Document Title	Specification of LIN State Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	255
Document Classification	Standard
Document Status	Final
Part of AUTOSAR Release	4.2.2

Document Change History

Release	Changed by	Change Description
4.2.2	AUTOSAR Release Management	• Modified header file structure • Debugging support marked as obsolete • Editorial changes
4.2.1	AUTOSAR Release Management	• Removed NULL pointer check requirement ➔ moved to BSW General • Corrections in ECU parameter configuration
4.1.3	AUTOSAR Release Management	• Editorial changes
4.1.2	AUTOSAR Release Management	• Minor bug fixes • Editorial changes • Removed chapter(s) on change documentation
4.1.1	AUTOSAR Administration	• LIN wakeup and sleep mode handling corrected
4.0.3	AUTOSAR Administration	• Added post-build configuration support • Added completion of Production error concept in Com Stack • Removed local network index
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
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
3.1.1	AUTOSAR Administration	• Legal disclaimer revised
3.0.1	AUTOSAR Administration	• Initial Release

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 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 [11] (ComM) and lower module LIN Interface [7] (LinIf).

It is assumed that the reader is familiar with the LIN 2.1 specification [14]. This document will not describe functionality already described in the LIN 2.1 specification [14].

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 [1] positions the LinSM within the BSW architecture as shown below.

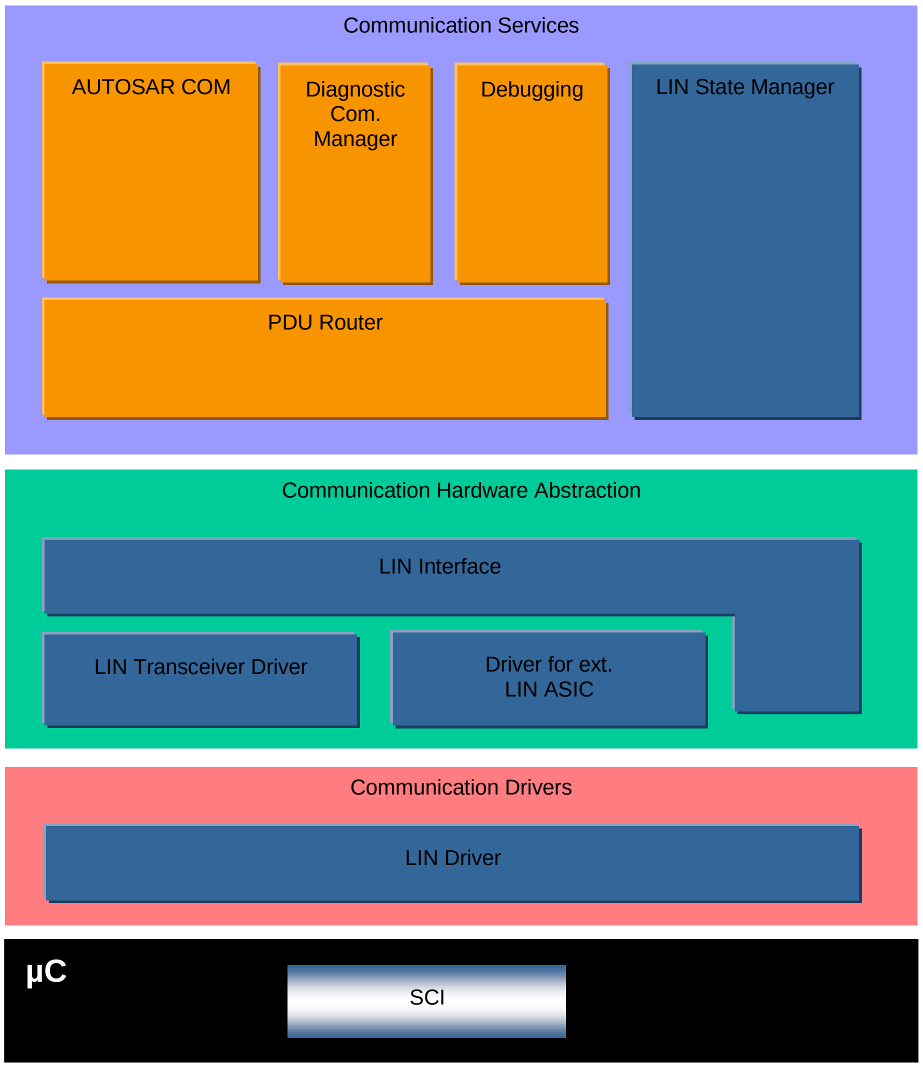

Figure 2 – AUTOSAR BSW software architecture - LIN stack scope

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)
- 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 LIN 2.1 specification [14].

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

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture,
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

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

[4] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[5] Specification of Default Error Tracer
AUTOSAR_SWS_DefaultErrorTracer.pdf

[6] Requirements on LIN
AUTOSAR_SRS_LIN.pdf

[7] Specification of LIN Interface
AUTOSAR_SWS_LINInterface.pdf

[8] Specification of Diagnostic Event Manager
AUTOSAR_SWS_ DiagnosticEventManager.pdf

[9] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[10] Specification of LIN Driver
AUTOSAR_SWS_LINDriver.pdf

[11] Specification of Communication Manager
AUTOSAR_SWS_COMManager.pdf

[12] Specification of Basic Software Mode Manager
AUTOSAR_SWS_BSWModeManager.pdf

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

3.2 Related standards and norms

[14] LIN Specification Package Revision 2.1, November 24, 2006
http://www.lin-subbus.org/

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [13] (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

The LinSM module can only be used as a LIN master in a LIN cluster. There is at most one instance of the LinSM in each ECU. If the underlying LIN Driver [10] supports multiple networks, 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.

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 3 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 (if enabled) – 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.

