

|                                   |                             |
|-----------------------------------|-----------------------------|
| <b>Document Title</b>             | Specification of LIN Driver |
| <b>Document Owner</b>             | AUTOSAR                     |
| <b>Document Responsibility</b>    | AUTOSAR                     |
| <b>Document Identification No</b> | 72                          |
| <b>Document Status</b>            | published                   |
| <b>Part of AUTOSAR Standard</b>   | Classic Platform            |
| <b>Part of Standard Release</b>   | R21-11                      |

| <b>Document Change History</b> |                |                                  |  |
|--------------------------------|----------------|----------------------------------|--|
| <b>Date</b>                    | <b>Release</b> | <b>Changed by</b>                | <b>Change Description</b>  |
| 2021-11-25                     | R21-11         | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>• Clarification of configuration parameter LinChannelWakeupSupport</li> <li>• Cleanup of Error classification chapter</li> <li>• Header file for EcuM_CheckWakeup changed</li> </ul>  |
| 2020-11-30                     | R20-11         | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>• Modified SWS_Lin_00266, Lin_GeneralTypes removed</li> <li>• Editorial change</li> </ul>   |
| 2019-11-28                     | R19-11         | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>• MCALMulticoreDistribution (CONC_639)</li> <li>• Changed Document Status from Final to published</li> </ul>  |
| 2018-10-31                     | 4.4.0          | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>• LIN Slave support (CONC_634)</li> <li>• MCALMulticoreDistribution (CONC_639) as DRAFT</li> <li>• Replace references to LIN 2.1 by ISO 17987:2016 (with no functional modification)</li> <li>• Header file cleanup</li> <li>• Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation</li> </ul> |
| 2017-12-08                     | 4.3.1          | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>• Minor corrections / clarifications / editorial changes</li> <li>• Resolve inconsistency on channel state upon initialization</li> </ul>   |

| <b>Document Change History</b> |                |                                  |  |
|--------------------------------|----------------|----------------------------------|--|
| <b>Date</b>                    | <b>Release</b> | <b>Changed by</b>                | <b>Change Description</b>  |
| 2016-11-30                     | 4.3.0          | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>• Updated tracing information</li> <li>• Removed chapter 'Variants'</li> </ul>  |
| 2015-07-31                     | 4.2.2          | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>• Chapter 6 "Requirements traceability" clean up</li> <li>• Reference to DET are named as "Default" Error Tracer instead of "Development" Error Tracer</li> <li>• Dependency on Module DET listed in Chapter 5 is linked to SWS_Lin_00048 instead of SWS_Lin_00052</li> </ul>                   |
| 2014-10-31                     | 4.2.1          | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>• Replaced SWS_Lin_00064 with SWS_Lin_00268</li> </ul>  |
| 2014-03-31                     | 4.1.3          | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>• Removed SWS_Lin_00243.</li> <li>• Modified SWS_Lin_00237, SWS_Lin_00058, SWS_Lin_00266, SWS_Lin_00255, SWS_Lin_00256, SWS_Lin_00258, SWS_Lin_00259, SWS_Lin_00260.</li> <li>• Updated Figure 7-1.</li> <li>• Removed references to SWS_Lin_00073 and SWS_Lin_00034 from chapter 6.</li> </ul> |

| <b>Document Change History</b> |                |                                  |  |
|--------------------------------|----------------|----------------------------------|--|
| <b>Date</b>                    | <b>Release</b> | <b>Changed by</b>                | <b>Change Description</b>  |
| 2013-10-31                     | 4.1.2          | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>• Removed outdated SWS_Lin_00109, SWS_Lin_00136 and SWS_Lin_00132.</li> <li>• Import of SWS_Lin_184 from R3.2.2</li> <li>• Wake-up LIN Functionality updated</li> <li>• New API Lin_WakeupInternal added. See chapter 8.3.2.5</li> <li>• Added the following type definition (with SWS item ID) to chapter 8:                             <ul style="list-style-type: none"> <li>• Lin_FrameCsModelType</li> <li>• Lin_FrameDType</li> <li>• Lin_FramePidType</li> <li>• Lin_FrameResponseType</li> <li>• Lin_PduType</li> <li>• Lin_StatusType</li> </ul> </li> <li>• Editorial changes</li> <li>• Removed chapter(s) on change documentation</li> </ul> |
| 2013-03-15                     | 4.1.1          | AUTOSAR<br>Administration        | <ul style="list-style-type: none"> <li>• Specified LIN_E_TIMEOUT as production error</li> <li>• Shifted all types used by other modules to Lin_GeneralTypes.h</li> <li>• Revised configuration container LinDemEventParamterRefs</li> <li>• Some minor updates</li> </ul>  |
| 2011-12-22                     | 4.0.3          | AUTOSAR<br>Administration        | <ul style="list-style-type: none"> <li>• Changed error reporting</li> <li>• Improved wake-up handling</li> <li>• Corrected call of Lin_Init</li> </ul>   |
| 2010-09-30                     | 3.1.5          | AUTOSAR<br>Administration        | <ul style="list-style-type: none"> <li>• Introduce Lin_GeneralTypes.h</li> <li>• Add missing DET error code (NULL pointer error)</li> <li>• Remove instance ID from Lin_GetVersionInfo API</li> <li>• Correct naming of “WakeUp” to “Wakeup”</li> <li>• Further maintenance for R4.0.2: see chapter 12</li> </ul>  |

| <b>Document Change History</b> |                |                        |   |
|--------------------------------|----------------|------------------------|---|
| <b>Date</b>                    | <b>Release</b> | <b>Changed by</b>      | <b>Change Description</b>   |
| 2010-02-02                     | 3.1.4          | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• Support of advanced LIN controllers (combination of Lin_SendHeader and Lin_SendResponse to Lin_SendFrame)</li> <li>• Integrating LIN channel initialization in LIN module initialization</li> <li>• Further maintenance for R4.0: see chapter 11</li> <li>• Legal disclaimer revised</li> </ul>  |
| 2008-08-13                     | 3.1.1          | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• Legal disclaimer revised</li> </ul>  |
| 2007-12-21                     | 3.0.1          | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• Editorial Changes</li> <li>• Tables generated in Chapter 8 and 10</li> <li>• Document meta information extended</li> <li>• Small layout adaptations made</li> </ul>  |
| 2007-01-24                     | 2.1.15         | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• Lin Transceiver Wake Up validation function added</li> <li>• Incorporate Feedback from Validator2</li> <li>• Updated Chapter 10.2 according to the Specification of ECU Configuration Parameters</li> <li>• Legal disclaimer revised</li> <li>• Release Notes added</li> <li>• “Advice for users” revised</li> <li>• “Revision Information” added</li> </ul> |
| 2006-05-16                     | 2.0            | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• Initial release</li> </ul>   |

## Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Table of Contents

|       |   |    |
|-------|---|----|
| 1     | Introduction and functional overview .....  | 8  |
| 1.1   | Scope .....                                 | 8  |
| 1.2   | Architectural overview .....                | 8  |
| 2     | Acronyms, abbreviations and glossary .....  | 10 |
| 2.1   | Acronyms and abbreviations .....            | 10 |
| 2.2   | Glossary .....                              | 10 |
| 2.3   | LIN hardware unit classification .....      | 11 |
| 3     | Related documentation .....                 | 12 |
| 3.1   | Input documents .....                       | 12 |
| 3.2   | Related standards and norms .....           | 12 |
| 3.3   | Related specification .....                 | 13 |
| 4     | Constraints and assumptions .....           | 14 |
| 4.1   | Limitations .....                           | 14 |
| 4.2   | Applicability to car domains .....          | 14 |
| 5     | Dependencies to other modules .....         | 15 |
| 5.1   | File structure .....                        | 15 |
| 5.1.1 | Code file structure .....                   | 15 |
| 5.1.2 | Header file structure .....                 | 15 |
| 6     | Requirements traceability .....             | 17 |
| 7     | Functional specification .....              | 22 |
| 7.1   | General Requirements .....                  | 22 |
| 7.2   | Version Check .....                         | 23 |
| 7.2.1 | Requirements .....                          | 23 |
| 7.3   | LIN driver and Channel Initialization ..... | 23 |
| 7.3.1 | Background & Rationale .....                | 23 |
| 7.3.2 | Requirements .....                          | 23 |
| 7.3.3 | State diagrams .....                        | 24 |
| 7.4   | Frame processing .....                      | 27 |
| 7.4.1 | Background & Rationale .....                | 27 |
| 7.4.2 | Requirements .....                          | 27 |
| 7.4.3 | Data Consistency .....                      | 30 |
| 7.4.4 | Data byte mapping .....                     | 31 |
| 7.5   | Sleep and wake-up functionality .....       | 32 |
| 7.5.1 | Background & Rationale .....                | 32 |
| 7.5.2 | Requirements .....                          | 33 |
| 7.6   | Error classification .....                  | 33 |
| 7.6.1 | Development Errors .....                    | 34 |
| 7.6.2 | Runtime Errors .....                        | 34 |
| 7.6.3 | Transient Faults .....                      | 34 |
| 7.6.4 | Production Errors .....                     | 34 |
| 7.6.5 | Extended Production Errors .....            | 35 |

|        |   |    |
|--------|---|----|
| 8      | API specification.....                                  | 36 |
| 8.1    | Imported types .....                                    | 36 |
| 8.2    | Type definitions .....                                  | 36 |
| 8.2.1  | Lin_ConfigType .....                                    | 36 |
| 8.2.2  | Lin_FramePidType .....                                  | 37 |
| 8.2.3  | Lin_FrameCsModelType .....                              | 37 |
| 8.2.4  | Lin_FrameResponseType .....                             | 37 |
| 8.2.5  | Lin_FrameDIType .....                                   | 38 |
| 8.2.6  | Lin_PduType .....                                       | 38 |
| 8.2.7  | Lin_StatusType .....                                    | 39 |
| 8.2.8  | Lin_SlaveErrorType.....                                 | 40 |
| 8.3    | Function definitions .....                              | 41 |
| 8.3.1  | Services affecting the complete LIN hardware unit ..... | 41 |
| 8.3.2  | Services affecting a single LIN channel .....           | 45 |
| 8.4    | Call-back notifications .....                           | 54 |
| 8.5    | Scheduled functions.....                                | 54 |
| 8.6    | Expected Interfaces .....                               | 54 |
| 8.6.1  | Mandatory Interfaces.....                               | 54 |
| 8.6.2  | Optional Interfaces .....                               | 54 |
| 8.6.3  | Configurable interfaces .....                           | 55 |
| 9      | Sequence diagrams .....                                 | 56 |
| 9.1    | Receiving a LIN Frame .....                             | 56 |
| 9.1.1  | LIN Master .....  | 56 |
| 9.1.2  | LIN Slave .....   | 57 |
| 10     | Configuration specification .....                       | 58 |
| 10.1   | How to read this chapter.....                           | 58 |
| 10.2   | Containers and configuration parameters .....           | 58 |
| 10.2.1 | Lin.....  | 59 |
| 10.2.2 | LinGeneral.....   | 60 |
| 10.2.3 | LinChannel .....  | 61 |
| 10.2.4 | LinGlobalConfig .....                                   | 64 |
| 10.2.5 | LinDemEventParameterRefs .....                          | 64 |
| 10.3   | Published Information .....                             | 65 |
| 11     | Not applicable requirements .....                       | 66 |

# 1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module LIN driver.

## 1.1 Scope

The base for this document is the ISO 17987 specifications [16]. It is assumed that the reader is familiar with this specification. This document will not describe ISO 17987 LIN functionality again.

The LIN driver applies to ISO 17987 master and slave nodes. The LIN implementation in AUTOSAR deviates from the ISO 17987 specifications as described in this specification of LIN driver, but there will be no change in the behavior on the LIN bus. It is the intention to be able to reuse all existing LIN nodes together with the AUTOSAR LIN implementation (i.e. the LIN driver).

**[SWS\_Lin\_00063]** †It is intended to support the complete range of LIN hardware from a simple SCI/UART to a complex LIN hardware controller. Using a SW-UART implementation is out of the scope. For a closer description of the LIN hardware unit, see chapter 2.3. †(SRS\_Lin\_01547)

## 1.2 Architectural overview

The LIN driver is part of the microcontroller abstraction layer (MCAL), performs the hardware access and offers a hardware independent API to the upper layer. The only upper layer, which has access to the LIN driver, is the LIN Interface.

