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

Document Version	1.3.1
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
28.02.2014	1.3.1	AUTOSAR Release Management	• Editorial Changes
17.05.2012	1.3.0	AUTOSAR Administration	• Changed handling of communication requests of ComM
27.04.2011	1.2.0	AUTOSAR Administration	• Changes concerning backporting of BswM from R4 to R3.2 • Partial Networking: BswM handles activation of I-PDU groups • API extension to identify the current LinSM SubMode • Legal disclaimer revised
10.09.2010	1.1.0	AUTOSAR Administration	• Chapter 10 updated to have configurable “initialize” in call to Com_IpduGroupStart • Legal disclaimer revised
23.06.2008	1.0.1	AUTOSAR Administration	Legal disclaimer revised
03.12.2007	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

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 LIN State Manager is designed to be hardware independent.

The LinSM is dependent on above module Communication Manager [13] (ComM) and below module LIN Interface [7] (LinIf).

It is assumed that the reader is familiar with the LIN 2.0 specification [16]. This document will not describe functionality already described in the LIN 2.0 specification [16].

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).

1.1 Architectural overview

The Layered Software Architecture [1] positions the LinSM within the BSW architecture as shown below.

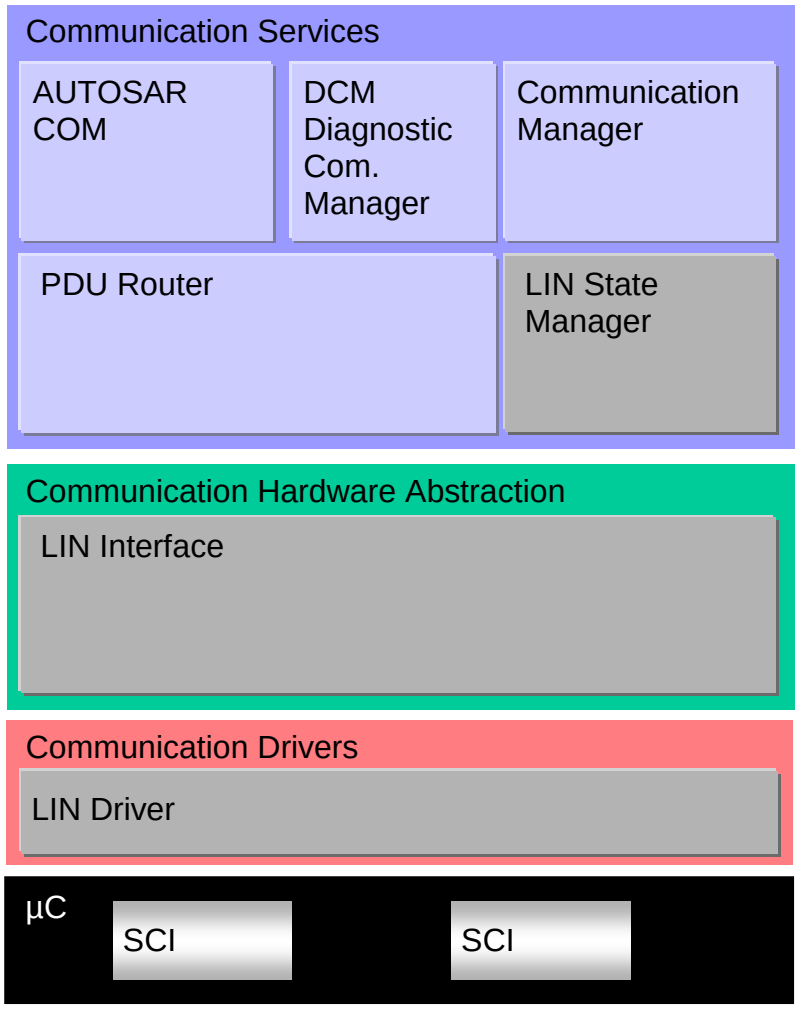

Figure 1 – AUTOSAR BSW software architecture - LIN stack scope

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).
Go to sleep and wake up handling, when requested by the upper layer(s)
Notify BswM when switching schedule table and if the current mode changes.

2 Acronyms and abbreviations

Acronyms and abbreviations used in this document. Additional abbreviations can be found in the LIN 2.0 specification [16].

Abbreviation / Acronym:	Description:
API	Application Program Interface
BSW	Basic Software
BswM	BSW Mode Manager
Com	Communication module
ComM	Communication Manager
DCM	Diagnostic Communication Manager
DEM	Diagnostic Event Manager
DET	Development Error Tracer
LIN	Local Interconnect Network
LinIf	LIN Interface
LinSM	LIN State Manager (the subject of this document)
PDU	Protocol Data Unit

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture,
AUTOSAR_LayeredSoftwareArchitecture.pdf

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

[4] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[5] Specification of Development Error Tracer
AUTOSAR_SWS_DET.pdf

[6] Requirements on LIN
AUTOSAR_SRS_LIN.pdf

[7] Specification of LIN Interface
AUTOSAR_SWS_LIN_Interface.pdf

[8] Specification of Diagnostics Event Manager
AUTOSAR_SWS_DEM.pdf

[9] Specification of ECU Configuration
AUTOSAR_ECU_Configuration.pdf

[10] AUTOSAR Requirements on Mode Management
AUTOSAR_SRS_ModeManagement.doc

[11] AUTOSAR Specification of Communication
AUTOSAR_SWS_COM.pdf

[12] AUTOSAR Specification of LIN Driver
AUTOSAR_SWS_LIN_Driver.pdf

[13] AUTOSAR Specification of Communication Manager
AUTOSAR_SWS_ComManager.pdf

[14] AUTOSAR Basic Software Module Description Template,
AUTOSAR_BSW_Module_Description.pdf

[15] AUTOSAR Basic Specification of Basic Software Mode Manager,
AUTOSAR_BSW_Module_Description.pdf

3.2 Related standards and norms

[16] LIN Specification Package Revision 2.0, September 23, 2003 http://www.lin-subbus.org/

4 Constraints and assumptions

4.1 Limitations

The LinSM module can only be used as a LIN master in a LIN cluster. There at most one instance of the LinSM in each ECU. If the underlying LIN Driver [12] supports multiple channels, the LinSM may be master on more than one cluster.

4.2 Applicability to car domains

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

4.3 Planned updates

The AUTOSAR LIN modules are currently using the term “channel” do describe a connection to the LIN bus. The correct term is “controller”. This will be changed in AUTOSAR release 4.0.

Currently no complete error concept is implemented in the AUTOSAR LIN modules. The LIN modules will detect errors on the bus but cannot detect failing/missing slave nodes. This will be introduced in AUTOSAR release 4.0.

Bug 5414: AUTOSAR release 4.0 will introduce Vehicle Mode Manager (VMM) and Application Mode Manager modules. One of these modules will call the LinSM_ScheduleRequest. Currently there is no user of this function, it must probably be called from the application.

Bug 18460: The task of the LinSM is to control the modes of the LIN transceiver. In AUTOSAR release 3.0 there is no LIN transceiver. The functionality in LinSM to control the LIN transceiver and the LIN transceiver itself will be introduced in AUTOSAR 4.0.

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 2 shows the modules that are required or optional for the realization of the LinSM module.