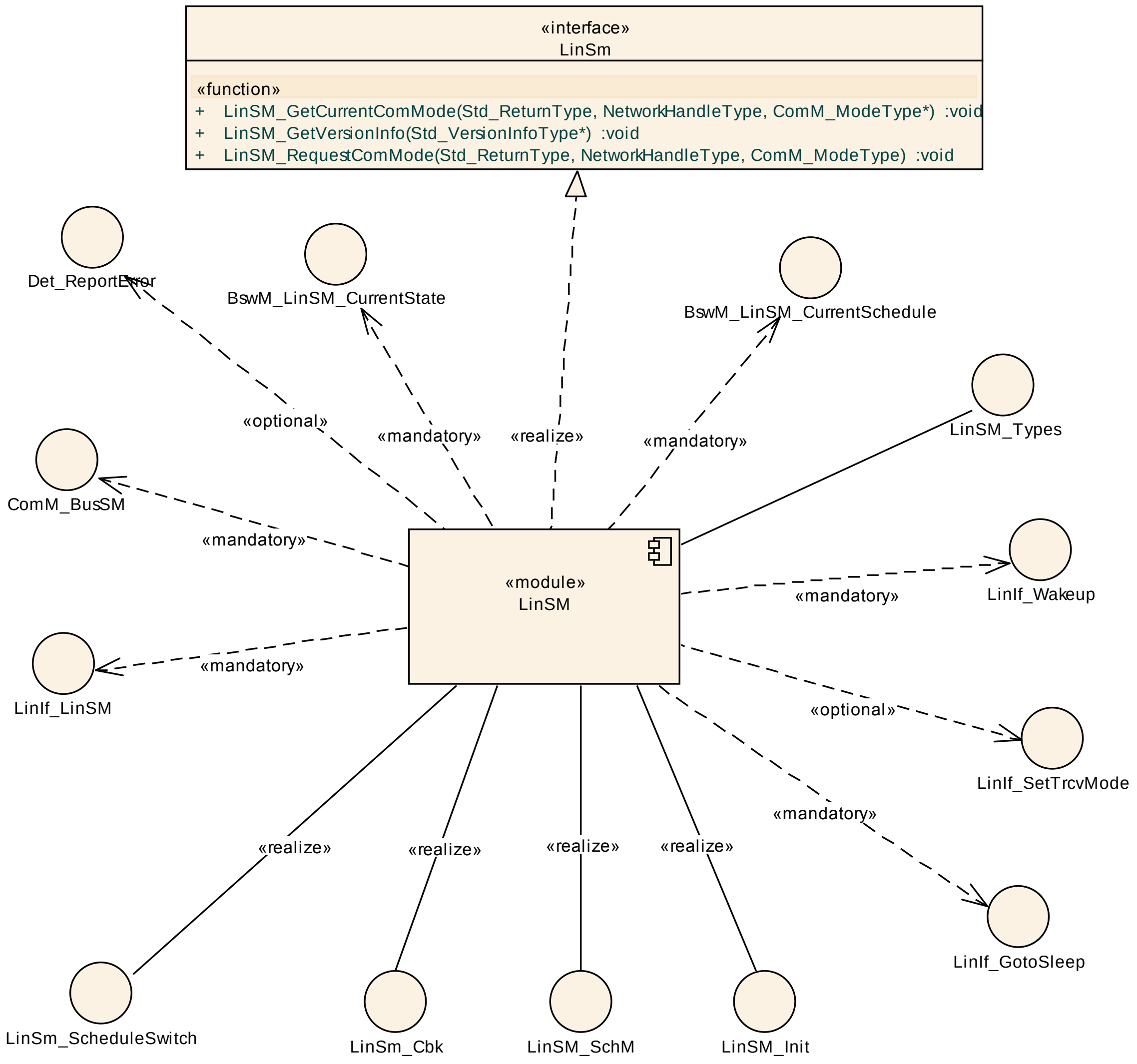

Figure 3 Dependencies to other modules

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)

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

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 [7] module:

- 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 module 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_GotoSleepConfirmation)
- Confirming a schedule change (LinSM_ScheduleRequestConfirmation)