A LIN driver can support more than one channel. This means that the LIN driver can handle one or more LIN channels as long as they are belonging to the same LIN hardware unit.

In the example below three different LIN drivers are connected to the LIN interface. However, one LIN driver is the most common configuration.



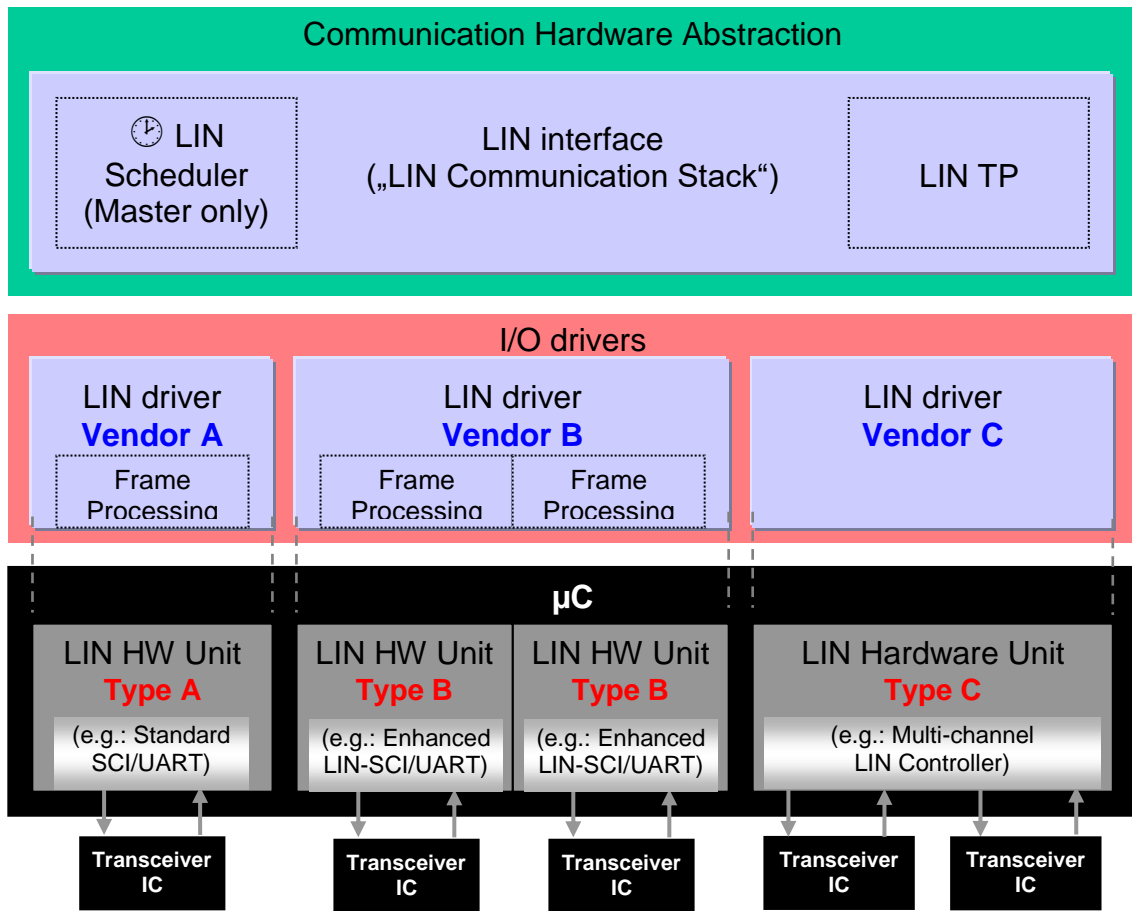


Figure 10-1: Overview LIN Software Architecture Layering

## 2 Acronyms, abbreviations and glossary

### 2.1 Acronyms and abbreviations

Acronyms, abbreviations and definitions that have a local scope for the LIN driver and therefore are not contained in the AUTOSAR glossary must appear here.

| <b>Acronym:</b> | <b>Description:</b>  |
|-----------------|--|
| AUTOSAR         | Automotive Open System Architecture                                    |
| COM             | Communication  |
| ECU             | Electronic Control Unit  |
| EcuM            | ECU Manager  |
| DEM             | Diagnostic Event Manager   |
| DET             | Default Error Tracer   |
| ISR             | Interrupt Service Routine  |
| LIN             | Local Interconnect Network (as defined by [16])                        |
| MCAL            | MicroController Abstraction Layer                                      |
| MCU             | Micro Controller Unit  |
| OS              | Operating System   |
| PDU             | Protocol Data Unit. Consists of Identifier, data length and Data (SDU) |
| PID             | Protected ID (as defined by [16])                                      |
| PLL             | Phase-Locked Loop  |
| RAM             | Random Access Memory   |
| RX              | Reception  |
| SCI             | Serial Communication Interface   |
| SDU             | Service Data Unit. Data that is transported inside the PDU             |
| SFR             | Special Function Register  |
| SPAL            | Standard Peripheral Abstraction Layer                                  |
| SRS             | Software Requirement Specification                                     |
| SW              | Software   |
| SWS             | Software Specification   |
| TP              | Transport Layer  |
| TX              | Transmission   |
| UART            | Universal Asynchronous Receiver Transmitter                            |
| XML             | Extensible Markup Language   |

| <b>Abbreviation</b> | <b>Description:</b> |
|---------------------|---------------------|
| Id                  | Identifier          |

### 2.2 Glossary

Besides AUTOSAR terminology this document also uses terms defined in the ISO 17987 specifications [16], e.g. LIN frame, header and message.

| <b>Glossary:</b> | <b>Description:</b>   |
|------------------|---|
| enumeration      | This can be in "C" programming language an enum or a #define.   |
| LIN channel      | The LIN channel entity interlinks the ECUs of a LIN cluster physically: An ECU is part of a LIN cluster if it contains one LIN controller that is connected to one LIN channel of the LIN cluster. An ECU is allowed to connect to a particular LIN cluster through one channel only. |
| LIN cluster      | As defined by [16]: "A cluster is the LIN bus wire plus all the nodes."   |

|                     |   |
|---------------------|---|
| LIN controller      | A dedicated LIN hardware with a build Frame processing state machine. A hardware which is capable to connect to several LIN clusters is treated as several LIN controllers. |
| LIN frame           | As defined by [16]: “All information is sent packed as frames; a frame consist of the header and a response.”   |
| LIN frame processor | Frame processing implies the complete LIN frame handling. Implementation could be achieved as software emulated solution or with a dedicated LIN controller.                |
| LIN hardware unit   | A LIN hardware unit may drive one or multiple LIN channels to control one or multiple LIN clusters.   |
| LIN header          | As defined by [16]: “A header is the first part of a frame; it is always sent by the master.”   |
| LIN node            | As defined by [16]: “Loosely speaking, a node is an ECU. However, a single ECU may be connected to multiple LIN clusters.”  |
| LIN response        | As defined by [16]: “A LIN frame consists of a header and a response. Also called a Frame response.”  |

## 2.3 LIN hardware unit classification

The on-chip LIN hardware unit combines one or several LIN channels.

The following figure shows a classification of different LIN hardware types connected to multiple LIN physical channels:

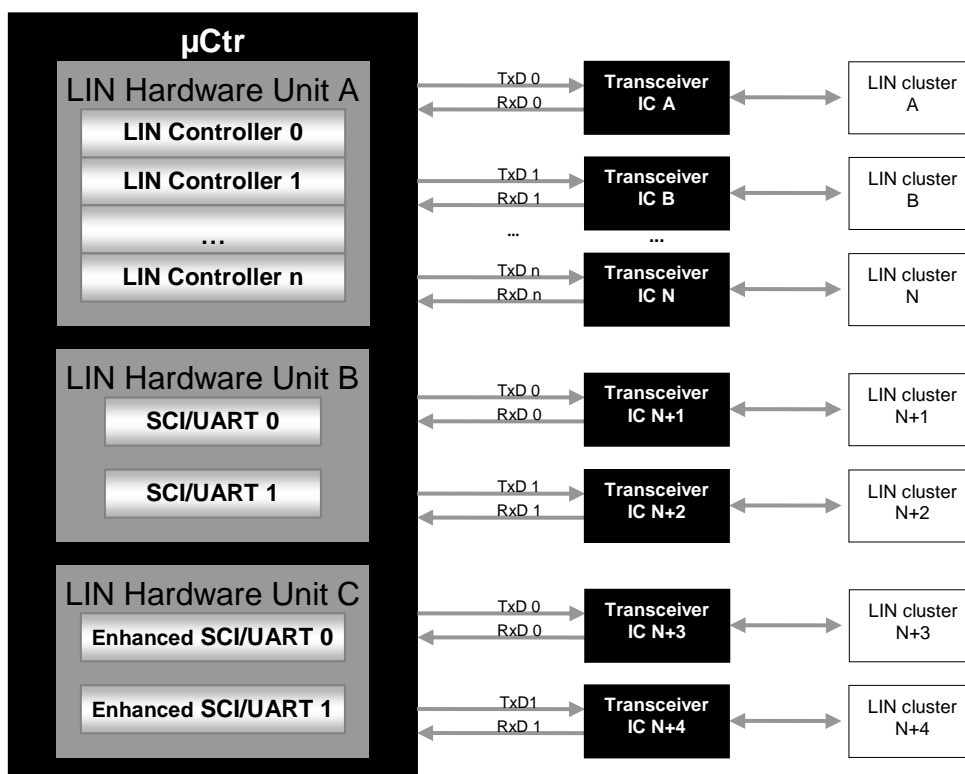


Figure 2-1: LIN hardware unit classification

### 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] General Requirements on SPAL  
AUTOSAR\_SRS\_SPALGeneral.pdf
- [7] Requirements on LIN  
AUTOSAR\_SRS\_LIN.pdf
- [8] Specification of LIN Interface  
AUTOSAR\_SWS\_LINInterface.pdf
- [9] Specification of ECU Configuration  
AUTOSAR\_TPS\_ECUConfiguration.pdf
- [10] Specification of MCU driver  
AUTOSAR\_SWS\_MCUDriver.pdf
- [11] Specification of Diagnostic Event Manager  
AUTOSAR\_SWS\_DiagnosticEventManager.pdf
- [12] Specification of ECU State Manager  
AUTOSAR\_SWS\_ECUCStateManager.pdf
- [13] Basic Software Module Description Template,  
AUTOSAR\_TPS\_BSWModuleDescriptionTemplate.pdf
- [14] Specification of LIN Transceiver Driver,  
AUTOSAR\_SWS\_LINTransceiverDriver.pdf
- [15] General Specification of Basic Software Modules  
AUTOSAR\_SWS\_BSWGeneral.pdf

#### 3.2 Related standards and norms

- [16] ISO 17987:2016 (all parts), Road vehicles – Local Interconnect Network (LIN)

### **3.3 Related specification**

AUTOSAR provides a General Specification on Basic Software modules [15] (SWS BSW General), which is also valid for LIN Driver.

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

## 4 Constraints and assumptions

### 4.1 Limitations

Only one LIN channel of an ECU is allowed to connect to a particular LIN cluster. Unless there are unused (not connected) channels in the ECU, the number of LIN channels is equal to the number of LIN clusters.

#### Driver scope

**[SWS\_Lin\_00045]** 「One LIN driver provides access to one LIN hardware unit type (simple UART or dedicated LIN hardware) that may consist of several LIN channels.」(SRS\_BSW\_00347)

**[SWS\_Lin\_00201]** 「For different LIN hardware units a separate LIN driver needs to be implemented. It is up to the implementer to adapt the driver to the different instances of similar LIN channels.」()

**[SWS\_Lin\_00177]** 「In case several LIN driver instances (of same or different vendor) are implemented in one ECU the file names, API names, and published parameters must be modified such that no two definitions with the same name are generated. The name shall be extended according to SRS\_BSW\_00347 with a Vendor Id (needed to distinguish LIN drivers from different vendors) and a Vendor specific name (needed to distinguish different hardware units implemented by one Vendor): <Module abbreviation>\_<Vendor Id>\_<Vendor specific name>.」()

The LIN Interface is responsible for calling the correct function. The necessary information shall be given in an XML file during configuration. See [8] for description how the LIN Interface handles several LIN drivers.