To be able for the LinSM to operate the following modules will be interfaced:
LINSM001: LIN Interface – LinIf
LINSM085 : Diagnostic Event Manager – DEM
LINSM086 : Development Error Tracer (if enabled) – DET
LINSM105: Communication Manager – ComM
LINSM156: Basic Software Mode Manager - BswM

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

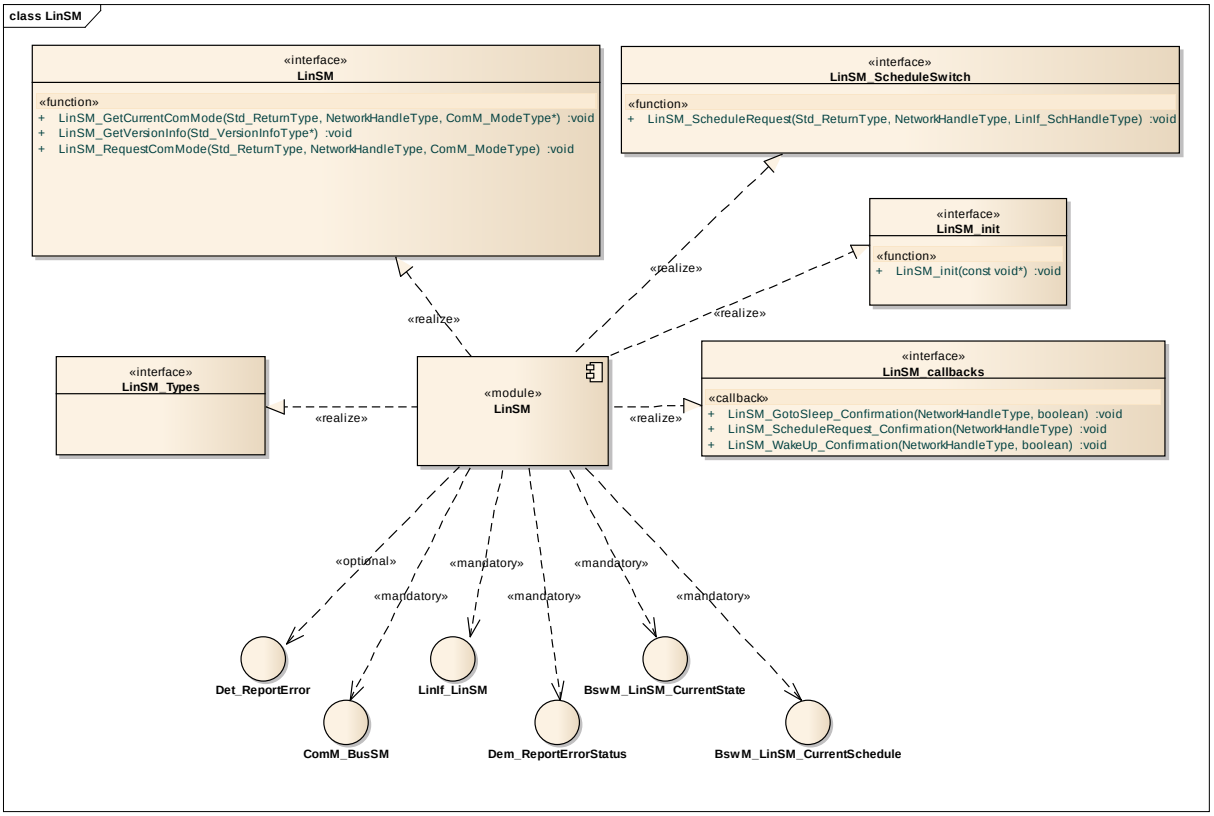

Figure 2 Dependencies to other modules

5.1 Upper layers

5.1.1 Operating System

The LinSM 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. It is up to the implementer to solve these accesses.

5.1.2 Module DET (Development Error Tracer)

In development mode the Det_ReportError – function of module DET [5] will be called.

5.1.3 Module DEM (Diagnostic Event Manager)

Production errors will be reported to the Diagnostic Event Manager [8].

5.1.4 BSW Mode Manager

The LinSM module will notify the BSW Mode Manager [15] module when a state is changed. The BSW Mode Manager module will interface the LinSM module when requesting a new schedule table.

5.1.5 Module using LinSM

The user will have these purposes for the LinSM:
The Com manager requests the communication via the LIN stack.
The Com manager queries the state of the LinSM.

5.2 Lower layers

5.2.1 LinIf

The LinSM assumes the following primitives to be provided by the LinIf [7]:
Transmission of the goto-sleep command (LinIf_GoToSleep)
The wakeup of the Lin bus (LinIf_WakeUp)
Request to change schedule tables (LinIf_ScheduleRequest)

It is assumed that the LinIf will call the following callbacks:
Confirming the Wake Up signals has been transmitted (LinSM_WakeUpConfirmation)
Confirming that the goto-sleep command has been transmitted (LinSM_GotoSleep_Confirmation)
Confirming a schedule change (LinSM_ScheduleRequest_Confirmation)

LINSM002: The LinSM shall not use or access the LIN driver or assume information about it any way other than what the LinIf provides through the function calls to the LinIf listed above.

5.3 File structure

5.3.1 Code file structure

This chapter describes the c-files that implement the LinSM 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, link time and post-build time configuration parameters shall be kept in separate files:
LINSM003: LinSM_Lcfg.c – for link time configurable parameters
LINSM004: LinSM_PBcfg.c – for post build time configurable parameters.
LINSM078: A c-file LinSM_Cfg.c shall exist that contains pre compile parameters that are “const”.

5.3.2 Header File structure

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

5.3.2.1 LinSM header files

Following header files will exist in a LinSM implementation:

LINSM005: A header file LinSM.h shall exist that contains all data exported from the LinSM – API declarations (except callbacks), extern types, and global data.

LINSM006: A header file LinSM_Cbk.h shall exist that contains function declarations for the callback functions in the LinSM.

LINSM007: A header file LinSM_Cfg.h shall exist that contains the pre compile time parameters that are pre processor directives.

LINSM008: The header file LinSM_Cfg.h shall contain declarations of the pre compile, link time and the post-build time configurable parameters.

LINSM009: The LinSM header files (LinSM_Cbk.h, LinSM.h and LinSM_Cfg.h) shall contain the version number, see 10.4.

LINSM010: The LinSM shall check (pre compile-time) that the correct versions of the header files are used. LinSM header files all versions must be checked, for external header files only major and minor versions are checked.

5.3.2.2 Included header files

Figure 3 shows the header file structure of the LinSM realization.

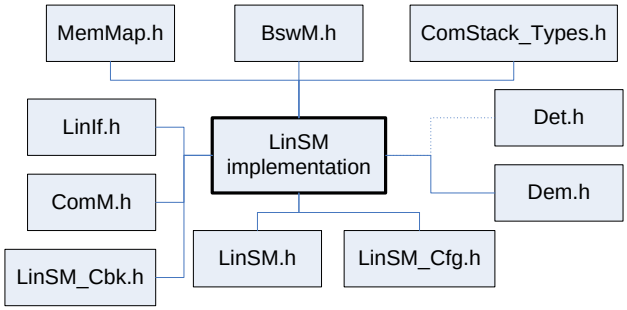

Figure 3 Header file structure

The LinSM implementation object in Figure 3 represent one or more .c files. It is not required to make the complete implementation in one file.

Following external header files shall be included:

LINSM011: The Dem.h file.

LINSM012: The LinSM shall include the LINIf.h file.

LINSM013: The LinSM shall include the ComM.h file

LINSM014: The LinSM shall include the MemMap.h file.

LINSM015: The Det.h file in case LinSMDevErrorDetect is enabled

LINSM016: The ComStack_Types.h
LINSM157: The LinSM module shall include the BswM.h

6 Requirements traceability

This chapter contains a matrix that shows the link between the SWS requirements defined for the LinSM and the input requirement documents (SRS).

The following SRSs acts as input requirements to the LinSM:

AUTOSAR - General Requirements on Basic Software Modules [3]
AUTOSAR -AUTOSAR Requirements on Basic Software Modules, Cluster: LIN [6]
AUTOSAR requirements on Mode Management, Communication Manager [10]

Document: AUTOSAR requirements on Basic Software, general

Requirement	Satisfied by
[BSW00344] Reference to link-time configuration	LINSM003
[BSW00404] Reference to post build time configuration	LINSM042, LINSM004
[BSW00405] Reference to multiple configuration sets	LINSM042
[BSW00345] Pre-compile-time configuration	LINSM007
[BSW159] Tool-based configuration	Chapter 10.2
[BSW167] Static configuration checking	LINSM073
[BSW171] Configurability of optional functionality	N/A for LinSM SWS
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	N/A for LinSM SWS
[BSW00380] Separate C-Files for configuration parameters	LINSM003, LINSM004
[BSW00419] Separate C-Files for pre-compile time configuration parameters	
[BSW00381] Separate configuration header file for pre-compile time parameters	LINSM007
[BSW00412] Separate H-File for configuration parameters	LINSM078
[BSW00383] List dependencies of configuration files	Configuration of LinIf is referenced, see 10.3
[BSW00384] List dependencies to other modules	LINSM001
[BSW00387] Specify the configuration class of callback function	N/A for LinSM
[BSW00388] Introduce containers	Chapter 10.2
[BSW00389] Containers shall have names	Chapter 10.2
[BSW00390] Parameter content shall be unique within the module	Chapter 10.2
[BSW00391] Parameter shall have unique names	Chapter 10.2
[BSW00392] Parameters shall have a type	Chapter 10.2
[BSW00393] Parameters shall have a range	Chapter 10.2
[BSW00394] Specify the scope of the parameters	Chapter 10.2
[BSW00395] List the required parameters (per parameter	Chapter 10.2