[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

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

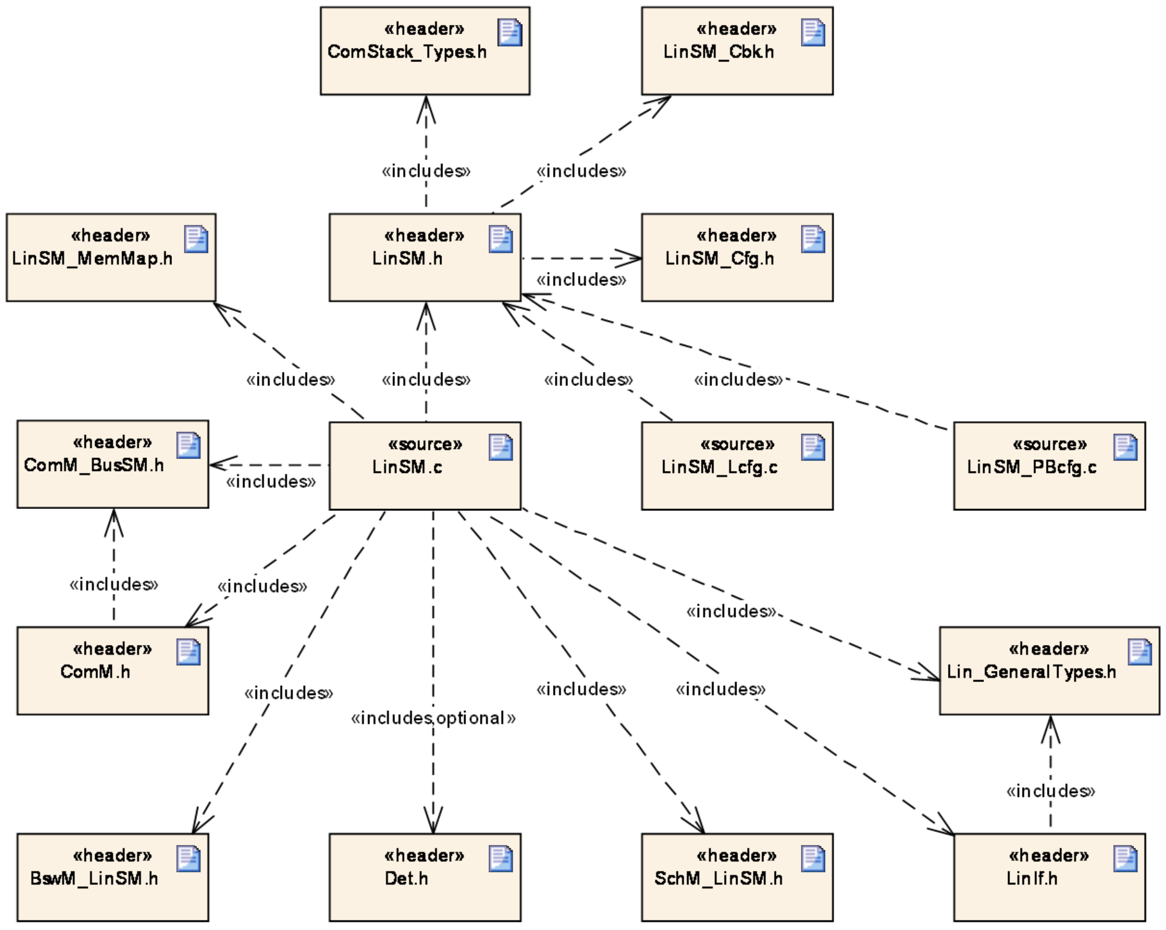

Figure 4 Header file structure

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

Following external header files shall be included:

[SWS_LinSM_00012] ⌈The LinSM module shall include the LinIf.h file. ⌋ ( )

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

[SWS_LinSM_00016] ⌈The LinSM module shall include the ComStack_Types.h⌋ ( )

[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_00306]⌈The LinSM module shall include the Lin_GeneralTypes.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)

[SWS_LinSM_00228]⌈The header file LinSM_Cfg.h shall contain references to the parameters of the c-source files LinSM_Lcfg.c and LinSM_PBcfg.c (see section 5.1.6 “Code file structure” in SWS_BSWGeneral) and shall contain pre-compile parameters, which are not declared as “const” parameter, but as defines.⌋ (TPS_STDT_00080)

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).

Requirement	Description	Satisfied by
-	-	SWS_LinSM_00002
-	-	SWS_LinSM_00005
-	-	SWS_LinSM_00012
-	-	SWS_LinSM_00013
-	-	SWS_LinSM_00016
-	-	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_00079
-	-	SWS_LinSM_00100
-	-	SWS_LinSM_00101
-	-	SWS_LinSM_00102
-	-	SWS_LinSM_00103
-	-	SWS_LinSM_00114
-	-	SWS_LinSM_00115
-	-	SWS_LinSM_00119
-	-	SWS_LinSM_00123
-	-	SWS_LinSM_00124
-	-	SWS_LinSM_00127

-	-	SWS_LinSM_00129
-	-	SWS_LinSM_00130
-	-	SWS_LinSM_00132
-	-	SWS_LinSM_00133
-	-	SWS_LinSM_00135
-	-	SWS_LinSM_00136
-	-	SWS_LinSM_00138
-	-	SWS_LinSM_00151
-	-	SWS_LinSM_00152
-	-	SWS_LinSM_00154
-	-	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_00167
-	-	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_00201
-	-	SWS_LinSM_00202
-	-	SWS_LinSM_00203
-	-	SWS_LinSM_00204
-	-	SWS_LinSM_00205

-	-	SWS_LinSM_00206
-	-	SWS_LinSM_00207
-	-	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_00225
-	-	SWS_LinSM_00226
-	-	SWS_LinSM_00227
-	-	SWS_LinSM_00301
-	-	SWS_LinSM_00302
-	-	SWS_LinSM_00304
-	-	SWS_LinSM_00305
-	-	SWS_LinSM_00306
-	-	SWS_LinSM_10208
-	-	SWS_LinSM_10209
-	-	SWS_LinSM_10211
BSW00443	-	SWS_LinSM_00211
BSW00444	-	SWS_LinSM_00211
BSW00445	-	SWS_LinSM_00211
BSW00446	-	SWS_LinSM_00211
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_00005	Modules of the ÂµC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_LinSM_00211
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_LinSM_00211
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_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_LinSM_00211

SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_LinSM_00211
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_LinSM_00073
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_LinSM_00211
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_LinSM_00211
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_LinSM_00211
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_LinSM_00211
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_LinSM_00211
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_LinSM_00211
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_LinSM_00211
SRS_BSW_00343	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	SWS_LinSM_00211
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_LinSM_00155
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_LinSM_00211
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_LinSM_00211
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_00375	Basic Software Modules shall report wake-up reasons	SWS_LinSM_00211
SRS_BSW_00376	-	SWS_LinSM_00156
SRS_BSW_00384	The Basic Software Module specifications shall specify at least in	SWS_LinSM_00001, SWS_LinSM_00085,

	the description which other modules they require	SWS_LinSM_00086, SWS_LinSM_00105, SWS_LinSM_00196
SRS_BSW_00399	Parameter-sets shall be located in a separate segment and shall be loaded after the code	SWS_LinSM_00211
SRS_BSW_00400	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	SWS_LinSM_00211
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_LinSM_00211
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_LinSM_00211
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
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_LinSM_00211
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_LinSM_00211
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_LinSM_00211
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_LinSM_00211
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_LinSM_00211
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_LinSM_00211
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_LinSM_00211
SRS_BSW_00436	-	SWS_LinSM_00211
SRS_BSW_00437	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	SWS_LinSM_00211