### 4.2 Applicability to car domains

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

## 5 Dependencies to other modules

### Module MCU [10]

The hardware of the internal LIN hardware unit depends on the system clock, prescaler(s) and PLL. Hence, the length of the LIN bit timing depends on the clock settings made in module [MCU](#).

The LIN driver module will not take care of setting the registers that configure the clock, prescaler(s) and PLL (e.g. switching on/off the PLL) in its init functions. The MCU module must do this.

### Module Port

The Port driver configures the port pins used for the LIN driver as input or output. Hence, the Port driver has to be initialized prior to the use of LIN functions. Otherwise, LIN driver functions will exhibit undefined behavior.

### Module DET (Default Error Tracer) [5]

In development mode, the Lin module reports development error through the Det\_ReportError function of module [DET](#). (see [SWS\\_Lin\\_00048](#))

### Module DEM (Diagnostic Event Manager) [11]

The Lin module reports production errors to the Diagnostic Event Manager. (see [SWS\\_Lin\\_00058](#))

### OS (Operating System)

The LIN driver uses interrupts and therefore there is a dependency on the OS, which configures the interrupt sources.

### LIN driver Users

The LIN Interface (specified by [8]) is the only user of the LIN driver services.

## 5.1 File structure

### 5.1.1 Code file structure

[SWS\_Lin\_00268]「The code file structure shall not be defined within this specification.」()

### 5.1.2 Header file structure

[SWS\_Lin\_00054]「The file `Lin.h` only contains external declarations of constants, global data, type definitions and services that are specified in the LIN driver SWS.」  
(SRS\_BSW\_00302)

**[SWS\_Lin\_00207]** 「Constants, global data types and functions that are only used by LIN driver internally, are declared in `Lin.c`.」()



## 6 Requirements traceability

Document: AUTOSAR requirements on Basic Software, general [3]

| Requirement   | Description   | Satisfied by  |
|---------------|---|---------------|
| SRS_BSW_00005 | Modules of the $\mu$ C Abstraction Layer (MCAL) may not have hard coded horizontal interfaces                                       | SWS_Lin_00999 |
| SRS_BSW_00007 | All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.  | SWS_Lin_00999 |
| SRS_BSW_00101 | The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function                  | SWS_Lin_00006 |
| SRS_BSW_00159 | All modules of the AUTOSAR Basic Software shall support a tool based configuration  | SWS_Lin_00029 |
| SRS_BSW_00162 | The AUTOSAR Basic Software shall provide a hardware abstraction layer   | SWS_Lin_00999 |
| SRS_BSW_00164 | The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules                  | SWS_Lin_00155 |
| SRS_BSW_00167 | All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks                  | SWS_Lin_00039 |
| SRS_BSW_00168 | SW components shall be tested by a function defined in a common API in the Basis-SW   | SWS_Lin_00999 |
| SRS_BSW_00302 | All AUTOSAR Basic Software Modules shall only export information needed by other modules  | SWS_Lin_00054 |
| SRS_BSW_00306 | AUTOSAR Basic Software Modules shall be compiler and platform independent   | SWS_Lin_00055 |
| SRS_BSW_00307 | Global variables naming convention  | SWS_Lin_00999 |
| SRS_BSW_00308 | AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file                                | SWS_Lin_00055 |
| SRS_BSW_00309 | All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword | SWS_Lin_00055 |
| SRS_BSW_00312 | Shared code shall be reentrant  | SWS_Lin_00999 |
| SRS_BSW_00323 | All AUTOSAR Basic Software Modules shall check passed API parameters for validity   | SWS_Lin_00048 |
| SRS_BSW_00325 | The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short                   | SWS_Lin_00999 |
| SRS_BSW_00327 | Error values naming convention  | SWS_Lin_00048 |
| SRS_BSW_00328 | All AUTOSAR Basic Software Modules shall avoid the duplication of code  | SWS_Lin_00999 |
| SRS_BSW_00330 | It shall be allowed to use macros instead of functions where source code is used and runtime is critical                            | SWS_Lin_00999 |

|               |   |               |
|---------------|---|---------------|
| SRS_BSW_00331 | All Basic Software Modules shall strictly separate error and status information   | SWS_Lin_00999 |
| SRS_BSW_00336 | Basic SW module shall be able to shutdown   | SWS_Lin_00999 |
| SRS_BSW_00337 | Classification of development errors  | SWS_Lin_00048 |
| SRS_BSW_00339 | Reporting of production relevant error status   | SWS_Lin_00999 |
| SRS_BSW_00342 | It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed              | SWS_Lin_00999 |
| SRS_BSW_00343 | The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit                                | SWS_Lin_00999 |
| SRS_BSW_00345 | BSW Modules shall support pre-compile configuration   | SWS_Lin_00013 |
| SRS_BSW_00347 | A Naming separation of different instances of BSW drivers shall be in place   | SWS_Lin_00045 |
| SRS_BSW_00353 | All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header                          | SWS_Lin_00999 |
| SRS_BSW_00357 | For success/failure of an API call a standard return type shall be defined  | SWS_Lin_00999 |
| SRS_BSW_00359 | All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible  | SWS_Lin_00999 |
| SRS_BSW_00360 | AUTOSAR Basic Software Modules callback functions are allowed to have parameters  | SWS_Lin_00999 |
| SRS_BSW_00361 | All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header | SWS_Lin_00999 |
| SRS_BSW_00373 | The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention                                | SWS_Lin_00999 |
| SRS_BSW_00375 | Basic Software Modules shall report wake-up reasons   | SWS_Lin_00098 |
| SRS_BSW_00378 | AUTOSAR shall provide a boolean type  | SWS_Lin_00999 |
| SRS_BSW_00383 | The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description    | SWS_Lin_00999 |
| SRS_BSW_00385 | List possible error notifications   | SWS_Lin_00048 |
| SRS_BSW_00395 | The Basic Software Module specifications shall list all configuration parameter dependencies  | SWS_Lin_00999 |
| SRS_BSW_00397 | The configuration parameters in pre-compile time are fixed before compilation starts  | SWS_Lin_00999 |
| SRS_BSW_00398 | The link-time configuration is achieved on object code basis in the stage after compiling and before linking                                      | SWS_Lin_00999 |
| SRS_BSW_00399 | Parameter-sets shall be located in a separate segment and shall be loaded after the code  | SWS_Lin_00999 |

|               |  |                                 |
|---------------|--|---------------------------------|
| SRS_BSW_00400 | Parameter shall be selected from multiple sets of parameters after code has been loaded and started  | SWS_Lin_00999                   |
| SRS_BSW_00404 | BSW Modules shall support post-build configuration   | SWS_Lin_00013                   |
| SRS_BSW_00405 | BSW Modules shall support multiple configuration sets  | SWS_Lin_00011,<br>SWS_Lin_00013 |
| 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_Lin_00006                   |
| SRS_BSW_00407 | Each BSW module shall provide a function to read out the version information of a dedicated module implementation                              | SWS_Lin_00001                   |
| SRS_BSW_00413 | An index-based accessing of the instances of BSW modules shall be done   | SWS_Lin_00999                   |
| SRS_BSW_00415 | Interfaces which are provided exclusively for one module shall be separated into a dedicated header file                                       | SWS_Lin_00999                   |
| SRS_BSW_00416 | The sequence of modules to be initialized shall be configurable  | SWS_Lin_00999                   |
| SRS_BSW_00417 | Software which is not part of the SW-C shall report error events only after the DEM is fully operational.                                      | SWS_Lin_00999                   |
| SRS_BSW_00422 | Pre-de-bouncing of error status information is done within the DEM   | SWS_Lin_00999                   |
| SRS_BSW_00423 | BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template   | SWS_Lin_00999                   |
| SRS_BSW_00424 | BSW module main processing functions shall not be allowed to enter a wait state  | SWS_Lin_00999                   |
| SRS_BSW_00425 | The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects                         | SWS_Lin_00999                   |
| SRS_BSW_00426 | BSW Modules shall ensure data consistency of data which is shared between BSW modules  | SWS_Lin_00999                   |
| SRS_BSW_00427 | ISR functions shall be defined and documented in the BSW module description template   | SWS_Lin_00999                   |
| SRS_BSW_00428 | A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence                                 | SWS_Lin_00999                   |
| SRS_BSW_00429 | Access to OS is restricted   | SWS_Lin_00999                   |
| SRS_BSW_00432 | Modules should have separate main processing functions for read/receive and write/transmit data path   | SWS_Lin_00999                   |
| SRS_BSW_00433 | Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler   | SWS_Lin_00999                   |
| SRS_Lin_01502 | The LIN Interface shall support an API for RX/TX notifications.  | SWS_Lin_00999                   |

|               |  |   |
|---------------|--|---|
| SRS_Lin_01503 | An API shall exist that enables the LIN driver to directly copy up to 8 byte directly from/to the frame buffers. | SWS_Lin_00024,<br>SWS_Lin_00025,<br>SWS_Lin_00274,<br>SWS_Lin_00283 |
| SRS_Lin_01504 | The usage of AUTOSAR architecture shall be applicable for LIN master nodes                                       | SWS_Lin_00005   |
| SRS_Lin_01514 | The LIN Interface shall inform an upper layer about wake-up events   | SWS_Lin_00999   |
| SRS_Lin_01515 | The LIN Interface shall provide an API to wake-up a LIN channel cluster  | SWS_Lin_00999   |
| SRS_Lin_01522 | LIN-SDU shall be copied consistently for transfer  | SWS_Lin_00025,<br>SWS_Lin_00053,<br>SWS_Lin_00060,<br>SWS_Lin_00283 |
| SRS_Lin_01523 | There shall be an API call to set the LIN bus to sleep-mode.   | SWS_Lin_00999   |
| SRS_Lin_01524 | The LIN Driver shall be able to put the LIN hardware to a reduced power operation mode if needed                 | SWS_Lin_00032   |
| SRS_Lin_01526 | The LIN Driver shall provide a status for error events on the bus.   | SWS_Lin_00053   |
| SRS_Lin_01534 | The AUTOSAR LIN Transport Layer shall support half-duplex physical connections.                                  | SWS_Lin_00999   |
| SRS_Lin_01539 | The Transport connection properties shall be statically configured.  | SWS_Lin_00999   |
| SRS_Lin_01540 | The LIN Transport Layer shall provide an API for initialization.   | SWS_Lin_00999   |
| SRS_Lin_01544 | Errors shall be handled  | SWS_Lin_00999   |
| SRS_Lin_01545 | The LIN Transport Layer services shall not be operational before initializing the module.                        | SWS_Lin_00999   |
| SRS_Lin_01546 | The LIN Interface shall contain a Schedule Table Handler for LIN master nodes.                                   | SWS_Lin_00999   |
| SRS_Lin_01547 | The LIN Driver shall support standard UART and LIN optimized HW  | SWS_Lin_00063   |
| SRS_Lin_01549 | The LIN Interface needs to use a timer service for scheduling for LIN master nodes                               | SWS_Lin_00999   |
| SRS_Lin_01551 | One LIN Interface shall support one or more LIN Drivers.   | SWS_Lin_00999   |
| SRS_Lin_01555 | The LIN driver shall have an interface to retrieve transmit / receive notifications.                             | SWS_Lin_00024,<br>SWS_Lin_00274,<br>SWS_Lin_00275                   |
| SRS_Lin_01556 | One LIN driver shall be able to handle more than one LIN channel   | SWS_Lin_00008,<br>SWS_Lin_00190                                     |
| SRS_Lin_01558 | The LIN Interface shall check for successful data transfer for LIN master nodes                                  | SWS_Lin_00999   |
| SRS_Lin_01560 | If a wakeup occurs during transition to sleep-mode, this channel shall go back to the running mode               | SWS_Lin_00033   |
| SRS_Lin_01561 | The LIN Interface shall define a main function per   | SWS_Lin_00999   |