[BSW00396] Configuration classes	Chapter 10.2
[BSW00397] Pre-compile-time parameters	Chapter 10.2
[BSW00398] Link-time parameters	Chapter 10.2
[BSW00399] Loadable Post-build time parameters	Chapter 10.2
[BSW00400] Selectable Post-build time parameters	Chapter 10.2
[BSW00438] Post Build Configuration Data Structure	LINSM042
[BSW00402] Published information	LINSM076
[BSW00375] Notification of wake-up	N/A for LinSM
[BSW101] Initialization interface	
[BSW00416] Sequence of Initialization	N/A for LinSM
[BSW00406] Check module initialization	
[BSW168] Diagnostic Interface of SW components	LinSM does not offer a diagnostic interface
[BSW00407] Function to read out published parameters	
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	LinSM has no AUTOSAR interfaces
[BSW00424] BSW main processing function task allocation	LinSM does not provide any task handling
[BSW00425] Trigger conditions for schedulable objects	N/A for LinSM
[BSW00426] Exclusive areas in BSW modules	LinSM does not require any exclusive areas. It may however be used in the implementation.
[BSW00427] ISR description for BSW modules	LinSM has no Interrupt functions defined
[BSW00428] Execution order dependencies of main processing functions	No main function in LinSM
[BSW00429] Restricted BSW OS functionality access	No access of OS operations required
[BSW00431] The BSW Scheduler module implements task bodies	N/A for LinSM
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	N/A for LinSM since the scheduling is made in LinIf
[BSW00433] Calling of main processing functions	N/A for LinSM, it is a requirement on BSW schedule
[BSW00434] The Schedule Module shall provide an API for exclusive areas	N/A for LinSM, it is a requirement on BSW schedule
[BSW00336] Shutdown interface	No shut down interface needed
[BSW00337] Classification of errors	LINSM053
[BSW00338] Detection and Reporting of development errors	Chapter 10.3
[BSW00369] Do not return development error codes	Chapter 8

via API	
[BSW00339] Reporting of production relevant error status	LINSM058
[BSW00422] Pre--de--bouncing of production relevant error status	N/A for LinSM
[BSW00417] Reporting of Error Events by Non-Basic Software	N/A for LinSM
[BSW00323] API parameter checking	LINSM083
[BSW004] Version check	LINSM010
[BSW00409] Header files for production code error IDs	N/A for LinSM
[BSW00385] List possible error notifications	LINSM052
[BSW00386] Configuration for detecting an error	N/A for LinSM
[BSW161] Microcontroller abstraction	N/A for LinSM
[BSW162] ECU layout abstraction	N/A for LinSM
[BSW005] No hard coded horizontal interfaces within MCAL	N/A for LinSM
[BSW00415] User dependent include files	N/A for LinSM
[BSW164] Implementation of interrupt service routines	No required interrupt functions
[BSW00325] Runtime of interrupt service routines	No required interrupt functions
[BSW00326] Transition from ISRs to OS tasks	No required interrupt functions
[BSW00342] Usage of source code and object code	The LinSM configuration enables both the source code and the binary way. See chapter 10
[BSW00343] Specification and configuration of time	N/A for LinSM
[BSW160] Human-readable configuration data	the generation of the configuration data is not in the scope of LinSM
[BSW007] HIS MISRA C	This is mostly a requirement on the construction and not the design (i.e. SWS). The API chapter 8 is following MISRA C
[BSW00300] Module naming convention	Chapter 5.3
[BSW00413] Accessing instances of BSW modules	There is only one instance of the LinSM
[BSW00347] Naming separation of different instances of BSW drivers	LinSM is not a driver
[BSW00305] Self-defined data types naming convention	Chapter 8.2
[BSW00307] Global variables naming convention	The naming of parameters is made in chapter 10
[BSW00310] API naming convention	Chapter 8.3
[BSW00373] Main processing function naming convention	N/A for LinSM

[BSW00327] Error values naming convention	LINSM053
[BSW00335] Status values naming convention	Use throughout the document
[BSW00350] Development error detection keyword	LINSM053
[BSW00408] Configuration parameter naming convention	Chapter 10.2
[BSW00410] Compiler switches shall have defined values	Construction requirement and not a design requirement
[BSW00411] Get version info keyword	Chapter 10.2
[BSW00346] Basic set of module files	Chapter 5.3
[BSW158] Separation of configuration from implementation	Chapter 5.3
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	Chapter 5.3
[BSW00314] Separation of interrupt frames and service routines	No interrupt functions in LinSM
[BSW00370] Separation of callback interface from API	Chapter 1.1
[BSW00348] Standard type header	Chapter 5.3
[BSW00353] Platform specific type header	Chapter 5.3
[BSW00361] Compiler specific language extension header	Chapter 5.3
[BSW00301] Limit imported information	Chapter 5.3
[BSW00302] Limit exported information	Chapter 5.3
[BSW00328] Avoid duplication of code	Implementation specific issue
[BSW00312] Shared code shall be reentrant	Some API calls needs to be reentrant. See chapter 8.
[BSW006] Platform independency	Chapter 1
[BSW00357] Standard API return type	Chapter 8.3
[BSW00377] Module specific API return types	No module specific return types needed. See chapter 8.
[BSW00304] AUTOSAR integer data types	Chapter 10
[BSW00355] Do not redefine AUTOSAR integer data types	No redefine made
[BSW00378] AUTOSAR boolean type	No Boolean return types used
[BSW00306] Avoid direct use of compiler and platform specific keywords	No platform specific keywords are used.
[BSW00308] Definition of global data	No global data is required.
[BSW00309] Global data with read-only constraint	No global data is required.
[BSW00371] Do not pass function pointers via API	Function pointers not used, Chapter 8.3
[BSW00358] Return type of init() functions	
[BSW00414] Parameter of init function	
[BSW00376] Return type and parameters of main processing functions	N/A for LinSM
[BSW00359] Return type of callback functions	N/A for LinSM
[BSW00360] Parameters of callback functions	N/A for LinSM
[BSW00329] Avoidance of generic interfaces	Generic interfaces are not used
[BSW00330] Usage of macros / inline functions	No restriction

instead of functions	
[BSW00331] Separation of error and status values	N/A for LinSM
[BSW009] Module User Documentation	SWS template 1.19 is used
[BSW00401] Documentation of multiple instances of configuration parameters	Chapter 10.2
[BSW172] Compatibility and documentation of scheduling strategy	Chapter 8
[BSW010] Memory resource documentation	N/A for LinSM
[BSW00333] Documentation of callback function context	N/A for LinSM
[BSW00374] Module vendor identification	LINSM076
[BSW00379] Module identification	LINSM076
[BSW003] Version identification	LINSM009
[BSW00318] Format of module version numbers	LINSM076
[BSW00321] Enumeration of module version numbers	N/A for LinSM
[BSW00341] Microcontroller compatibility documentation	N/A for LinSM
[BSW00334] Provision of XML	N/A for LinSM
[BSW00435] Header File Structure for the Basic Software Scheduler	N/A for LinSM

Document: AUTOSAR requirements on Basic Software, cluster LIN

Requirement	Satisfied by
[BSW01501] Usage of LIN 2.0 specification	LINSM017
[BSW01504] Usage of AUTOSAR architecture only in LIN master	LINSM017
[BSW01522] Consistent data transfer	The LinSM will not make any copying of data from unprotected buffers
[BSW01560] Support for wakeup during transition to sleep-mode	N/A for LinSM
[BSW01567] Compatibility to LIN 2.0 protocol specification	LINSM017
[BSW01551] Multiple LIN channel support for interface	Chapter 7.3
[BSW01568] Hardware independence	LinSM is not dependent on hardware
[BSW01569] LIN Interface initialization	LinIf requirement
[BSW01570] Selection of static configuration sets	Chapter 1
[BSW01564] Schedule Table Manager	LinIf requirement
[BSW01546] Schedule Table Handler	LinIf requirement
[BSW01561] Main function	LinIf requirement
[BSW01549] Timer service for Scheduling	LinIf requirement
[BSW01571] Transmission request service	LinIf requirement
[BSW01514] Wake-up notification support	LinIf requirement
[BSW01515] API to wake-up by upper layer to LIN Interface	LinIf requirement