SRS_BSW_00438	Configuration data shall be defined in a structure	SWS_LinSM_00211
SRS_BSW_00439	Enable BSW modules to handle interrupts	SWS_LinSM_00211
SRS_BSW_00442	{OBSOLETE} The AUTOSAR architecture shall support standardized debugging and tracing features	SWS_LinSM_00188, SWS_LinSM_00189
SRS_Lin_01560	If a wakeup occurs during transition to sleep-mode, this channel shall go back to the running mode	SWS_LinSM_00211
SRS_Lin_01577	It shall be compatible to LIN protocol specification	SWS_LinSM_00211
SRS_Lin_01590	The node configuration of LIN slaves shall only be done via defined schedule table(s).	SWS_LinSM_00211
TPS_STDT_00080	Representation of specification items in AUTOSAR documents	SWS_LinSM_00228

7 Functional specification

This chapter specifies the requirements on the module LinSM module. 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 networks
- Handle schedule change requests
- 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 of the master in the LIN 2.1 specification. The master behavior provided by the LinSM module will allow the reuse of existing slaves conforming to the LIN 1.2, 1.3, 2.0 and 2.1 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 4 shows a simplified version of the requirements below, the intention of the figure 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. ⌋ ( )

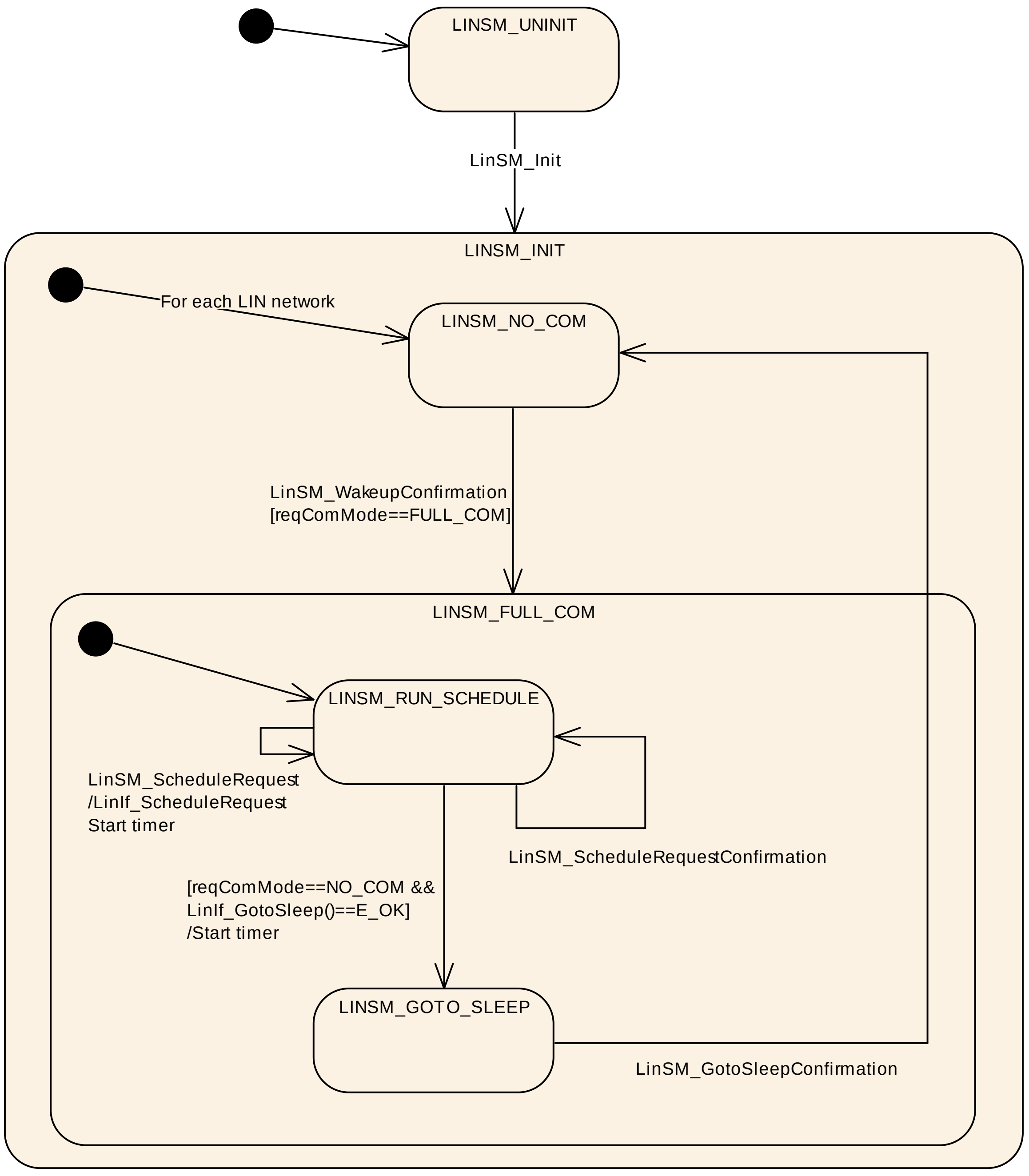

Figure 5: LinSM State Machine

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 schedule tables are active and all communication is made. 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. ⌋ ( )

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.

[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 LinSMTransceiverPassiveMode is configured for the channel. ⌋ （ ）

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

7.1.5 Goto sleep

The LinSM will request the goto-sleep command to be sent on the LIN bus, and notify the BswM and ComM of the new state when entering no communication mode. 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 LinIf when the state LINSM_FULL_COM and the sub-state LINSM_RUN_SCHEDULE is active.⌋ ( )

[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_GotoSleep 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_RequestComMode 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_CurrentState are called). ⌋ ( )

[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 sub-state LINSM_GOTOSLEEP.⌋ ( )

7.1.6 Changing schedule table

[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_00167] ⌈If LinIf_ScheduleRequest returns with E_OK the LinSM_ScheduleRequest shall return with E_OK. ⌋ ( )

[SWS_LinSM_00168] ⌈If LinIf_ScheduleRequest returns with E_NOT_OK the LinSM_ScheduleRequest shall return with E_NOT_OK.

