SWS_LinSM_00231] [SWS_LinSM_00232] [SWS_LinSM_00233] [SWS_LinSM_00234] [SWS_LinSM_00235] [SWS_LinSM_00236] [SWS_LinSM_00237] [SWS_LinSM_00239] [SWS_LinSM_00240] [SWS_LinSM_00241] [SWS_LinSM_00242] [SWS_LinSM_00243] [SWS_LinSM_00301] [SWS_LinSM_00302] [SWS_LinSM_00304] [SWS_LinSM_00305] [SWS_LinSM_00307] [SWS_LinSM_10208] [SWS_LinSM_10209] [SWS_LinSM_10211] [SWS_LinSM_91000] [SWS_LinSM_NA_00211]

A.2.2 Changed Specification Items in R22-11

none

A.2.3 Deleted Specification Items in R22-11

none

A.3 Traceable item history of this document according to AUTOSAR Release R23-11

A.3.1 Added Specification Items in R23-11

[SWS_LinSM_00165]

A.3.2 Changed Specification Items in R23-11

[SWS_LinSM_00047] [SWS_LinSM_00100] [SWS_LinSM_00102] [SWS_LinSM_00113] [SWS_LinSM_00122] [SWS_LinSM_00126] [SWS_LinSM_00138] [SWS_LinSM_00154] [SWS_LinSM_00159] [SWS_LinSM_00162] [SWS_LinSM_00163] [SWS_LinSM_00175] [SWS_LinSM_00219] [SWS_LinSM_00220] [SWS_LinSM_00221] [SWS_LinSM_00229] [SWS_LinSM_00235] [SWS_LinSM_00236] [SWS_LinSM_00237] [SWS_LinSM_00241] [SWS_LinSM_00242] [SWS_LinSM_00243] [SWS_LinSM_00304]

A.3.3 Deleted Specification Items in R23-11

[SWS_LinSM_00101] [SWS_LinSM_00103] [SWS_LinSM_00170] [SWS_LinSM_-00172] [SWS_LinSM_00215] [SWS_LinSM_00307]