[BSW01502] RX indication and TX confirmation call-backs	LinIf requirement
[BSW01558] Check successful communication	LinIf requirement
[BSW01527] Notification for missing or erroneous receive LIN-PDU	LinIf requirement
[BSW01523] API to send the LIN to sleep-mode	LinIf requirement
[BSW01565] Compatibility to LIN 2.0 protocol specification	LIN driver requirement
[BSW01553] Basic Software SPAL General requirements	LIN driver requirement
[BSW01552] Hardware abstraction LIN	LIN driver requirement
[BSW01503] Frame based API for send and received data	LIN driver requirement
[BSW01555] LIN Interface shall poll the LIN Driver for transmit/receive notifications	LIN driver requirement
[BSW01547] Support of standard UART and LIN optimised	LIN driver requirement
[BSW01572] LIN driver initialization	LIN driver requirement
[BSW01573] Selection of static configuration sets	LIN driver requirement
[BSW01563] Wake-up Notification	LIN driver requirement
[BSW01556] Multiple LIN channel support for driver	LIN driver requirement
[BSW01566] Transition to sleep-mode mode	LIN driver requirement
[BSW01524] Support of reduced power operation mode	LIN driver requirement
[BSW01526] Error notification	LIN driver requirement
[BSW01533] Usage of LIN 2.0 specification	LIN driver requirement
[BSW01540] LIN Transport Layer Initialization	LinIf requirement
[BSW01545] LIN Transport Layer Availability	LinIf requirement
[BSW01534] Concurrent connection configuration	LinIf requirement
[BSW01574] Multiple Transport Layer instances	LinIf requirement
[BSW01539] Transport connection properties	LinIf requirement
[BSW01544] Error handling	LinIf requirement

Document: AUTOSAR requirements on Mode Management, Communication Manager

Requirement	Satisfied by
[BSW09090] User-to-channel relationship	Chapter 10.3
[BSW09133] Assigning physical channels to the Communication Manager	Chapter 10.3
[BSW09132] Assigning Network Management to physical channels	N/A for LinSM
[BSW09141] Configuration of physical channel wake-up	N/A for LinSM
[BSW09078] Coordinating communication requests	N/A for LinSM
[BSW049] Initiating wake-up and keeping awake physical channels	N/A for LinSM
[BSW09080] Physical channel independency	LINSM021
[BSW09081] API for requesting communication	

[BSW09083] Support of different communication modes	LINSM026, LINSM032
[BSW09084] API for querying the current communication	
[BSW09149] API for querying the requested communication mode	N/A for LinSM
[BSW09085] Indication of communication mode changes	LINSM027, LINSM033
[BSW00437] NoInit--Area in RAM	N/A for LinSM
[BSW09168] Pseudo-channel for local	N/A for LinSM
[BSW09071]Limit Communication Manager modes	N/A for LinSM
[BSW09157] Revoke Communication Manager mode limitation	N/A for LinSM
[BSW09087] Proxy communication request after wake-up	N/A for LinSM
[BSW09088] Handling of different physical channel types	N/A for LinSM
[BSW09089] Preventing waking up physical channels	N/A for LinSM
[BSW09155] Counting of inhibited communication requests	N/A for LinSM
[BSW09156] API to retrieve the number of inhibited "Full Communication" mode requests	N/A for LinSM
[BSW09079] Transparent relationship between software components and physical channels	N/A for LinSM

7 Functional specification

This chapter specifies the requirements on the module LinSM. See the Basic Software Modules document [2] 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 channels
Handle schedule change request
Notify of state changes to upper layers

LINSM017: The LinSM shall support the behavior of the master in the LIN 2.0 specification.

LINSM018: It is required that the master behavior that is supported by the LinSM is constructed so that existing slaves can be reused.

LINSM019: The LinSM shall be able to handle one or more LIN channels.

The identification of the LIN channels are handled by the configuration (LinSMChannelIndex).

7.1 State Machine

The LinSM will operate in a state-machine. Each channel connected will operate in a sub-state-machine.

LINSM020: The LinSM shall have one state-machine.

The State Machine is depicted in Figure 4.

LINSM021: Each LIN Channel has a separate sub-state-machine.

The sub-state-machine is depicted in Figure 4.

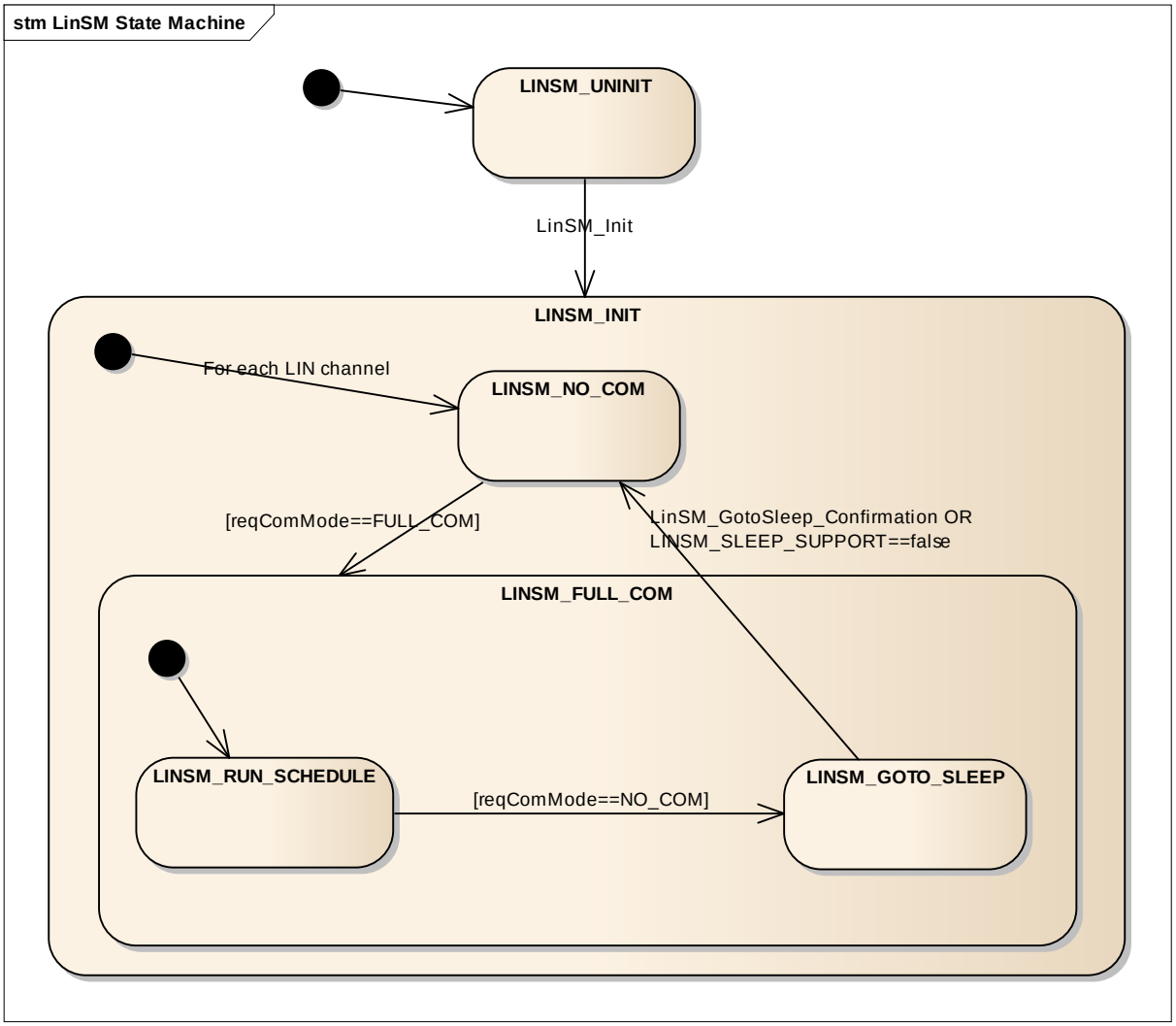

Figure 4: LinSM State Machine

7.1.1 LINSM_UNINIT

LINSM022: There shall be a state called LINSM_UNINIT

7.1.2 LINSM_INIT

LINSM024: There shall be a state called LINSM_INIT

LINSM025: The LinSM state-machine shall transit from any state or sub-state to the state LINSM_INIT when LinSM_Init is called.

LINSM152: The LinSM state-machine shall transit from any state or sub-state to sub-state LINSM_NO_COM when LinSM_Init is called.