|               |  |   |
|---------------|--|---|
|               | channel  |   |
| SRS_Lin_01563 | The LIN Driver shall provide a notification for wake-up events   | SWS_Lin_00098                                     |
| SRS_Lin_01564 | A Schedule Table Manager shall be available for LIN master nodes.  | SWS_Lin_00999                                     |
| SRS_Lin_01566 | Transition to sleep-mode shall be handled  | SWS_Lin_00033,<br>SWS_Lin_00266                   |
| SRS_Lin_01568 | The LIN Interface implementation and interface shall be independent from underlying LIN hardware.          | SWS_Lin_00999                                     |
| SRS_Lin_01569 | The LIN Interface shall support initialization of each LIN channel separately                              | SWS_Lin_00999                                     |
| SRS_Lin_01570 | The LIN Interface shall support dynamic selection of configuration sets.                                   | SWS_Lin_00999                                     |
| SRS_Lin_01571 | Transmission request service shall be provided   | SWS_Lin_00999                                     |
| SRS_Lin_01572 | The LIN Driver shall support the initialization of each LIN channel separately                             | SWS_Lin_00011                                     |
| SRS_Lin_01573 | The LIN Driver shall support dynamic selection of configuration sets.                                      | SWS_Lin_00011                                     |
| SRS_Lin_01574 | It shall be possible to have one instance of the TP for each channel                                       | SWS_Lin_00999                                     |
| SRS_Lin_01576 | The ISO 17987 specifications shall be reused as far as possible  | SWS_Lin_00005                                     |
| SRS_Lin_01577 | It shall be compatible to LIN protocol specification   | SWS_Lin_00005                                     |
| SRS_Lin_01578 | It shall be compatible to LIN Datalinklayer  | SWS_Lin_00017,<br>SWS_Lin_00272,<br>SWS_Lin_00273 |
| SRS_Lin_01590 | The node configuration of LIN slaves shall only be done via defined schedule table(s) in master nodes.     | SWS_Lin_00999                                     |
| SRS_Lin_01594 | LIN slave shall support the node configuration and identification services for slave nodes.                | SWS_Lin_00999                                     |
| SRS_Lin_01595 | The LIN Interface shall support the setting and clearing of the response error signal for LIN slave nodes. | SWS_Lin_00999                                     |
| SRS_Lin_01596 | The LIN Interface shall provide bus idle condition observation for slave nodes.                            | SWS_Lin_00999                                     |
| SRS_Lin_01597 | The Lin Interface shall take care of the behavior of the slavenode transmission handler for slave nodes.   | SWS_Lin_00999                                     |

## 7 Functional specification

The LIN driver module is required to manage the hardware dependent aspects of communication via any LIN cluster attached to the node the driver resides in.

This includes accepting header data for transmission onto the bus, response frame data to transmit, the retrieval of header information and of response frame data intended for the node.

The need for sleep mode management of both the node and of the cluster exists. This implies the ability to detect and generate a 'wake-up' pulse as defined in the ISO 17987 specifications. If the underlying hardware supports a low-power mode then entering and exiting from that state is included.

### 7.1 General Requirements

The Lin module is a Basic Software Module that has direct access to hardware resources.

**[SWS\_Lin\_00005]** 「The Lin module shall conform to the ISO 17987 specifications [16]. This applies to ISO 17987 LIN Master and Slave nodes. 」(SRS\_Lin\_01576, SRS\_Lin\_01504, SRS\_Lin\_01577)

**[SWS\_Lin\_00055]** 「The Lin module shall fulfill all design and implementation guidelines as described in [3]. 」(SRS\_BSW\_00306, SRS\_BSW\_00308, SRS\_BSW\_00309)

**[SWS\_Lin\_00155]** 「The Lin module shall implement the ISRs for all LIN hardware unit interrupts that are needed. 」(SRS\_BSW\_00164)

**[SWS\_Lin\_00156]** 「The Lin module shall ensure that all unused interrupts are disabled. 」()

**[SWS\_Lin\_00157]** 「The Lin module shall reset the interrupt flag at the end of the ISR (if not done automatically by hardware). 」()

The Lin module shall not configure the interrupt (i.e. priority) nor set the vector table entry.

## 7.2 Version Check

### 7.2.1 Requirements

For details refer to the chapter 5.1.8 “Version Check” in *SWS\_BSWGeneral*.

## 7.3 LIN driver and Channel Initialization

### 7.3.1 Background & Rationale

Before communication can be started on a LIN bus, both the LIN driver and the relevant LIN channel must be initialized.

The driver initialization (see `Lin_Init`) handles all aspects of initialization that are of relevance to all channels present in the LIN hardware unit. This may include any static variables or hardware register settings common to all LIN channels that are available. Additionally each channel must also be initialized according to the configuration supplied. This will for example include (but is not limited to) the baud rate over the bus.

**[SWS\_Lin\_00225]** 「There must be at least one statically defined configuration set available for the LIN driver. When the EcuM invokes the initialization function, it has to provide a specific pointer to the configuration that it wishes to use.」()

### 7.3.2 Requirements

The Lin module shall not initialize or configure LIN channels, which are not used. The Lin module shall allow the environment to select between different static configuration data at runtime.

**[SWS\_Lin\_00011]** 「The Lin module’s configuration shall include a data communication rate set as defined by static configuration data.」(SRS\_BSW\_00405, SRS\_Lin\_01572, SRS\_Lin\_01573)

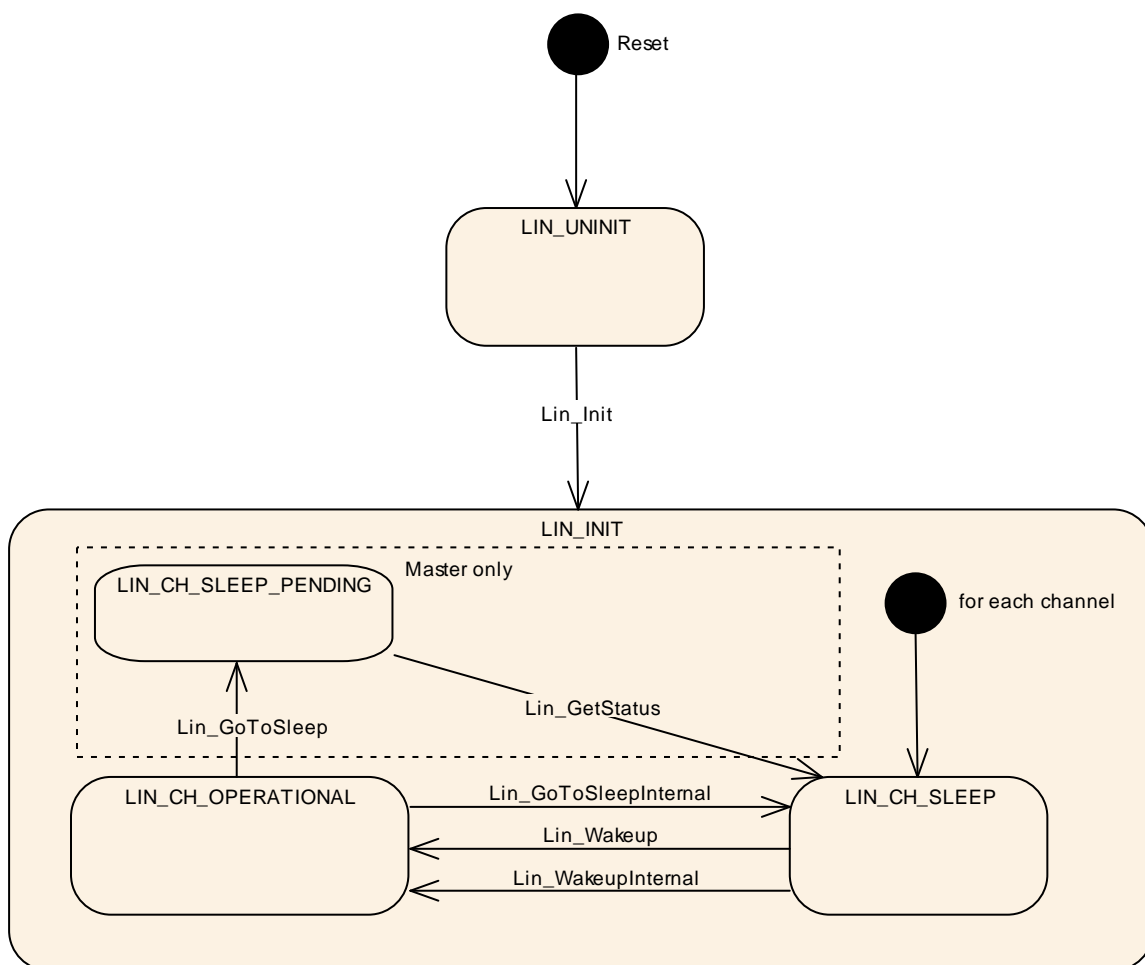
**[SWS\_Lin\_00013]** 「The Lin module’s configuration data, intended for hardware registers, shall be stored as hardware specific data structures in ROM (see [Lin\\_ConfigType](#)).」(SRS\_BSW\_00345, SRS\_BSW\_00404, SRS\_BSW\_00405)

**[SWS\_Lin\_00014]** 「Each LIN PID shall be associated with a checksum model (either ‘enhanced’ where the PID is included in the checksum, or ‘classic’ where only the response data is check-summed) (see [Lin\\_PduType](#)).」()

**[SWS\_Lin\_00015]** 「Each LIN PID shall be associated with a response data length in bytes (see [Lin\\_PduType](#)).」()

### 7.3.3 State diagrams

The LIN driver has a state machine that is shown in Figure 7-1.



**Figure 7-1: LIN driver states**

| Module State      | Meaning / Activities in the state   |
|-------------------|---|
| <b>LIN_UNINIT</b> | The state LIN_UNINIT means that the Lin module has not been initialized yet and cannot be used.                         |
| <b>LIN_INIT</b>   | The LIN_INIT state indicates that the LIN driver has been initialized, making each available channel ready for service. |



| Channel State             | Meaning / Activities in the state  |
|---------------------------|--|
| <b>LIN_CH_OPERATIONAL</b> | The individual channel has been initialized (using at least one statically configured data set) and is able to participate in the LIN cluster. |
| <b>LIN_CH_SLEEP</b>       | The detection of a 'wake-up' pulse is enabled. The LIN hardware is into a low power mode if such a mode is provided by the hardware.           |

**[SWS\_Lin\_00145]** **Reset -> LIN\_UNINIT:** After reset, the Lin module shall set its state to LIN\_UNINIT. `⌈()`

**[SWS\_Lin\_00146]** **LIN\_UNINIT -> LIN\_INIT:** The Lin module shall transition from LIN\_UNINIT to LIN\_INIT when the function `Lin_Init` is called. `⌈()`

The LIN module's environment shall call the function `Lin_Init` only once during runtime.

**[SWS\_Lin\_00171]** **On entering the state LIN\_INIT,** the Lin module shall set each channel into state LIN\_CH\_SLEEP, enable bus monitoring for a wake-up request on that channel if external wake-up detection is supported by configuration parameter `LinChannelWakeupSupport`, and optionally set the LIN hardware unit to reduced power operation mode (if supported by HW). `⌈()`

**[SWS\_Lin\_00263]** **LIN\_CH\_OPERATIONAL -> LIN\_CH\_SLEEP\_PENDING through Lin\_GoToSleep:** If a go to sleep is requested by the LIN interface, the Lin module shall ensure that the rest of the LIN cluster goes to sleep also. This is achieved by issuing a go-to-sleep-command on the bus before entering the LIN\_CH\_SLEEP\_PENDING state. This requirement is only applicable for LIN master nodes. `⌈()`

**[SWS\_Lin\_00264]** **LIN\_CH\_SLEEP\_PENDING -> LIN\_CH\_SLEEP:** When `Lin_GetStatus` is called, the LIN driver shall directly enter the LIN\_CH\_SLEEP state, even if the go-to-sleep-command has not yet been sent. This requirement is only applicable for LIN master nodes. `⌈()`

**[SWS\_Lin\_00265]** **LIN\_CH\_OPERATIONAL -> LIN\_CH\_SLEEP through Lin\_GoToSleepInternal:** If an internal go to sleep is requested by the LIN interface, the LIN driver shall directly enter the LIN\_CH\_SLEEP state. `⌈()`

**[SWS\_Lin\_00174]** **LIN\_CH\_SLEEP -> LIN\_CH\_OPERATIONAL through Lin\_Wakeup:** If a LIN channel is in the state LIN\_CH\_SLEEP, the function `Lin_Wakeup` shall put the LIN channel into the state LIN\_CH\_OPERATIONAL. `⌈()`

**[SWS\_Lin\_00261]** 「 LIN\_CH\_SLEEP -> LIN\_CH\_OPERATIONAL through Lin\_WakeupInternal: If a LIN channel is in the state LIN\_CH\_SLEEP, the function Lin\_WakeupInternal shall put the LIN channel into the state LIN\_CH\_OPERATIONAL. 」()

**[SWS\_Lin\_00209]** 「Lin\_Wakeup: During the state transition from LIN\_CH\_SLEEP to LIN\_CH\_OPERATIONAL the LIN Driver shall ensure that the rest of the cluster is awake. This is achieved by issuing a wake-up request, forcing the bus to the dominant state for 250  $\mu$ s to 5 ms.」()

**[SWS\_Lin\_00184]** 「A mode switch request to the current mode is allowed and shall not lead to an error, even if DET is enabled.」()

## 7.4 Frame processing

### 7.4.1 Background & Rationale

A LIN frame is composed of two parts, the LIN header and the LIN response. The LIN header is always transmitted by the LIN master node and indicates the start of frame, including the LIN PID. The LIN response is transmitted on the bus after the LIN header and can be transmitted by either the master or one of the slaves [16].