The caller of LinSM_ScheduleRequest will assume that LinSM module calls the callback: ⌋ ( )

[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 slave node transmits the wakeup signal on the netowork. 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 slave 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.⌋ ( )

[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_RequestComMode shall return E_NOT_OK directly with no further action⌋ ( )

7.1.8 Timeout of requests

After calling LinIf_GotoSleep, LinIf_Wakeup or LinIf_ScheduleRequest the LinSM module is waiting for the LinIf module 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 module 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.

[SWS_LinSM_00175] ⌈There shall be request timers for each network. One network shall be independent of another network. ⌋ ( )

[SWS_LinSM_00162] ⌈The handling (countdown and expiration) of the all request timers used by the LinSM module shall be made done in the LinSM_MainFunction. ⌋ ( )

[SWS_LinSM_00159] ⌈All request timers shall have a time that is a divisible by the LinSM_MainFunction (i.e. LinSM_MainFunction period * m; m integer >0) ⌋ ( )

[SWS_LinSM_00100] ⌈Before the LinSM calls the LinIf_GotoSleep, LinIf_Wakeup or LinIf_ScheduleRequest is called, the LinSM module shall start a timer. ⌋ ( )

[SWS_LinSM_00101] ⌈When a timer expires, i.e. greater than the configuration parameter LinSMConfirmationTimeout, a timeout occurs. ⌋ ( )

[SWS_LinSM_00154] ⌈If the LinIf module calls the confirmation callback before the timeout occurs, the active timer shall stop, so that the timeout will not occur. ⌋ ( )

[SWS_LinSM_00102] ⌈ if LinSMDevErrorDetect is enabled: When a timeout occurs, the error code LINSM_E_CONFIRMATION_TIMEOUT shall be reported to the DET module. ⌋ ( )

[SWS_LinSM_00170] ⌈If request timer elapses (i.e. module LinIf is not notifying within the timeout) and the maximum number of retries have been reached, in the case of a LinIf_Wakeup request, the LinSM module shall notify ComM module with same state. ⌋ ( )

[SWS_LinSM_00215] ⌈If request timer elapses (i.e. module LinIf is not notifying within the timeout)) and the maximum number of retries have been reached, in the case of a LinIf_Wakeup request, the LinSM module shall notify BswM module with same state. ⌋ ( )

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

[SWS_LinSM_00103] ⌈If the configuration parameter LinSMConfirmationTimeout is set to zero the timer is not used, and hence a timeout cannot occur. This means that requirements **[SWS_LinSM_00102]**, **[SWS_LinSM_00170]** and **[SWS_LinSM_00215]** will not happen. ⌋ ( )

[SWS_LinSM_00172] ⌈If LinIf module calls the confirmation callback after the timer has elapsed, no further notification shall be made to the ComM modules, i.e. the confirmation is ignored. ⌋ ( )

[SWS_LinSM_00304] ⌈If request timout has occurred for LinIf_Wakeup and the maximum retries (LinSMModeRequestRepetitionMax) have not been reached, the LinIf_Wakeup request will be sent again. ⌋ ( )

[SWS_LinSM_00305] ⌈The timer elapses for LinIf_Wakeup only, in the sense of **[SWS_LinSM_00170]** and **[SWS_LinSM_00215]**, if the maximum number of retries (LinSMModeRequestRepetitionMax) has been reached. ⌋ ( )

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 LinSM 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 LinSM 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

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

7.3.1 Development Errors

[SWS_LinSM_00053] ⌈

Type or error	***Related error code***	***Value [hex]***
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
Timeout of the callbacks	LINSM_E_CONFIRMATION_TIMEOUT	0x50

from LinIf		
Init function failed	LINSM_E_INIT_FAILED	0x60

⌋ ( )

7.3.2 Runtime Errors

[SWS_LinSM_00224] ⌈

Type or error	***Related error code***	***Value [hex]***
Not possible to perform the request, e.g. not initialized.	E_NOT_OK	

⌋ ( )

7.3.3 Transient Faults

[SWS_LinSM_00225] ⌈
There are no extended transient faults.

⌋ ( )

7.3.4 Production Errors

[SWS_LinSM_00226] ⌈
There are no production errors.
⌋ ( )

7.3.5 Extended Production Errors

[SWS_LinSM_00227] ⌈
There are no extended production errors.
⌋ ( )

7.4 Error detection

For details refer to the chapters 7.2 “Error classification” & 7.3 “Error Detection” in *SWS_BSWGeneral.*

7.5 Error notification

For details refer to the chapters 7.4 “Error notification” in *SWS_BSWGeneral*

7.6 Debugging

Note: The debugging specifciations are obsolete and will be removed from the standard in an upcoming release.

To allow standardized debugging of the LinSM module, following requirements applies:

[SWS_LinSM_00188] {OBSOLETE} ⌈The state of the state-machine (LINSM_UNINIT and LINSM_INIT) shall be accessible for debugging. ⌋ (SRS_BSW_00442)

[SWS_LinSM_00189] {OBSOLETE} ⌈The state of the sub-state-machine (LINSM_NO_COM and LINSM_FULL_COM) shall be accessible for debugging. ⌋ (SRS_BSW_00442)

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].

Following types are used by the LinSM module:

[SWS_LinSM_00219] ⌈

Module	*Imported Type*
ComM	ComM_ModeType
ComStack_Types	NetworkHandleType
LinIf	LinIf_SchHandleType
LinTrcv	LinTrcv_TrcvModeType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋ ( )

8.2 Type definitions

Following types are defined by the LinSM module:

8.2.1 LinSM_ModeType

[SWS_LinSM_00220] ⌈

Name:	`LinSM_ModeType`		
Type:	`uint8`		
Range:	`LINSM_FULL_COM`	`1`	Full communication
	`LINSM_NO_COM`	`2`	No communication
Description:	Type used to report the current mode to the BswM		

⌋ ( )

8.2.2 LinSM_ConfigType

[SWS_LinSM_00221] ⌈

Name:	`LinSM_ConfigType`	
Type:	`Structure`	
Range:	`implementation specific`	--

Description:	Data structure type for the post-build configuration parameters.