To make the LinSM independent from the LinIf the LinSM should not call the LinIf in when the LinSm is in the init function. The LinSM_init has therefore additional requirement, see 8.3.1. Design note: Therefore the LinIf may make a callout when setting the NULL schedule in its init function.

Following sections sets further requirements on the state-machine and the sub-state-machine.

7.1.3 LINSM_NO_COM

LINSM026: There shall be a state called LINSM_NO_COM in the state LINSM_INIT.

LINSM027: When entering LINSM_NO_COM the ComM shall be notified of the state change by calling the ComM_BusSm_ModeIndication and BswM_LinSM_CurrentState with the parameter LINSM_NO_COM.

LINSM028: In this state the LinSM shall not command the LinIf to communicate, i.e. bus shall be silent.

LINSM029: If full communication is requested the wakeup process, 7.2.3, shall be followed.

LINSM030: If wakeup process is finished the state shall be changed to LINSM_FULL_COM.

LINSM031: Upon entering or exiting the LINSM_NO_COM state the LinSM will not set the hardware interface or µ-controller into a new power mode. This is not in the scope of the LinSM

LINSM158: 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.

7.1.4 LINSM_FULLCOM

LINSM032: There shall be a state called LINSM_FULL_COM in the state LINSM_INIT.

LINSM033: When entering LINSM_FULL_COM the ComM shall be notified of the state change by calling the ComM_BusSm_ModeIndication.

LINSM034: When entering LINSM_FULL_COM the sub-state LINSM_RUN_SCHEDULE shall be activated and BswM_LinSM_CurrentState with the parameter LINSM_RUN_SCHEDULE shall be called.

LINSM035: The LinSM shall only command the LinIf to communicate in the state LINSM_FULLCOM.

LINSM036: If no communication is requested the sub-state shall be changed to LINSM_GOTOSLEEP and BswM_LinSM_CurrentState with the parameter LINSM_GOTO_SLEEP shall be called.

The LINSM_FULL_COM state has two sub-states:

7.1.4.1 Sub-state LINSM_RUN_SCHEDULE

In this sub-state normal communication is processed.

LINSM037: There shall be a sub-state called LINSM_RUN_SCHEDULE in the state LINSM_FULL_COM.

LINSM038: This is the only sub-state where schedule table change requests are accepted.

LINSM079: If a new schedule is requested the LinSM shall forward the request to the LinIf.

LINSM159: If LinIf has confirmed the schedule change, then LinSM shall call BswM_LinSM_CurrentSchedule.

Be aware of that the LinIf will switch to a NULL schedule when entering sleep, then it may make a schedule switch callout.

7.1.4.2 Sub-state LINSM_GOTOSLEEP

LINSM039: There shall be a sub-state LINSM_GOTOSLEEP in the state LINSM_FULL_COM.

LINSM040: The goto sleep process, 7.2.2, shall be started when entering this sub-state.

LINSM041: When the goto sleep process, 7.2.2, is finished the state shall be changed to LINSM_NO_COM

7.2 Initialization process

7.2.1.1 LinSM_Init process

LINSM042: The LinSM_Init function shall accept a parameter that references to a LinSM post-build configuration descriptor.

The configuration descriptor is not specified in this specification. It is up to each implementer to define it.

LINSM043: 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). It shall e possible to set the LinSM in a default state regardless of the previous state.

7.2.2 Go to sleep process

LINSM044: When entering LINSM_GOTOSLEEP the LinSM shall call the LinIf function LinIf_GotoSleep.

LINSM045: The state LINSM_NO_COM shall be set, after the LinIf indicates that the goto-sleep command was sent successfully on the bus.

LINSM046: The state LINSM_FULL_COM shall be set, after the LinIf indicates that the goto-sleep command was NOT sent successfully on the bus.

7.2.3 Wake up process

A LIN channel will only be woken up if the upper layer requests a wake up through the LinSM_RequestComMode call. In this case the ComM will request full communication.

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

LINSM047: If the state is LINSM_NO_COM, the LinSMSleepSupport is true and the ComM requests full communication; the LinSM calls LinIf_WakeUp to the LinIf to transmit a wake up signal on the requested channel (LinSMChannelIndex).

LINSM048: If the state is LINSM_NO_COM, the LinSMSleepSupport is true and the ComM requests full communication; the LinSM will change state to LINSM_FULL_COM.

LINSM049: When the LinIf notifies that the Wake Up is sent, the state shall be set to LINSM_FULL_COM.

7.2.4 Timeout of requests

After calling LinIf_GotoSleep, LinIf_WakeUp or schedule request the LINSM is waiting for the LinIf to confirm the transmission of the goto sleep command, the wake up on the bus or schedule is changed. There is a possibility that the confirmation is not received, and therefore the LinSM will wait forever. The only cause for this situation is problem in the software, i.e. no bus event or similar can cause this situation.

LINSM100: When the LinIf_GotoSleep, LinIf_WakeUp or LinIf_ScheduleRequest are called, the LinSM shall start a timer.

LINSM154: If the callout is made before the timer times out, the timer shall be stopped. So that the timeout will not occur.

LINSM101: When the timer expires, i.e. greater than LinSMConfirmationTimeout, the LinSM shall set the state to LINSM_NO_COM.

LINSM102: When the timer expires, the LinSM shall send LINSM_E_CONFIRMATION_TIMEOUT to DEM.

LINSM103: If the LinSMConfirmationTimeout is set to zero the timer is not used.

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

LINSM104: If the callback is received after the timeout it shall be ignored.

7.3 Handling multiple channels and drivers

Usually only one LIN driver (supporting multiple channels) is needed in an ECU to handle all LIN channels. However, rarely, some hardware configurations the ECU contain different LIN hardware. In such case, more than one different LIN drivers are required. This will not affect the LinSM since the LIN driver only interfaces the LinIf.

The LinSM will only handle channels, and is not concerned to which driver it maps to.

7.3.1 Multiple channels

LINSM050: Each channel has a unique channel index (LinSMChannelIndex) in the LinIf configuration. The LinSM shall use the same index when interfacing to the LinIf (when channel is required as a parameter).

This means that no mapping between channels has to be made in the LinSM when interfacing to the LinIf. The channel index may be used directly to the LinIf APIs.

7.4 Error classification

This chapter lists and classifies errors that can be detected within this software module. Each error is classified according to relevance (development / production) and related error code. For development errors, a value is defined.

LINSM051: Values for production code Event Ids are assigned externally by the configuration of the Dem [8]. They are included via Dem.h.

The Dem.h includes the APIs to report errors as well as the required Event Id symbols. The table below defines the name of the Event Id symbols that are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

LINSM052: Development error values are of type uint8.

The Table 1 shows the available error codes (to DET and DEM) for the LinSM. The list of production errors is complete. The development errors consist of a minimum number of errors detected. It is possible for the implementer to extend the development errors.

LINSM053:

Type or error	Relevance	Related error code	Value [hex]
API called without initialization of LinSM	Development	LINSM_E_UNINIT	0x00
Initialization API is used when already initialized	Development	LINSM_E_ALREADY_INITIALIZED	0x10
Referenced channel does not exist (identification is out of range)	Development	LINSM_E_NONEXISTENT_CHANNEL	0x20
API service called with wrong parameter	Development	LINSM_E_PARAMETER	0x30
API service called with invalid pointer	Development	LINSM_E_PARAMETER_POINTER	0x40
Timeout of the callbacks from LinIf	Production error	LINSM_E_CONFIRMATION_TIMEOUT	Assigned by DEM

Table 1 - Error codes for DET and DEM

7.5 Error detection

LINSM054: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch `LinSMDevErrorDetect` (see chapter 10) shall activate or deactivate the detection of all development errors.

LINSM055: If the `LinSMDevErrorDetect` switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.4 and chapter 8.

LINSM083: If the development errors are enabled, all APIs in chapter 8 shall check ranges of the incoming and outgoing parameters. In case there is a parameter is out of range, the DET shall be reported with LINSM_E_PARAMETER.

LINSM056: It shall not be possible to switch off detection and reporting of production code errors.

7.6 Error notification

LINSM057: Detected development errors will be reported to the Det_ReportError API of the Development Error Tracer (DET) [5] if the pre-processor switch `LinSMDevErrorDetect` is set (see chapter 10).

LINSM058: Production errors shall be reported to Diagnostic Event Manager (DEM) [8].