The driver must also be able to access data concerning the checksum model and data length for each LIN PID. LIN 2.0 and newer versions have a different checksum model compared to LIN 1.3, but the LIN master must be able to communicate with all slave node types (LIN 1.3, LIN 2.0, LIN 2.1, LIN 2.2 and ISO 17987). However, a LIN 1.3 master is not able to communicate with LIN 2.0 or newer slaves.

The checksum is a part of the response, and may or may not include the PID depending upon the checksum model for the PID in question. The LIN ID's 60 (0x3c) to 63 (0x3f) must always use the classic (response data only) checksum model [16].

The LIN driver module works with LIN frames as its basic building block. From the LIN driver module point of view, the frame handling significantly differs depending on the implemented node type:

- For a LIN master node, the LIN interface layer requests a particular frame to be sent during one of its scheduler time-slots. Any response from the frame should be available latest before the next frame will be sent.  
In the case that the master is also responsible for sending the frame response, an indication (`PduInfoPtr->Drc= LIN_FRAMERESPONSE_TX`) will be given at the same time as the request to send the frame.
- A LIN slave node waits for the reception of a LIN header. A received LIN header is indicated to the LIN interface that evaluates the PID and returns information about the response. In the case that the slave is responsible for sending the frame response, the transmission data is also directly provided.

The LIN driver module must be able to retrieve data from the response and make it available to the LIN interface module. It must retrieve all data from the response without blocking.

### 7.4.2 Requirements

#### 7.4.2.1 LIN Master specific

Following requirements are only applicable for LIN master nodes.

**[SWS\_Lin\_00016]** 「The LIN driver shall interpret the supplied identifier as PID. The identifier is then transmitted *as-supplied* within the LIN header (see [Lin\\_SendFrame](#)).」()

**[SWS\_Lin\_00017]** 「The LIN driver shall be able to send a LIN header. This is composed of the break field, synch byte field, and protected identifier byte field as detailed in [16] (see [Lin\\_SendFrame](#)).」(SRS\_Lin\_01578)

**[SWS\_Lin\_00018]** 「The LIN driver shall be able to send a LIN header and response.」()

**[SWS\_Lin\_00021]** 「The LIN driver shall abort the current frame transmission if a new frame transmission is requested by the LIN interface (see [Lin\\_SendFrame](#)), also if an ongoing transmission may be still in progress or unsuccessfully completed.」()

**[SWS\_Lin\_00022]** 「The function `Lin_GetStatus` shall return the status of the current frame transmission request for the channel.」()

**[SWS\_Lin\_00024]** 「The LIN driver shall make received data available to the LIN interface module. After successful reception of a whole LIN frame, the received data shall be prepared for function call of the LIN interface (see [Lin\\_GetStatus](#)).」  
(SRS\_Lin\_01555, SRS\_Lin\_01503)

**[SWS\_Lin\_00025]** 「The LIN driver shall send response data as provided by the LIN interface module (see [Lin\\_SendFrame](#)).」( SRS\_Lin\_01522, SRS\_Lin\_01503)

#### 7.4.2.2 LIN Slave specific

Following requirements are only applicable for LIN Slave nodes.

**[SWS\_Lin\_00272]** 「The LIN driver shall be able to receive a LIN header at any time in `LIN_CH_OPERATIONAL` state. The header is composed of the break field, synch byte field, and protected identifier byte field as detailed in [16].」(SRS\_Lin\_01578)

**[SWS\_Lin\_00280]** 「 On LIN header reception, the LIN driver shall call the header indication callback function `LinIf_HeaderIndication` with `PduPtr->Pid` set to the received PID value and `PduPtr->SduPtr` set to the (hardware or shadow) buffer of the LIN driver, to which the slave response shall be written by the upper layer. 」()

Note: If the LIN hardware unit provides the ID (frame identifier without parity bits) instead of the PID, the LIN driver is responsible to calculate the PID from the ID to comply with the callback interface.

**[SWS\_Lin\_00271]** ⌈ If the LIN hardware unit cannot detect invalid PIDs, the LIN driver shall not evaluate the PID value (i.e. it shall not verify parity bits in software). The LIN driver shall provide the PID *as-received* to the LIN interface module. ⌋()

**[SWS\_Lin\_00281]** ⌈ While waiting for a new LIN header, the LIN driver shall call the error indication callback function `LinIf_LinErrorIndication` with error parameter `LIN_ERR_HEADER` when it detects bus events that do not comply to a valid LIN header (e.g incomplete LIN header). ⌋()

**[SWS\_Lin\_00273]** ⌈ The LIN driver shall be able to send, receive or ignore a LIN response. ⌋(SRS\_Lin\_01578)

**[SWS\_Lin\_00282]** ⌈ After the call of `LinIf_HeaderIndication` when the return value is `E_OK`, the LIN driver shall evaluate the `PduPtr->Drc` to determine the type of LIN response. ⌋()

**[SWS\_Lin\_00284]** ⌈ If the LIN response is going to be received (`LIN_FRAMERESPONSE_RX`), the LIN driver shall evaluate the `Cs` and `DI` members in parameter `PduPtr` (after the call of `LinIf_HeaderIndication` with return value `E_OK`) to configure the LIN response reception. ⌋()

**[SWS\_Lin\_00274]** ⌈ After successful reception of a LIN response, the LIN driver shall directly make the received data available to the LIN interface module by calling the Rx indication callback function `LinIf_RxIndication` with the `Lin_SduPtr` parameter set to the reception data. ⌋(SRS\_Lin\_01555, SRS\_Lin\_01503)

**[SWS\_Lin\_00283]** ⌈ If the LIN response is going to be transmitted (`LIN_FRAMERESPONSE_TX`), the LIN driver shall evaluate the `Cs`, `DI` and `SduPtr` members in parameter `PduPtr` (after the call of `LinIf_HeaderIndication` with return value `E_OK`) to setup and transmit the LIN response. ⌋(SRS\_Lin\_01503, SRS\_Lin\_01522)

**[SWS\_Lin\_00286]** ⌈ If the return value of `LinIf_HeaderIndication` is `E_NOT_OK` or the returned `PduPtr->Drc` is `LIN_FRAMERESPONSE_IGNORE`, the LIN driver shall ignore the response. ⌋()

**[SWS\_Lin\_00275]** ⌈ After successful transmission of a LIN response, the transmission shall be directly confirmed to the LIN Interface module by calling the Tx confirmation callback function `LinIf_TxConfirmation`. ⌋( `SRS_Lin_01555`)

**[SWS\_Lin\_00276]** ⌈The LIN driver shall not report any events to the LIN interface module if the LIN response is ignored until the reception of a new LIN header. ⌋()

**[SWS\_Lin\_00277]** ⌈The LIN driver shall detect communication errors during response transmission and response reception. Once an error is detected, the current frame handling shall be aborted and the error indication callback function `LinIf_LinErrorIndication` shall be called. ⌋()

**[SWS\_Lin\_00285]** ⌈ The handling of each relevant (i.e. not ignored) LIN response must be completed by a call to either `LinIf_RxIndication`, `LinIf_TxConfirmation` or `LinIf_LinErrorIndication`, latest before a new LIN header reception is indicated by a call of `Lin_HeaderIndication`. ⌋()

### 7.4.2.3 Common

**[SWS\_Lin\_00019]** ⌈The LIN driver shall be able to calculate either a ‘classic’ or an ‘enhanced’ checksum depending upon the checksum model for the current LIN PDU. ⌋()

**[SWS\_Lin\_00026]** ⌈If the LIN hardware unit cannot queue the bytes for transmission or reception (e.g. simple UART implementation), the LIN driver shall provide a temporary communication buffer. ⌋()

**[SWS\_Lin\_00027]** ⌈The LIN driver shall initiate transmission without blocking, including the check of the next byte transmission only upon successful reception of the previous one (receive-back). ⌋()

**[SWS\_Lin\_00028]** ⌈The LIN driver shall receive data without blocking. ⌋()

## 7.4.3 Data Consistency

### 7.4.3.1 Transmit Data Consistency:

**[SWS\_Lin\_00053]** 「The LIN driver shall directly copy the data from the upper layer buffers. 」( SRS\_Lin\_01522, SRS\_Lin\_01526)

**[SWS\_Lin\_00210]** 「For LIN Master nodes, the upper layer of the LIN Driver has to keep the buffer data consistent until return of function call.  
For LIN Slave nodes, the upper layer of the LIN Driver has to keep the buffer data consistent until end of response transmission. 」()

#### **7.4.3.2 Receive Data Consistency:**

**Following is applicable for LIN master and LIN slave nodes:**

For the LIN response reception the bytes of the SDU buffer shall be allocated in increasingly consecutive address order. The LIN frame data length information defines the minimum SDU buffer length.

**[SWS\_Lin\_00060]** 「The complete LIN frame receive processing (including copying to destination layer) can be implemented in an ISR. The received data shall be consistent until either next LIN frame has been received successfully or LIN channel state has changed. 」(SRS\_Lin\_01522)

As long as it is guaranteed that neither the ISRs nor Lin\_GetStatus (Master only) can be interrupted by itself, the LIN hardware (or shadow) buffer is always consistent, because it is written and read in sequence in exactly one function that is never interrupted by itself.

##### **7.4.3.2.1 Receive data consistency (Master only)**

**[SWS\_Lin\_00211]** 「The complete LIN frame receive processing (including copying to destination layer) can be implemented in the Lin\_GetStatus function. The received data shall be consistent until either next LIN frame has been received successfully or LIN channel state has changed. 」()

#### **7.4.4 Data byte mapping**

**[SWS\_Lin\_00096]** 「Data mapping between memory and the LIN frame is defined in a way that the array element 0 is containing the LSB (the data byte to send/receive first) and the array element (n-1) is containing the MSB (the data byte to send/receive last). 」()

## 7.5 Sleep and wake-up functionality

There are two different possibilities to wake-up a LIN hardware channel:

1. Internal (Top-Down) wake-up  
The upper layer requests a wake-up through a call to `Lin_Wakeup`.
2. External (Bottom-Up) wake-up  
A bus wake-up event is detected and forwarded to the upper layer through the `Lin_CheckWakeup` API which has to be called by module `LinIf`. After a successful validation of the wake-up source, `Lin_WakeupInternal` is also called.

The selection of the wake-up modes is controlled by configuration parameter `LinChannelWakeupSupport`:

- Both internal and external wake-up modes are supported by a LIN hardware channel having `LinChannelWakeupSupport = TRUE`
- Only internal wake-up mode is supported by a LIN hardware channel having `LinChannelWakeupSupport = FALSE`.

### 7.5.1 Background & Rationale

**Following is applicable for LIN master nodes only:**

The LIN Master node can be awakened either by a wake-up signal generated by one of the slaves, or by a request from the higher layer (LIN interface) (see [SWS\\_Lin\\_00209](#)). The LIN interface controls the message schedule table and so must be able to instruct the LIN driver to put the hardware unit to sleep, or to wake it up (see [SWS\\_LinIf\\_00296](#), [SWS\\_LinIf\\_00488](#)).

For this purpose, the LIN driver provides functions to put the LIN channel into its `LIN_CH_SLEEP` state (see [Lin\\_GoToSleep/Lin\\_GoToSleepInternal](#)).

**Following is applicable for LIN slave nodes only:**

The LIN slave can be awakened either by a wake-up signal generated by the master node or any other slave node on the same channel, or by a request from the higher layer (LIN interface) (see [SWS\\_Lin\\_00209](#)).