⌋ ( )

8.3 LinSM API

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

8.3.1 LinSM_Init

[SWS_LinSM_00155] ⌈

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.	

⌋ (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

[SWS_LinSM_00113] ⌈

Service name:	LinSM_ScheduleRequest	

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

Service ID[hex]:	0x10	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	network	Identification of the LIN channel
	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) * Sub-state is not LINSM_FULL_COM
Description:	The upper layer requests a schedule table to be changed on one LIN network.	

⌋ (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_00163] ⌈If the function LinSM_ScheduleRequest is called and another request is in process on the same network, the LinSM_ScheduleRequest shall return directly with E_NOT_OK. ⌋ ( )

[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. ⌋ ( )

8.3.3 LinSM_GetVersionInfo

[SWS_LinSM_00117] ⌈

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:	--	

⌋ (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] ⌈

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.	

⌋ (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] ⌈

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.	

⌋ (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 7.1 refers to this stored request as reqComMode. ⌋ ( )

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] ⌈

Service name:	LinSM_MainFunction

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

Service ID[hex]:	0x30
Description:	Periodic function that runs the timers of different request timeouts

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). ⌋ (SRS_BSW_00373, SRS_BSW_00376)

[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 7.1.8) ⌋ ( )

8.5 LinSM callbacks

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

8.5.1 LinSM_ScheduleRequestConfirmation

[SWS_LinSM_00129] ⌈

Service name:	LinSM_ScheduleRequestConfirmation	
Syntax:	`void LinSM_ScheduleRequestConfirmation( NetworkHandleType network, LinIf_SchHandleType schedule )`	
Service ID[hex]:	0x20	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	network	Identification of the LIN channel
	schedule	Pointer to the new active Schedule 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.	

⌋ ( )

[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)

8.5.2 LinSM_GotoSleepConfirmation

[SWS_LinSM_00135] ⌈

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.	

⌋ ( )

[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)

8.5.3 LinSM_WakeupConfirmation

[SWS_LinSM_00132] ⌈

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.	

⌋ ( )

[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.

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 and BswM.
LinIf_GotoSleep	Initiates a transition into the Sleep Mode on the selected channel.
LinIf_ScheduleRequest	Requests a schedule table to be executed.
LinIf_Wakeup	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] ⌈

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

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

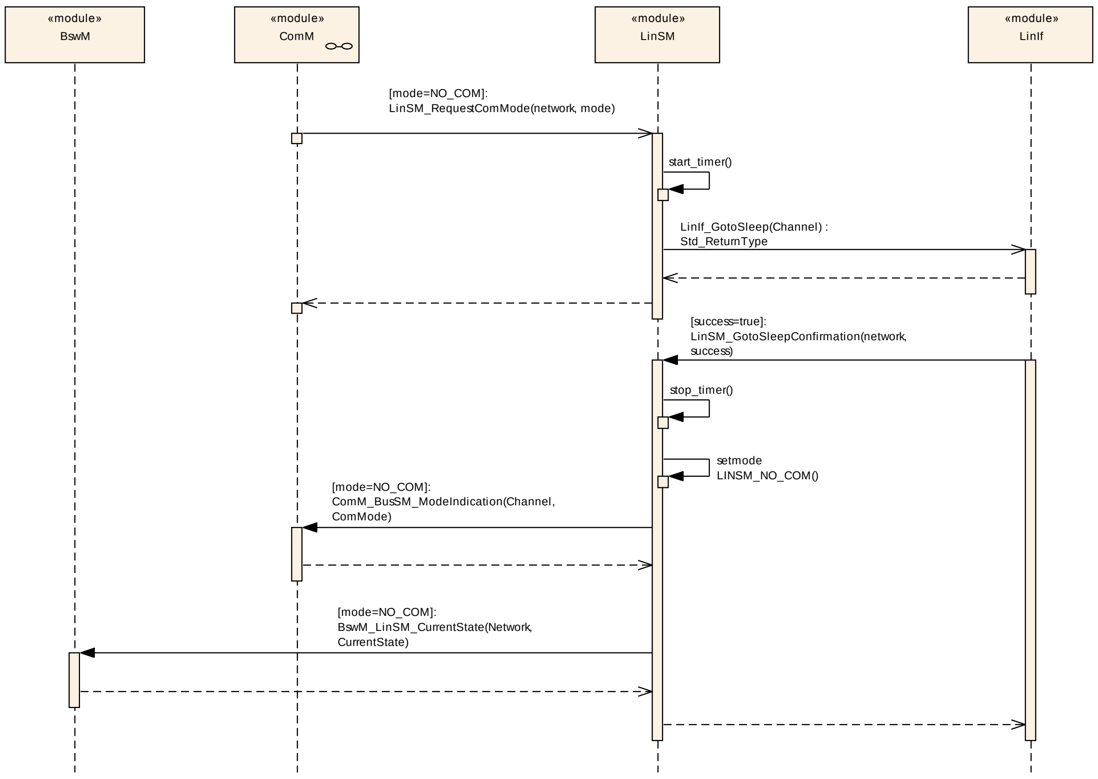
«module» BswM
«module» ComM
«module» LinSM
«module» LinIf
[mode=NO_COM]: LinSM_RequestComMode(network, mode)
start_timer()
LinIf_GotoSleep(Channel) : Std_ReturnType
[success=true]: LinSM_GotoSleepConfirmation(network, success)
stop_timer()
setmode LINSM_NO_COM()
[mode=NO_COM]: ComM_BusSM_ModeIndication(Channel,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

Figure 6 - Goto-sleep–command process

9.2 Internal wake up

The Figure 7 shows the internal wakeup. A wakeup is requested from module above the LinSM.