8 API specification

8.1 Imported types

8.1.1 Standard types

In this chapter all types included from the following files are listed. The standard AUTOSAR types are defined in the AUTOSAR Specification of Standard Types document [4].

Module	*Imported Type*
ComM	ComM_ModeType
ComStack_Types	NetworkHandleType
Dem	Dem_EventIdType
LinIf	LinIf_SchHandleType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Type definitions

Following type is defined by the LinSM module:

Name:	`LinSM_ModeType`		
Type:	`uint8`		
Range:	`LINSM_NO_COM`	1	No communication
	`LINSM_RUN_SCHEDULE`	2	LinSM mode FULL_COM and run schedule is active
	`LINSM_GOTO_SLEEP`	3	LinSM mode FULL_COM and wait for sleep confirmation
Description:	Type used to report the current mode to the BswM		

8.3 LinSM API

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

8.3.1 LinSM_Init

Service name:	LinSM_init	

Syntax:	`void LinSM_init(`
	`  const void* ConfigPtr`
	`)`	

Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Not reentrant	
Parameters (in):	ConfigPtr	Pointer to the LinSM configuration
Parameters (inout):	None	

Parameters (out):	None
Return value:	None
Description:	This function initializes the LinSM. Note that in some implementations other values of the pointer than NULL may be considered faulty. Configuration dependent on Variant (see parameter)

LINSM060: The ConfigPtr parameter is only relevant for post-build. It shall be ignored for Variant 1 and Variant 2. The parameter will still be there for compatibility reasons.

As seen in the configuration parameters chapter there is no post-build parameters. The parameter to the init function acts as a pointer to a post-build selectable.

LINSM090 : If LinSMDevErrorDetect is enabled: If the ConfigPtr is invalid (e.g. NULL) the LinSM shall report LINSM_E_PARAMETER_POINTER to the DET.

LINSM151: No other modules shall be called from the LinSM_init function.

8.3.2 LinSM_ScheduleRequest

LINSM113

Service name:	LinSM_ScheduleRequest	

Syntax:	`Std_ReturnType LinSM_ScheduleRequest(`
	`  NetworkHandleType channel,`
	`  LinIf_SchHandleType schedule`
	`)`	

Service ID[hex]:	0x12	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	channel	Identification of the LIN channel (LINSM_CHANNEL_INDEX)
	schedule	Pointer to the new Schedule table
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK - Schedule table request has been accepted. E_NOT_OK - Schedule table switch request has not been accepted due to one of the following reasons: * LinSM has not been initialized referenced channel does not exist (identification is out of range) * Referenced schedule table does not exist (identification is out of range) * No more requests are allowed because the request queue is full * Sub-state is not LIN_RUNSCHEDULE
Description:	The upper layer requests a schedule table to be changed on one LIN channel.	

LINSM114 If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value the LinSM shall report LINSM_E_NONEXISTENT_CHANNEL to the DET.

LINSM153: If LinSMDevErrorDetect is enabled: If the sub-state is not LINSM_RUN_SCHEDULE, the LinSM shall report LINSM_E_NOT_IN_RUN_SCHEDULE to the DET.

LINSM115 If LinSMDevErrorDetect is enabled: If the Schedule parameter has an invalid value the LinSM shall report LINSM_E_PARAMETER to the DET.

LINSM116 If LinSMDevErrorDetect is enabled: When the state LINSM_UNINIT is active then LINSM_E_UNINIT shall be reported to DET.

8.3.3 LinSM_GetVersionInfo

LINSM117

Service name:	LinSM_GetVersionInfo	
Syntax:	`void LinSM_GetVersionInfo( Std_VersionInfoType* versioninfo )`	
Service ID[hex]:	0x13	
Sync/Async:	Synchronous	
Reentrancy:	Non 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:	--	

LINSM118 The version number consists of three parts:
Two bytes for the vendor ID
One byte for the module ID
Three bytes version number. The numbering shall be vendor specific; it consists of the major, the minor and the patch version number of the module.
The AUTOSAR specification version number shall not be included.

Hint:
If source code this function may be realized as a macro. The macro will then be defined in the modules header file.

LINSM119 If the versioninfo pointer parameter is invalid (e.g. NULL), the LinSM shall report LINSM_E_PARAMETER_POINTER to the DET. If LinSMDevErrorDetect is enabled.

LINSM120 When the state LINSM_UNINIT is active then LINSM_E_UNINIT shall be reported to DET. If LinSMDevErrorDetect is enabled.

LINSM121 This function shall be pre compile time configurable On/Off by the configuration parameter: LinSMVersionInfoApi

8.3.4 LinSM_GetCurrentComMode

LINSM122

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 (LINSM_CHANNEL_INDEX)
Parameters (inout):	None	
Parameters (out):	mode	See ComM_ModeType
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.	

LINSM123 If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value the LinSM shall report LINSM_E_NONEXISTENT_CHANNEL to the DET.

LINSM124 If LinSMDevErrorDetect is enabled: If the mode pointer parameter is invalid (e.g. NULL), the LinSM shall report LINSM_E_PARAMETER_POINTER to the DET.

LINSM125 If LinSMDevErrorDetect is enabled: When the state LINSM_UNINIT is active then LINSM_E_UNINIT shall be reported to DET.

8.3.5 LinSM_RequestComMode

LINSM126

Service name:	LinSM_RequestComMode	
Syntax:	`Std_ReturnType LinSM_RequestComMode( NetworkHandleType network, ComM_ModeType mode )`	
Service ID[hex]:	0x10	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	network	Identification of the LIN channel (LINSM_CHANNEL_INDEX)
	mode	Name of the requested 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 by ComM.	

	The mode switch will not be made instant. The LinSM will notify the ComM when mode transition is made.

LINSM127 If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value the LinSM shall report LINSM_E_NONEXISTENT_CHANNEL to the DET.

LINSM128 If LinSMDevErrorDetect is enabled: When the state LINSM_UNINIT is active then LINSM_E_UNINIT shall be reported to DET.

[LINSM160]⌈ 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 7.1 refers to this stored request as reqComMode.⌋()

8.4 LinSM callouts

The function prototypes of the callback functions will be provided in the file LinSM_Cbk.h

8.4.1 LinSM_ScheduleRequest_Confirmation

LINSM129

Service name:	LinSM_ScheduleRequest_Confirmation
Syntax:	`void LinSM_ScheduleRequest_Confirmation( NetworkHandleType channel )`
Service ID[hex]:	0x22
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	channel: Identification of the LIN channel (LINSM_CHANNEL_INDEX)
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.

LINSM130 If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value the LinSM shall report LINSM_E_NONEXISTENT_CHANNEL to the DET.

LINSM131 If LinSMDevErrorDetect is enabled: When the state LINSM_UNINIT is active then LINSM_E_UNINIT shall be reported to DET.

8.4.2 LinSM_WakeUp_Confirmation

LINSM132

Service name:	LinSM_WakeUp_Confirmation	
Syntax:	`void LinSM_WakeUp_Confirmation( NetworkHandleType channel, boolean success )`	
Service ID[hex]:	0x21	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	channel	Identification of the LIN channel (LINSM_CHANNEL_INDEX)
	success	True if wakeup was successfully sent, false otherwise
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	The LinIf will call this callout when the wake up signal command is sent not successfully/successfully on the bus.	

LINSM133 If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value the LinSM shall report LINSM_E_NONEXISTENT_CHANNEL to the DET.

LINSM134 If LinSMDevErrorDetect is enabled: When the state LINSM_UNINIT is active then LINSM_E_UNINIT shall be reported to DET.

8.4.3 LinSM_GotoSleep_Confirmation

LINSM135

Service name:	LinSM_GotoSleep_Confirmation	
Syntax:	`void LinSM_GotoSleep_Confirmation( NetworkHandleType channel, boolean success )`	
Service ID[hex]:	0x20	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	channel	Identification of the LIN channel (LINSM_CHANNEL_INDEX)
	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 callout when the go to sleep command is sent not successfully/successfully on the bus.	

LINSM136 If LinSMDevErrorDetect is enabled: If the network parameter has an invalid value the LinSM shall report LINSM_E_NONEXISTENT_CHANNEL to the DET.

LINSM137 If LinSMDevErrorDetect is enabled: When the state LINSM_UNINIT is active then LINSM_E_UNINIT shall be reported to DET.

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 that are required to fulfill the core functionality.