The LIN Interface detects the conditions to enter sleep mode and instructs the LIN driver to put the hardware unit to sleep, or to wake it up (see [SWS\\_LinIf\\_00296](#), [SWS\\_LinIf\\_00488](#)).

For this purpose, the LIN driver provides a function to put the LIN channel into its `LIN_CH_SLEEP` state (see [Lin\\_GoToSleepInternal](#)). Because the slave node cannot transmit a go-to-sleep-command, the function [Lin\\_GoToSleep](#) is not applicable for slave nodes.



**Following is applicable for LIN master and LIN slave nodes:**

Upon sleep or wake-up the master must communicate the status change with the rest of the network.

**7.5.2 Requirements**

**[SWS\_Lin\_00032]** 「When the LIN channel enters sleep mode, it shall perform the transition to low-power mode of the LIN hardware unit (if available) (see [Lin\\_GoToSleep/Lin\\_GoToSleepInternal](#)).」( SRS\_Lin\_01524)

**[SWS\_Lin\_00033]** 「Each LIN channel shall be able to accept a sleep request independently of the other channel states (see [Lin\\_GoToSleep/Lin\\_GoToSleepInternal](#)).」(SRS\_Lin\_01560, SRS\_Lin\_01566)

**[SWS\_Lin\_00037]** 「When a LIN channel is in LIN\_CH\_SLEEP state and wake-up detection is supported by configuration parameter LinChannelWakeupSupport, the LIN hardware unit shall monitor the bus for a wake-up request on that channel.」()

**[SWS\_Lin\_00043]** 「Lin\_Wakeup: If the LIN driver receives a wake-up request from the LIN interface, the requested channel shall send a wake-up pulse to the LIN bus. (see [Lin\\_Wakeup](#))」()

**[SWS\_Lin\_00262]** 「Lin\_WakeupInternal: If the LIN driver receives an internal wake-up request from the LIN interface, the requested channel shall send no wake-up pulse to the LIN bus. (see [Lin\\_WakeupInternal](#))」()

For LIN Master nodes, the function [Lin\\_GetStatus](#) returns the current state of a given LIN channel.

**7.6 Error classification**

Section 7.x "Error Handling" of the document "General Specification of Basic Software Modules" describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types, which may occur in BSW modules.

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

## 7.6.1 Development Errors

### [SWS\_Lin\_00048]

| <i>Type of error</i>   | <i>Related error code</i> | <i>Error value</i> |
|--|---------------------------|--------------------|
| API service used without module initialization                 | LIN_E_UNINIT              | 0x00               |
| API service used with an invalid or inactive channel parameter | LIN_E_INVALID_CHANNEL     | 0x02               |
| API service called with invalid configuration pointer          | LIN_E_INVALID_POINTER     | 0x03               |
| Invalid state transition for the current state                 | LIN_E_STATE_TRANSITION    | 0x04               |
| API service called with a NULL pointer                         | LIN_E_PARAM_POINTER       | 0x05               |
| --   |                           | --                 |
| Timeout caused by hardware error                               | LIN_E_TIMEOUT             | Assigned by DEM    |

](SRS\_BSW\_00323, SRS\_BSW\_00327, SRS\_BSW\_00337, SRS\_BSW\_00385)

## 7.6.2 Runtime Errors

There are no runtime errors.

## 7.6.3 Transient Faults

There are no transient faults.

## 7.6.4 Production Errors

### 7.6.4.1 LIN\_E\_TIMEOUT

#### [SWS\_Lin\_00290]

|                            |   |   |
|----------------------------|---|---|
| <b>Error Name:</b>         | LIN_E_TIMEOUT   |   |
| <b>Short Description:</b>  | This error is reported when time out caused by hardware error occurs.   |   |
| <b>Long Description:</b>   | If a change to the LIN hardware control registers results in the need to wait for a status change, this shall be protected by a configurable time out mechanism. If such a time out is detected the LIN_E_TIMEOUT error shall be raised. This situation should only arise in the event of a LIN hardware unit fault and should be communicated to the rest of the system. |   |
| <b>Recommended DTC:</b>    | -   |   |
| <b>Detection Criteria:</b> | Fail  | A LIN hardware control register has changed and the configured time (see <a href="#">LinTimeoutDuration</a> ) has elapsed |

|                              |   |   |
|------------------------------|---|---|
|                              |   | without a status change of the LIN Hardware.  |
|                              | Pass  | A LIN hardware control register has changed and the status change is done within the configured time (see <a href="#">LinTimeoutDuration</a> ). |
| <b>Secondary Parameters:</b> | The LIN_E_TIMEOUT is only used (Fail/Pass detection is active) if a change in the LIN hardware control registers does not immediately result in a status change, but it needs some time and time is measureable. For such hardware, it means, the timeout mechanism is started whenever the LIN hardware register is changed. The timeout mechanism is stopped and reset, when the status change is successfully done (Pass detection) or the configured time (see <a href="#">LinTimeoutDuration</a> ) has elapsed (Fail detection). |   |
| <b>Time Required:</b>        | 1s  |   |
| <b>Monitor Frequency:</b>    | once-per-trip   |   |
| <b>MIL illumination:</b>     | -   |   |

⌋()

**[SWS\_Lin\_00058]** ⌈ The only production error that can be reported by the LIN driver is the LIN\_E\_TIMEOUT error. ⌋()

### 7.6.5 Extended Production Errors

There are no extended production errors.

## 8 API specification

### 8.1 Imported types

In this chapter all types included from other modules are listed:

[SWS\_Lin\_00226]

| <i>Module</i>  | <i>Header File</i> | <i>Imported Type</i>  |
|----------------|--------------------|-----------------------|
| ComStack_Types | ComStack_Types.h   | NetworkHandleType     |
| Dem            | Rte_Dem_Type.h     | Dem_EventIdType       |
|                | Rte_Dem_Type.h     | Dem_EventStatusType   |
| EcuM           | EcuM.h             | EcuM_WakeupSourceType |
| Icu            | Icu.h              | Icu_ChannelType       |
| Std            | Std_Types.h        | Std_ReturnType        |
|                | Std_Types.h        | Std_VersionInfoType   |

()

### 8.2 Type definitions

[SWS\_Lin\_00245] [The content of `Lin_GeneralTypes.h` shall be protected by a `LIN_GENERAL_TYPES` define.]()

[SWS\_Lin\_00246] [If different LIN drivers are used, only one instance of this file has to be included in the source tree. For implementation all `Lin_GeneralTypes.h` related types in the documents mentioned before shall be considered.]()

#### 8.2.1 Lin\_ConfigType

[SWS\_Lin\_00227]

|                 |   |  |
|-----------------|---|--|
| <b>Name</b>     | Lin_ConfigType                                  |  |
| <b>Kind</b>     | Structure                                       |  |
| <b>Elements</b> | Hardware and Implementation dependent structure |  |
|                 | <b>Type</b>                                     | --   |
|                 | <b>Comment</b>                                  | The contents of the initialization data structure are LIN hardware |

|                      |   |          |
|----------------------|---|----------|
|                      |   | specific |
| <b>Description</b>   | This is the type of the external data structure containing the overall initialization data for the LIN driver and the SFR settings affecting the LIN channels. A pointer to such a structure is provided to the LIN driver initialization routine for configuration of the driver, LIN hardware unit and LIN hardware channels. |          |
| <b>Available via</b> | Lin.h   |          |

]()

### 8.2.2 Lin\_FramePidType

[SWS\_Lin\_00228]

|                      |  |    |  |
|----------------------|--|----|--|
| <b>Name</b>          | Lin_FramePidType   |    |  |
| <b>Kind</b>          | Type   |    |  |
| <b>Derived from</b>  | uint8  |    |  |
| <b>Range</b>         | 0...0xFE   | -- | The LIN identifier (0...0x3F) together with its two parity bits. |
| <b>Description</b>   | Represents all valid protected identifier used by Lin_SendFrame(). |    |  |
| <b>Available via</b> | Lin_GeneralTypes.h   |    |  |

]()

### 8.2.3 Lin\_FrameCsModelType

[SWS\_Lin\_00229]

|                      |   |    |                         |
|----------------------|---|----|-------------------------|
| <b>Name</b>          | Lin_FrameCsModelType  |    |                         |
| <b>Kind</b>          | Enumeration   |    |                         |
| <b>Range</b>         | LIN_ENHANCED_CS   | -- | Enhanced checksum model |
|                      | LIN_CLASSIC_CS  | -- | Classic checksum model  |
| <b>Description</b>   | This type is used to specify the Checksum model to be used for the LIN Frame. |    |                         |
| <b>Available via</b> | Lin_GeneralTypes.h  |    |                         |

]()

### 8.2.4 Lin\_FrameResponseType

[SWS\_Lin\_00230]

|                      |  |    |   |
|----------------------|--|----|---|
| <b>Name</b>          | Lin_FrameResponseType  |    |   |
| <b>Kind</b>          | Enumeration  |    |   |
| <b>Range</b>         | LIN_FRAMERESPONSE_TX   | -- | Response is generated from this node                                    |
|                      | LIN_FRAMERESPONSE_RX   | -- | Response is generated from another node and is relevant for this node.  |
|                      | LIN_FRAMERESPONSE_IGNORE   | -- | Response is generated from another node and is irrelevant for this node |
| <b>Description</b>   | This type is used to specify whether the frame processor is required to transmit the response part of the LIN frame. |    |   |
| <b>Available via</b> | Lin_GeneralTypes.h   |    |   |

]()

Note: For LIN master nodes, the mapping to the previous definition is as follows:

LIN\_MASTER\_RESPONSE <-> LIN\_FRAMERESPONSE\_TX

LIN\_SLAVE\_RESPONSE <-> LIN\_FRAMERESPONSE\_RX

LIN\_SLAVE\_TO\_SLAVE <-> LIN\_FRAMERESPONSE\_IGNORE

### 8.2.5 Lin\_FrameDType

[SWS\_Lin\_00231]

|                      |  |    |                            |
|----------------------|--|----|----------------------------|
| <b>Name</b>          | Lin_FrameDType   |    |                            |
| <b>Kind</b>          | Type   |    |                            |
| <b>Derived from</b>  | uint8  |    |                            |
| <b>Range</b>         | 1...8  | -- | Data length of a LIN Frame |
| <b>Description</b>   | This type is used to specify the number of SDU data bytes to copy. |    |                            |
| <b>Available via</b> | Lin_GeneralTypes.h   |    |                            |

]()

### 8.2.6 Lin\_PduType

[SWS\_Lin\_00232]

|                 |             |  |  |
|-----------------|-------------|--|--|
| <b>Name</b>     | Lin_PduType |  |  |
| <b>Kind</b>     | Structure   |  |  |
| <b>Elements</b> | Pid         |  |  |

|                      |   |                       |
|----------------------|---|-----------------------|
|                      | <b>Type</b>   | Lin_FramePidType      |
|                      | <b>Comment</b>  | --                    |
|                      | Cs  |                       |
|                      | <b>Type</b>   | Lin_FrameCsModelType  |
|                      | <b>Comment</b>  | --                    |
|                      | Drc   |                       |
|                      | <b>Type</b>   | Lin_FrameResponseType |
|                      | <b>Comment</b>  | --                    |
|                      | DI  |                       |
|                      | <b>Type</b>   | Lin_FrameDIType       |
|                      | <b>Comment</b>  | --                    |
|                      | SduPtr  |                       |
|                      | <b>Type</b>   | uint8*                |
|                      | <b>Comment</b>  | --                    |
| <b>Description</b>   | This Type is used to provide PID, checksum model, data length and SDU pointer from the LIN Interface to the LIN driver. |                       |
| <b>Available via</b> | Lin_GeneralTypes.h  |                       |

l()

Description for each element of Lin\_PduType is given in:

- Section 8.2.2 for Lin\_FramePidType
- Section 8.2.3 for Lin\_FrameCsModelType
- Section 8.2.4 for Lin\_FrameResponseType
- Section 8.2.5 for Lin\_FrameDIType