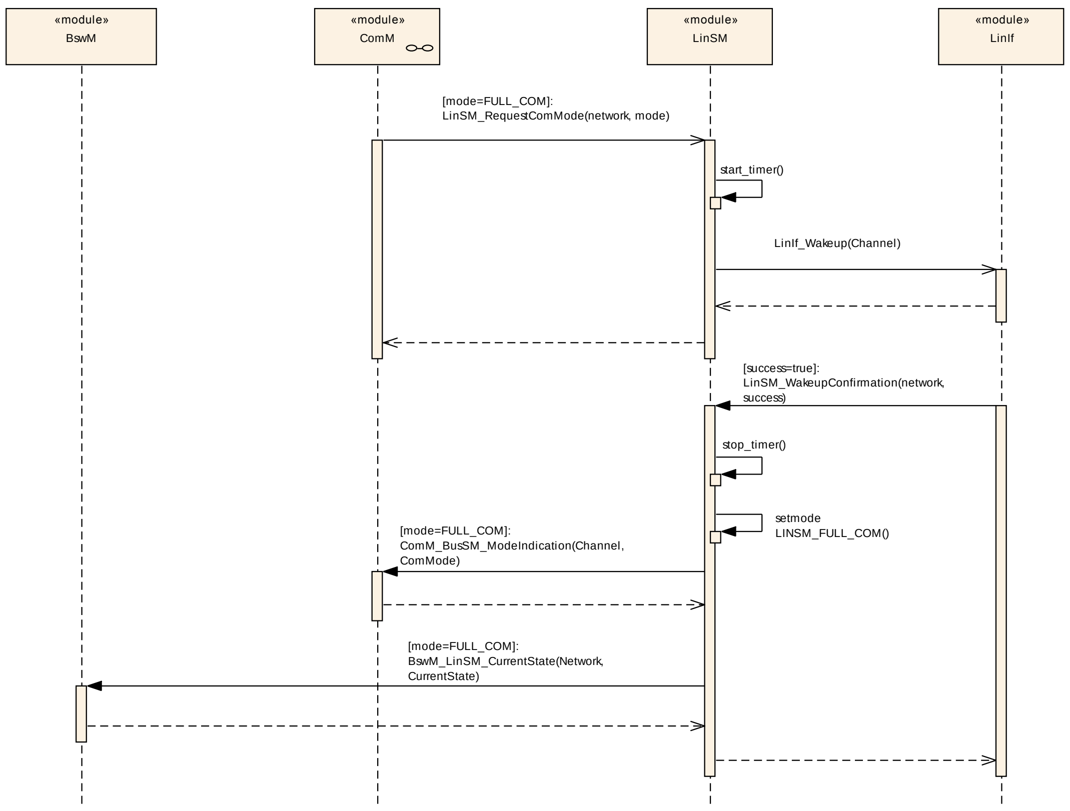

Figure 7 - Internal wake up

9.3 Schedule switch

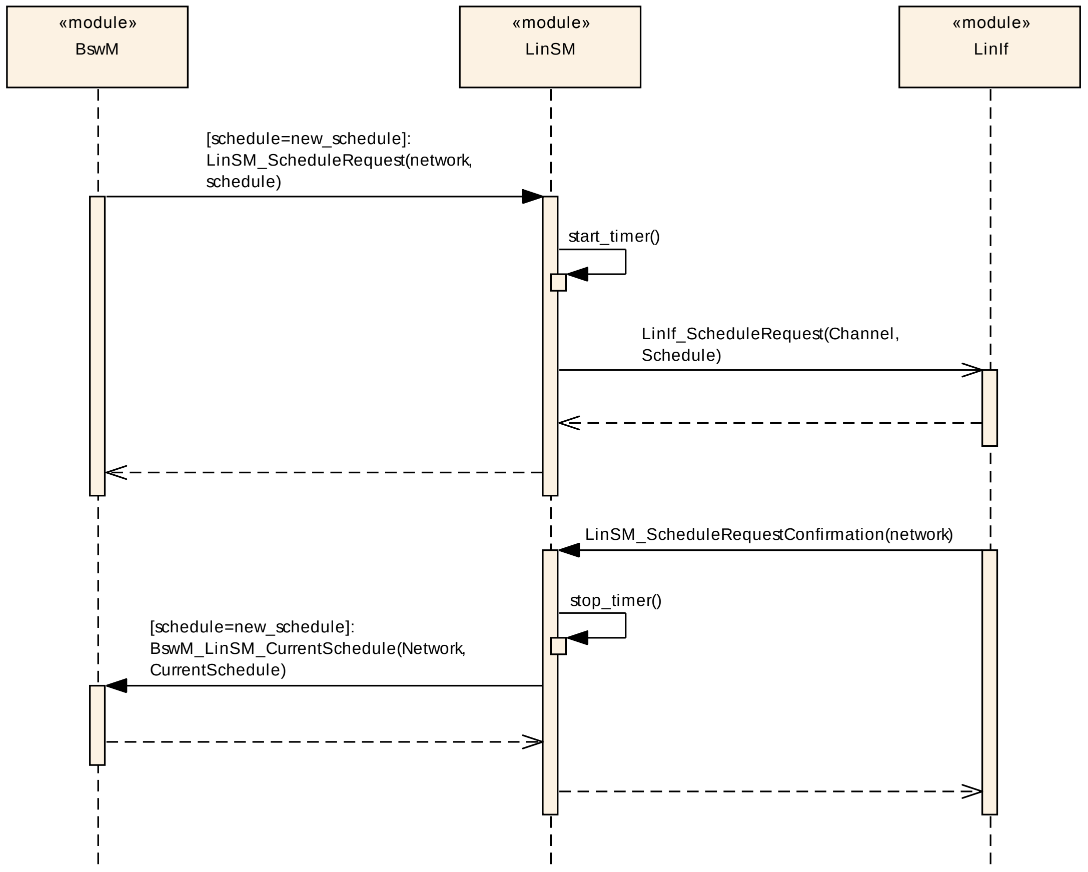

Figure 8 shows the use-cases of switching the schedule table when the LinSM accepts the request.