API function	*Description*
BswM_LinSM_CurrentSchedule	Function called by LinSM to indicate the currently active schedule table for a specific LIN channel.
BswM_LinSM_CurrentState	Function called by LinSM to indicate its current state.
ComM_BusSM_ModeIndication	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 (see ComM661).
Dem_ReportErrorStatus	Reports errors to the DEM.
LinIf_GotoSleep	Initiates a transition into the Sleep Mode on the selected channel.
LinIf_ScheduleRequest	Requests a schedule table to be executed.

8.5.2 Optional interfaces

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

LINSM138

API function	*Description*
Det_ReportError	Service to report development errors.

8.5.3 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

This use-case in Figure 5 shows the transition into the LINSM_GOTOSLEEP state when the goto-sleep command has been sent successfully on the bus.

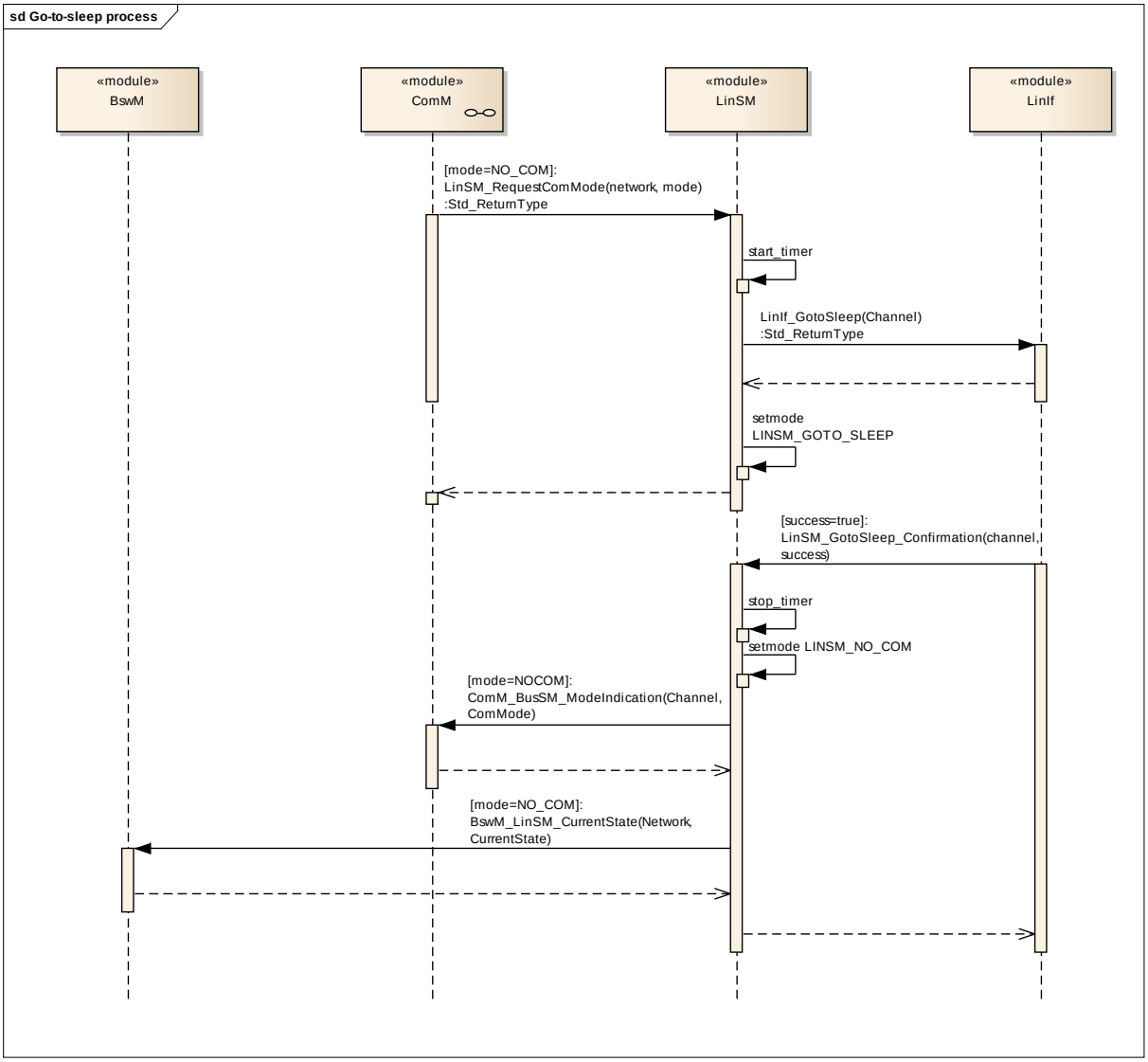

Figure 5 - Goto-sleep–command process

9.2 Internal wake up

The Figure 6 shows the internal wakeup. A wakeup is requested from module above the LinSM.

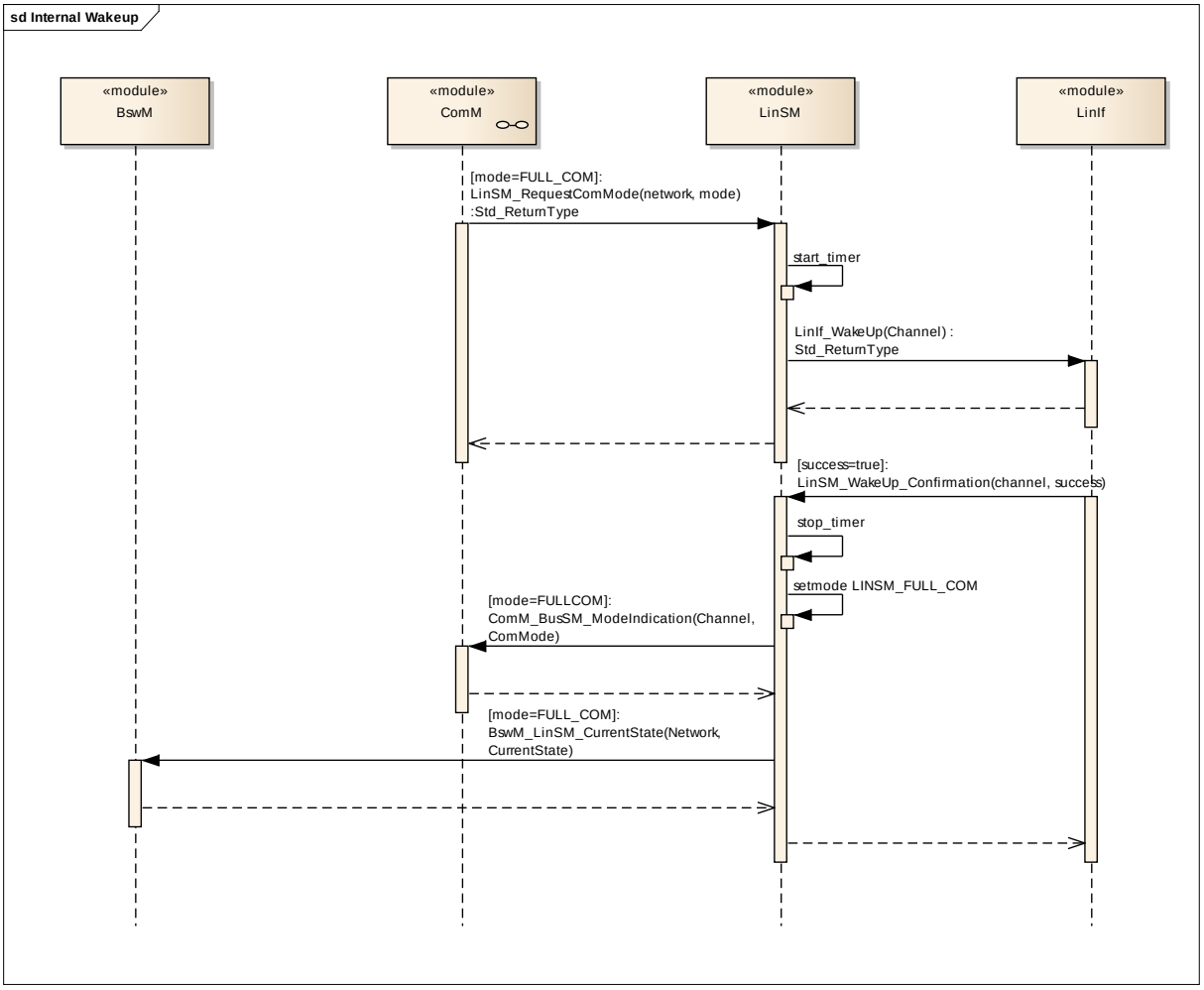

Figure 6 - Internal wake up

9.3 Schedule switch

Figure 7 shows the use-cases of switching the schedule table when the LinSM accepts the request.