## 8.2.7 Lin\_StatusType

[SWS\_Lin\_00233]

|              |                |    |  |
|--------------|----------------|----|--|
| <b>Name</b>  | Lin_StatusType |    |  |
| <b>Kind</b>  | Enumeration    |    |  |
| <b>Range</b> | LIN_NOT_OK     | -- | LIN frame operation return value. Development or production error occurred |
|              | LIN_TX_OK      | -- | LIN frame operation return value. Successful transmission.                 |
|              | LIN_TX_BUSY    | -- | LIN frame operation return value. Ongoing transmission                     |

|                      |  |    |   |
|----------------------|--|----|---|
|                      |  |    | (Header or Response).   |
|                      | LIN_TX_HEADER_ERROR  | -- | LIN frame operation return value. Erroneous header transmission such as: <ul style="list-style-type: none"> <li>• Mismatch between sent and read back data</li> <li>• Identifier parity error or</li> <li>• Physical bus error</li> </ul> |
|                      | LIN_TX_ERROR   | -- | LIN frame operation return value. Erroneous response transmission such as: <ul style="list-style-type: none"> <li>• Mismatch between sent and read back data</li> <li>• Physical bus error</li> </ul>                                     |
|                      | LIN_RX_OK  | -- | LIN frame operation return value. Reception of correct response.  |
|                      | LIN_RX_BUSY  | -- | LIN frame operation return value. Ongoing reception: at least one response byte has been received, but the checksum byte has not been received.   |
|                      | LIN_RX_ERROR   | -- | LIN frame operation return value. Erroneous response reception such as: <ul style="list-style-type: none"> <li>• Framing error</li> <li>• Overrun error</li> <li>• Checksum error or</li> <li>• Short response</li> </ul>                 |
|                      | LIN_RX_NO_RESPONSE   | -- | LIN frame operation return value. No response byte has been received so far.  |
|                      | LIN_OPERATIONAL  | -- | LIN channel state return value. Normal operation; the related LIN channel is ready to transmit next header. No data from previous frame available (e.g. after initialization)   |
|                      | LIN_CH_SLEEP   | -- | LIN channel state return value. Sleep state operation; in this state wake-up detection from slave nodes is enabled.   |
| <b>Description</b>   | LIN operation states for a LIN channel or frame, as returned by the API service Lin_GetStatus(). |    |   |
| <b>Available via</b> | Lin_GeneralTypes.h   |    |   |

]()

### 8.2.8 Lin\_SlaveErrorType

[SWS\_Lin\_91140]



|                      |   |    |  |
|----------------------|---|----|--|
| <b>Name</b>          | Lin_SlaveErrorType  |    |  |
| <b>Kind</b>          | Enumeration   |    |  |
| <b>Range</b>         | LIN_ERR_HEADER  | -- | Error in header                                      |
|                      | LIN_ERR_RESP_STOPBIT  | -- | Framing error in response                            |
|                      | LIN_ERR_RESP_CHKSUM   | -- | Checksum error                                       |
|                      | LIN_ERR_RESP_DATABIT  | -- | Monitoring error of transmitted data bit in response |
|                      | LIN_ERR_NO_RESP   | -- | No response  |
|                      | LIN_ERR_INC_RESP  | -- | Incomplete response                                  |
| <b>Description</b>   | This type represents the slave error types that are detected during header reception and response transmission / reception. |    |  |
| <b>Available via</b> | Lin_GeneralTypes.h  |    |  |

]()

## 8.3 Function definitions

This is a list of functions provided for upper layer modules.

### 8.3.1 Services affecting the complete LIN hardware unit

#### 8.3.1.1 Lin\_Init

[SWS\_Lin\_00006]

|                           |   |  |
|---------------------------|---|--|
| <b>Service Name</b>       | Lin_Init  |  |
| <b>Syntax</b>             | <pre>void Lin_Init (     const Lin_ConfigType* Config )</pre> |  |
| <b>Service ID [hex]</b>   | 0x00  |  |
| <b>Sync/Async</b>         | Synchronous   |  |
| <b>Reentrancy</b>         | Non Reentrant   |  |
| <b>Parameters (in)</b>    | Config  | Pointer to LIN driver configuration set. |
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | None  |  |

|                      |                             |
|----------------------|-----------------------------|
| <b>Return value</b>  | None                        |
| <b>Description</b>   | Initializes the LIN module. |
| <b>Available via</b> | Lin.h                       |

](SRS\_BSW\_00406, SRS\_BSW\_00101)

**[SWS\_Lin\_00084]** 「The function Lin\_Init shall initialize the Lin module (i.e. static variables, including flags and LIN HW Unit global hardware settings), as well as the LIN channels. 」()

Different sets of static configuration may have been configured.

**[SWS\_Lin\_00150]** 「The function Lin\_Init shall initialize the module according to the configuration set pointed to by the parameter Config.」()

**[SWS\_Lin\_00008]** 「The function Lin\_Init shall invoke initializations for relevant hardware register settings common to all channels available on the LIN hardware unit.」( SRS\_Lin\_01556)

**[SWS\_Lin\_00190]** 「The function Lin\_Init shall also invoke initializations for LIN channel specific settings.」( SRS\_Lin\_01556)

**[SWS\_Lin\_00106]** 「The Lin module's environment shall not call any function of the Lin module before having called Lin\_Init except Lin\_GetVersionInfo.」()

**[SWS\_Lin\_00099]** 「If development error detection for the Lin module is enabled: the function Lin\_Init shall check the parameter Config for being within the allowed range. If Config is not in the allowed range, the function Lin\_Init shall raise the development error LIN\_E\_INVALID\_POINTER.」()

**[SWS\_Lin\_00105]** 「If development error detection for the Lin module is enabled: the function Lin\_Init shall check the Lin driver for being in the state LIN\_UNINIT. If the Lin driver is not in the state LIN\_UNINIT, the function Lin\_Init shall raise the development error LIN\_E\_STATE\_TRANSITION.」()

**[SWS\_Lin\_00097]** 「If a change to the LIN hardware control registers results in the need to wait for a status change, this shall be protected by a configurable time out mechanism (LinTimeoutDuration). If such a time out is detected the LIN\_E\_TIMEOUT error shall be raised to the [DET](#) or [DEM](#). This situation should only

arise in the event of a LIN hardware unit fault, and should be communicated to the rest of the system. ]()

A LIN\_E\_TIMEOUT will affect the complete LIN stack in a way that the LIN driver must be re-initialized or the LIN functionality must be switched off.

### 8.3.1.2 Lin\_CheckWakeup

#### [SWS\_Lin\_00160]

|                           |   |  |
|---------------------------|---|--|
| <b>Service Name</b>       | Lin_CheckWakeup   |  |
| <b>Syntax</b>             | Std_ReturnType Lin_CheckWakeup (<br>uint8 Channel<br>)                      |  |
| <b>Service ID [hex]</b>   | 0x0a  |  |
| <b>Sync/Async</b>         | Synchronous   |  |
| <b>Reentrancy</b>         | Non Reentrant   |  |
| <b>Parameters (in)</b>    | Channel   | LIN channel to be addressed  |
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | None  |  |
| <b>Return value</b>       | Std_Return-Type   | E_OK: No error has occurred during execution of the API<br>E_NOT_OK: An error has occurred during execution of the API |
| <b>Description</b>        | This function checks if a wakeup has occurred on the addressed LIN channel. |  |
| <b>Available via</b>      | Lin.h   |  |

]()

There are two methods in which wake up detection shall happen, one is from LIN controller hardware [Micro peripheral device] and/or another from LinTranceiver.

After a wake up caused by LIN bus Transceiver the function Lin\_CheckWakeup will be called by the LIN Interface module to identify the corresponding LIN channel (e.g. in case of multiple transceivers are physically connected to one MCU wake up pin) (see SWS\_LinIf\_00503). In this case, LIN Driver only plays a role on validation of this wake up signal.

**[SWS\_Lin\_00098]** 「The function Lin\_CheckWakeup shall evaluate the wakeup on the addressed LIN channel. When a wake-up event on the addressed LIN channel (e.g. RxD pin has constant low level) is detected, the function Lin\_CheckWakeup shall notify the ECU State Manager module immediately via the

EcuM\_SetWakeupEvent and the Lin Interface module via LinIf\_WakeupConfirmation callback function. (SRS\_BSW\_00375, SRS\_Lin\_01563)

**[SWS\_Lin\_00251]** If development error detection for the LIN module is enabled: if the channel parameter is invalid, the function Lin\_CheckWakeup shall raise the development error LIN\_E\_INVALID\_CHANNEL otherwise (if DET is disabled) return E\_NOT\_OK.()

**[SWS\_Lin\_00107]** If development error detection for the LIN module is enabled: if the function Lin\_CheckWakeup is called before the LIN module was initialized, the function Lin\_CheckWakeup shall raise the development error LIN\_E\_UNINIT.()

### 8.3.1.3 Lin\_GetVersionInfo

**[SWS\_Lin\_00161]**

|                           |   |  |
|---------------------------|---|--|
| <b>Service Name</b>       | Lin_GetVersionInfo  |  |
| <b>Syntax</b>             | <pre>void Lin_GetVersionInfo (     Std_VersionInfoType* versioninfo )</pre> |  |
| <b>Service ID [hex]</b>   | 0x01  |  |
| <b>Sync/Async</b>         | Synchronous   |  |
| <b>Reentrancy</b>         | Reentrant   |  |
| <b>Parameters (in)</b>    | None  |  |
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | versioninfo   | Pointer to where is stored the version information of this module. |
| <b>Return value</b>       | None  |  |
| <b>Description</b>        | Returns the version information of this module.                             |  |
| <b>Available via</b>      | Lin.h   |  |

()

**[SWS\_Lin\_00001]** The function Lin\_GetVersionInfo shall return the version information of the LIN module. The version information includes:

- Two bytes for the vendor ID
- Two 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. The AUTOSAR specification version number is

checked during compile time and therefore not required in this API. (SRS\_BSW\_00407)

**[SWS\_Lin\_00248]** If development error detection for the LIN module is enabled: If the parameter versioninfo is a NULL pointer, the function Lin\_GetVersionInfo shall raise the error LIN\_E\_PARAM\_POINTER.()

### 8.3.2 Services affecting a single LIN channel

#### 8.3.2.1 Lin\_SendFrame

Note: This service is only applicable for LIN master node (available only if the ECU has any LIN master channel).

**[SWS\_Lin\_00191]**

|                           |   |  |
|---------------------------|---|--|
| <b>Service Name</b>       | Lin_SendFrame   |  |
| <b>Syntax</b>             | <pre>Std_ReturnType Lin_SendFrame (     uint8 Channel,     const Lin_PduType* PduInfoPtr )</pre>  |  |
| <b>Service ID [hex]</b>   | 0x04  |  |
| <b>Sync/Async</b>         | Asynchronous  |  |
| <b>Reentrancy</b>         | Non Reentrant   |  |
| <b>Parameters (in)</b>    | Channel   | LIN channel to be addressed  |
|                           | PduInfoPtr  | Pointer to PDU containing the PID, checksum model, response type, DI and SDU data pointer  |
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | None  |  |
| <b>Return value</b>       | Std_Return-Type   | E_OK: Send command has been accepted.<br>E_NOT_OK: Send command has not been accepted, development or production error occurred. |
| <b>Description</b>        | Sends a LIN header and a LIN response, if necessary. The direction of the frame response (master response, slave response, slave-to-slave communication) is provided by the PduInfoPtr. Only used for LIN master nodes. |  |
| <b>Available via</b>      | Lin.h   |  |

]()

**[SWS\_Lin\_00192]** [The function `Lin_SendFrame` shall send the header part (Break Field, Synch Byte Field and PID Field) and, depending on the direction of the frame response, a complete LIN response part of a LIN frame on the addressed LIN channel.]()

**[SWS\_Lin\_00193]** [In case of receiving data the LIN Interface has to wait for the corresponding response part of the LIN frame by polling with the function `Lin_GetStatus()` after using the function `Lin_SendFrame()`.]()

**[SWS\_Lin\_00194]** [The Lin module's environment shall only call `Lin_SendFrame` on a channel which is in state `LIN_CH_OPERATIONAL` or in one of the sub-states of `LIN_CH_OPERATIONAL`.]()

**[SWS\_Lin\_00239]** [In case of errors during header transmission, it is up to the implementer how to handle these errors (stop/continue transmission) and to decide if the corresponding response is valid or not. ]()

**[SWS\_Lin\_00240]** [In case of response transmission errors, the ISO 17987 specifications describe within the frame processor state machine how to handle such errors. It is stated that a mismatch between sent and readback data shall be detected not later than after the completion of the byte field containing the mismatch. Furthermore, ISO 17987 specifications specify that the transmission shall be aborted. ]()

**[SWS\_Lin\_00195]** [If development error detection for the LIN module is enabled: if the function `Lin_SendFrame` is called before the LIN module was initialized, the function `Lin_SendFrame` shall raise the development error `LIN_E_UNINIT` otherwise (if DET is disabled) return `E_NOT_OK`.]()

**[SWS\_Lin\_00197]** [If development error detection for the LIN module is enabled: if the channel parameter is invalid, the function `Lin_SendFrame` shall raise the development error `LIN_E_INVALID_CHANNEL` otherwise (if DET is disabled) return `E_NOT_OK`.]()

**[SWS\_Lin\_00198]** [If development error detection for the LIN module is enabled: the function `Lin_SendFrame` shall check the parameter `PduInfoPtr` for not being a NULL pointer. If `PduInfoPtr` is a NULL pointer, the function `Lin_SendFrame` shall raise the development error `LIN_E_PARAM_POINTER` otherwise (if DET is disabled) return `E_NOT_OK`.]()

**[SWS\_Lin\_00199]** ⌈If development error detection for the LIN module is enabled: if the LIN channel state-machine is in the state LIN\_CH\_SLEEP, the function Lin\_SendFrame shall raise the development error LIN\_E\_STATE\_TRANSITION otherwise (if DET is disabled) return E\_NOT\_OK.⌋()

**[SWS\_Lin\_00287]** ⌈The function Lin\_SendFrame is only available if the Lin module is configured as LIN master node on at least one channel. In a pure LIN slave configuration, this function is not available. This depends on the configuration parameters LinNodeType. ⌋()

### 8.3.2.2 Lin\_GoToSleep

Note: This service is only applicable for LIN master node (available only if the ECU has any LIN master channel).