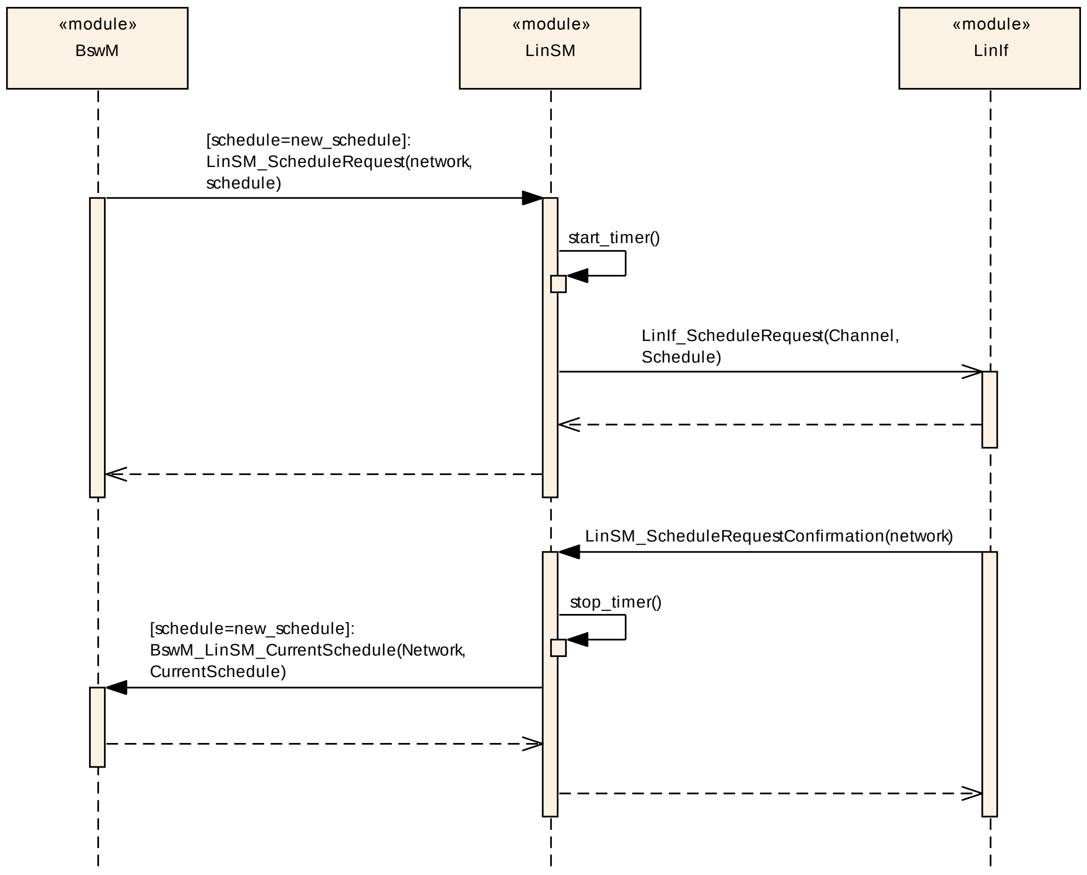

Figure 8: Schedule Table switch

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

The 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 LinSM.

[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)

10.2.2 Variants

Following configuration variants are defined for the LinSM module.

10.2.2.1 VARIANT-PRE-COMPILE

In the the variant VARIANT-PRE-COMPILE 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 VARIANT-LINK-TIME

The variant VARIANT-LINK-TIME shall include all configuration options of the variant VARIANT-PRE-COMPILE. Additionally, all parameters that are marked 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.

10.2.2.3 VARIANT-POST_BUILD

The variant VARIANT-POST-BUILD shall include all configuration options of the variant VARIANT-LINK-TIME. Additionally, all parameters that are marked as post-build configurable shall be configurable post build for example by flashing configuration data.

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.
Post-Build Variant Support	true

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
Description	This container contains the configuration parameters and sub containers of the AUTOSAR LinSm module.
Configuration Parameters	

SWS Item	**ECUC_LinSM_00208 :**		
Name	LinSMModeRequestRepetitionMax		
Description	Specifies the maximal amount of mode request repetitions without a respective mode indication from the LinIf module until the LinSM module reports a Default Error to the Det and tries to go back to no communication.		
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		

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
Description	Describes each LIN channel the LinSM is connected to.
Configuration Parameters	

SWS Item	ECUC_LinSM_00144 :		
Name	LinSMConfirmationTimeout		
Description	Timeout in seconds for the goto sleep, wakeup and schedule request 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	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_00202 :		
Name	LinSMTransceiverPassiveMode		
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 :
Name	LinSMComMNetworkHandleRef
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	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 LinSMGeneral

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

SWS Item	**ECUC_LinSM_00206 :**		
Name	LinSMDevErrorDetect		
Description	Switches the Default Error Tracer (Det) detection and notification ON or OFF. • true: enabled (ON). • false: disabled (OFF).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinSM_00141 :**		
Name	LinSMMainProcessingPeriod		
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 :**
Name	LinSMVersionInfoApi

Description	Switches the LinSM_GetVersionInfo function ON or OFF.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.3.5 LinSMSchedule

SWS Item	**ECUC_LinSM_00146 :**
Container Name	LinSMSchedule
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	**ECUC_LinSM_00001 :**		
Name	LinSMScheduleIndex		
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		

SWS Item	**ECUC_LinSM_00149 :**		
Name	LinSMScheduleIndexRef		
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.*

11 Not applicable requirements

[SWS_LinSM_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_00436, SRS_BSW_00439, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00331, BSW00443, BSW00444, BSW00445, BSW00446, 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)