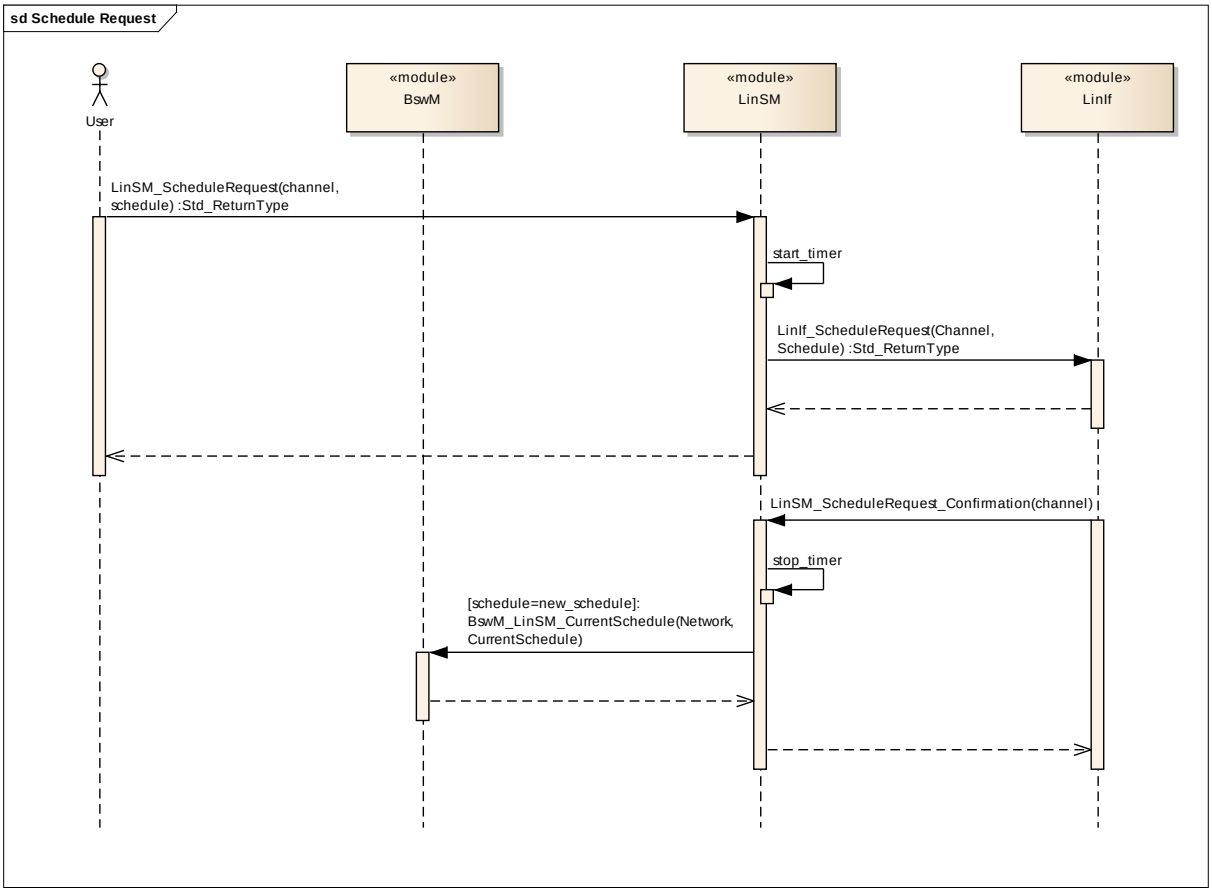

Figure 7: Schedule Table switch

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 module LinSM.

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:
AUTOSAR Layered Software Architecture [2]
AUTOSAR ECU Configuration Specification [9] - This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration meta-model 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: pre 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 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.

10.1.3 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 Chapter 8.

LINSM072: For post-build time support, the LinSM configuration structure LinSM_Configuration shall be constructed so that it may be exchangeable in memory.

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 LinSM.

LINSM073: The LinSM shall not make any consistency check of the configuration in run-time in production software. It may be done if the Development Error Detection is enabled.

10.2.2 Variants

Following configuration variants are defined for LinSM.

10.2.2.1 Variant1 (Pre-compile Configuration)

In the pre-compile configuration all parameters below that are marked as Pre-compile configurable shall be configurable in a pre-compile manner, for example as #defines.

The module is most likely delivered as source code.

10.2.2.2 Variant2 (Link-time Configuration)

This configuration includes all configuration options of the “Pre-compile Configuration”. Additionally all parameters defined below, as link-time configurable shall be configurable at link time for example by linking a special configured parameter object file.

The module is most likely delivered as object code.

10.3 LinSM_Configuration

The paragraph defines the LinSM configuration.

10.3.1 LinSM

Module Name	*LinSM*
Module Description	Configuration of the Lin State Manager module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
LinSMChannel	1..*	Describes each LIN channel the LinSM is connected to.
LinSMGeneral	1	The general parameters fo this module.

10.3.2 LinSMGeneral

SWS Item	**LINSM139 :**
Container Name	LinSMGeneral
Description	The general parameters fo this module.
Configuration Parameters	

SWS Item	**LINSM141 :**
Name	LinSMDevErrorDetect {LINSM_DEV_ERROR_DETECT}
Description	Switches the Development Error Detection and Notification ON or OFF.
Multiplicity	1
Type	BooleanParamDef

Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**LINSM140 :**		
Name	LinSMVersionInfoApi {LINSM_VERSION_INFO_API}		
Description	Switches the LinSM_GetVersionInfo function ON or OFF.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.3.3 LinSMChannel

SWS Item	**LINSM142 :**
Container Name	LinSMChannel{LinSM_ChannelReference} [Multi Config Container]
Description	Describes each LIN channel the LinSM is connected to.
Configuration Parameters	

SWS Item	**LINSM144 :**		
Name	LinSMConfirmationTimeout {LINSM_CONFIRMATION_TIMEOUT}		
Description	Timeout in seconds for the goto sleep and wakeup calls to LinIf. The timeout must be longer than a goto-sleep command on the bus (i.e. it is bit rate dependent).		
Multiplicity	1		
Type	FloatParamDef		
Range	-INF .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**LINSM143 :**
Name	LinSMSleepSupport {LINSM_SLEEP_SUPPORT}
Description	Some LIN clusters does not need sleep, they will just shut off. This parameter will affect the behavior to achieve the no communication state. True - The LinSM will request the LinIf to send the goto sleep command before going to no communication state LINSM_NOCOM. False - The LinSM will directly go to no communication state LINSM_NOCOM.
Multiplicity	1
Type	BooleanParamDef
Default value	--

ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**LINSM145 :**		
Name	LinSMChannelIndex {LINSM_CHANNEL_INDEX}		
Description	Index of the controller. To be able to interface to the correct hw.		
Multiplicity	1		
Type	Reference to [LinIfChannel]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
LinSMSchedule	1..*	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).

10.3.4 LinSMSchedule

SWS Item	**LINSM146 :**
Container Name	LinSMSchedule{LinSM_Schedule}
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).
Configuration Parameters	

SWS Item	**LINSM149 :**		
Name	LinSMScheduleIndexRef {LINSM_SCHEDULE_INDEX_REF}		
Description	Reference to a schedule table in the LinIf configuration		
Multiplicity	1		
Type	Reference to [LinIfScheduleTable]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.4 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

The standard common published information like

vendorId (<Module>_VENDOR_ID),
moduleId (<Module>_MODULE_ID),
arMajorVersion (<Module>_AR_MAJOR_VERSION),
arMinorVersion (<Module>_ AR_MINOR_VERSION),
arPatchVersion (<Module>_ AR_PATCH_VERSION),
swMajorVersion (<Module>_SW_MAJOR_VERSION),
swMinorVersion (<Module>_ SW_MINOR_VERSION),
swPatchVersion (<Module>_ SW_PATCH_VERSION),
vendorApiInfix (<Module>_VENDOR_API_INFIX)

is provided in the BSW Module Description Template (see [14] Figure 4.1 and Figure 7.1).

Additional published parameters are listed below if applicable for this module.