**[SWS\_Lin\_00166]**⌈

|                           |   |  |
|---------------------------|---|--|
| <b>Service Name</b>       | Lin_GoToSleep   |  |
| <b>Syntax</b>             | Std_ReturnType Lin_GoToSleep (<br>uint8 Channel<br>)  |  |
| <b>Service ID [hex]</b>   | 0x06  |  |
| <b>Sync/Async</b>         | Asynchronous  |  |
| <b>Reentrancy</b>         | Non Reentrant   |  |
| <b>Parameters (in)</b>    | Channel   | LIN channel to be addressed  |
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | None  |  |
| <b>Return value</b>       | Std_ReturnType  | E_OK: Sleep command has been accepted<br>E_NOT_OK: Sleep command has not been accepted, development or production error occurred |
| <b>Description</b>        | The service instructs the driver to transmit a go-to-sleep-command on the addressed LIN channel.<br>Only used for LIN master nodes. |  |
| <b>Available via</b>      | Lin.h   |  |

⌋()

**[SWS\_Lin\_00089]** ⌈The function Lin\_GoToSleep shall send a go-to-sleep-command on the addressed LIN channel as defined in LIN Specification 2.1.⌋()

**[SWS\_Lin\_00266]** ⌈ The function `Lin_GoToSleep` shall set the channel state to `LIN_CH_SLEEP_PENDING`, even in case of an erroneous transmission of the go-to-sleep-command. ⌋(SRS\_Lin\_01566)

**[SWS\_Lin\_00220]** ⌈ If external wake-up detection is supported by configuration parameter `LinChannelWakeupSupport`, then the function `Lin_GoToSleep` shall enable bus monitoring for a wake-up request on that channel, even in case of an erroneous transmission of the go-to-sleep command. ⌋()

**[SWS\_Lin\_00221]** ⌈ The function `Lin_GoToSleep` shall optionally set the LIN hardware unit to reduced power operation mode (if supported by HW), even in case of an erroneous transmission of the go-to-sleep-command. ⌋()

**[SWS\_Lin\_00255]** ⌈ The LIN channel shall enter the state `LIN_CH_SLEEP` the next time `Lin_GetStatus` is called, independent of the success of the transmission of the goto-sleep-command on the bus. ⌋()

**[SWS\_Lin\_00074]** ⌈ The function `Lin_GoToSleep` shall terminate ongoing frame transmission of prior transmission requests, even if the transmission is unsuccessfully completed. ⌋()

**[SWS\_Lin\_00129]** ⌈ If development error detection for the LIN module is enabled: if the function `Lin_GoToSleep` is called before the LIN module was initialized, the function `Lin_GoToSleep` shall raise the development error `LIN_E_UNINIT`. ⌋()

**[SWS\_Lin\_00131]** ⌈ If development error detection for the LIN module is enabled: the function `Lin_GoToSleep` shall raise the development error `LIN_E_INVALID_CHANNEL` if the channel parameter is invalid. ⌋()

**[SWS\_Lin\_00288]** ⌈ The function `Lin_GotoSleep` is only available if the Lin module is configured as LIN master node on at least one channel. In a pure LIN slave configuration, this function is not available. This depends on the configuration parameters `LinNodeType`. ⌋()

### 8.3.2.3 Lin\_GoToSleepInternal

**[SWS\_Lin\_00167]** ⌈

|                     |   |
|---------------------|---|
| <b>Service Name</b> | <code>Lin_GoToSleepInternal</code>  |
| <b>Syntax</b>       | <code>Std_ReturnType Lin_GoToSleepInternal (</code><br><code>uint8 Channel</code> |



|                           |   |  |
|---------------------------|---|--|
|                           | )   |  |
| <b>Service ID [hex]</b>   | 0x09  |  |
| <b>Sync/Async</b>         | Synchronous   |  |
| <b>Reentrancy</b>         | Non Reentrant   |  |
| <b>Parameters (in)</b>    | Channel   | LIN channel to be addressed  |
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | None  |  |
| <b>Return value</b>       | Std_ReturnType  | E_OK: Command has been accepted<br>E_NOT_OK: Command has not been accepted, development or production error occurred |
| <b>Description</b>        | Sets the channel state to LIN_CH_SLEEP, enables the wake-up detection and optionally sets the LIN hardware unit to reduced power operation mode (if supported by HW). |  |
| <b>Available via</b>      | Lin.h   |  |

]()

**[SWS\_Lin\_00095]** [The function Lin\_GoToSleepInternal shall set the channel state to LIN\_CH\_SLEEP.]()

**[SWS\_Lin\_00222]** [If external wake-up detection is supported by configuration parameter LinChannelWakeupSupport, then the function Lin\_GoToSleepInternal shall enable bus monitoring for a wake-up request on that channel.]()

**[SWS\_Lin\_00223]** [The function Lin\_GoToSleepInternal shall optionally set the LIN hardware unit to reduced power operation mode (if supported by HW).]()

**[SWS\_Lin\_00133]** [If development error detection for the LIN module is enabled: if the function Lin\_GoToSleepInternal is called before the LIN module was initialized, the function Lin\_GoToSleepInternal shall raise the development error LIN\_E\_UNINIT.]()

**[SWS\_Lin\_00135]** [If development error detection for the LIN module is enabled: the function Lin\_GoToSleepInternal shall raise the development error LIN\_E\_INVALID\_CHANNEL if the channel parameter is invalid.]()

### 8.3.2.4 Lin\_Wakeup

#### [SWS\_Lin\_00169]

|                           |   |  |
|---------------------------|---|--|
| <b>Service Name</b>       | Lin_Wakeup  |  |
| <b>Syntax</b>             | Std_ReturnType Lin_Wakeup (<br>uint8 Channel<br>)                           |  |
| <b>Service ID [hex]</b>   | 0x07  |  |
| <b>Sync/Async</b>         | Asynchronous  |  |
| <b>Reentrancy</b>         | Non Reentrant   |  |
| <b>Parameters (in)</b>    | Channel   | LIN channel to be addressed  |
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | None  |  |
| <b>Return value</b>       | Std_ReturnType  | E_OK: Wake-up request has been accepted<br>E_NOT_OK: Wake-up request has not been accepted, development or production error occurred |
| <b>Description</b>        | Generates a wake up pulse and sets the channel state to LIN_CH_OPERATIONAL. |  |
| <b>Available via</b>      | Lin.h   |  |

]()

**[SWS\_Lin\_00137]** ⌈If development error detection for the LIN module is enabled: if the function Lin\_Wakeup is called before the LIN module was initialized, the function Lin\_Wakeup shall raise the development error LIN\_E\_UNINIT. ]()

**[SWS\_Lin\_00139]** ⌈If development error detection for the LIN module is enabled: the function Lin\_Wakeup shall raise the development error LIN\_E\_INVALID\_CHANNEL if the channel parameter is invalid or the channel is inactive. ]()

**[SWS\_Lin\_00140]** ⌈If development error detection for the LIN module is enabled: the function Lin\_Wakeup shall raise the development error LIN\_E\_STATE\_TRANSITION if the LIN channel state-machine is not in the state LIN\_CH\_SLEEP. ]()

**Note:** The Lin driver's environment shall only call Lin\_Wakeup when the LIN channel is in state LIN\_CH\_SLEEP.

### 8.3.2.5 Lin\_WakeupInternal

#### [SWS\_Lin\_00256]

|                           |  |  |
|---------------------------|--|--|
| <b>Service Name</b>       | Lin_WakeupInternal   |  |
| <b>Syntax</b>             | Std_ReturnType Lin_WakeupInternal (<br>uint8 Channel<br>)                        |  |
| <b>Service ID [hex]</b>   | 0x0b   |  |
| <b>Sync/Async</b>         | Asynchronous   |  |
| <b>Reentrancy</b>         | Non Reentrant  |  |
| <b>Parameters (in)</b>    | Channel  | LIN channel to be addressed  |
| <b>Parameters (inout)</b> | None   |  |
| <b>Parameters (out)</b>   | None   |  |
| <b>Return value</b>       | Std_Return-Type  | E_OK: Wake-up request has been accepted<br>E_NOT_OK: Wake-up request has not been accepted, development or production error occurred |
| <b>Description</b>        | Sets the channel state to LIN_CH_OPERATIONAL without generating a wake up pulse. |  |
| <b>Available via</b>      | Lin.h  |  |

]()

[SWS\_Lin\_00257] [The function Lin\_WakeupInternal sets the addressed LIN channel to state LIN\_CH\_OPERATIONAL without generating a wake up pulse.]()

[SWS\_Lin\_00258] [If development error detection for the LIN module is enabled: if the function Lin\_WakeupInternal is called before the LIN module was initialized, the function Lin\_WakeupInternal shall raise the development error LIN\_E\_UNINIT.]()

[SWS\_Lin\_00259] [If development error detection for the LIN module is enabled: the function Lin\_WakeupInternal shall raise the development error LIN\_E\_INVALID\_CHANNEL if the channel parameter is invalid or the channel is inactive.]()

[SWS\_Lin\_00260] [If development error detection for the LIN module is enabled: the function Lin\_WakeupInternal shall raise the development error LIN\_E\_STATE\_TRANSITION if the LIN channel state-machine is not in the state LIN\_CH\_SLEEP.]()

**Note:** The Lin driver's environment shall only call Lin\_WakeupInternal when the LIN channel is in state LIN\_CH\_SLEEP.

### 8.3.2.6 Lin\_GetStatus

Note: This service is only applicable for LIN master node (available only if the ECU has any LIN master channel).

#### [SWS\_Lin\_00168]

|                           |   |   |
|---------------------------|---|---|
| <b>Service Name</b>       | Lin_GetStatus   |   |
| <b>Syntax</b>             | <pre>Lin_StatusType Lin_GetStatus (     uint8 Channel,     const uint8** Lin_SduPtr )</pre> |   |
| <b>Service ID [hex]</b>   | 0x08  |   |
| <b>Sync/Async</b>         | Synchronous   |   |
| <b>Reentrancy</b>         | Non Reentrant   |   |
| <b>Parameters (in)</b>    | Channel   | LIN channel to be checked   |
| <b>Parameters (inout)</b> | None  |   |
| <b>Parameters (out)</b>   | Lin_SduPtr  | Pointer to pointer to a shadow buffer or memory mapped LIN Hardware receive buffer where the current SDU is stored.   |
| <b>Return value</b>       | Lin_Status-<br>Type   | <p>LIN_NOT_OK: Development or production error occurred</p> <p>LIN_TX_OK: Successful transmission</p> <p>LIN_TX_BUSY: Ongoing transmission (Header or Response)</p> <p>LIN_TX_HEADER_ERROR: Erroneous header transmission such as:</p> <ul style="list-style-type: none"> <li>- Mismatch between sent and read back data</li> <li>- Identifier parity error or Physical bus error</li> </ul> <p>LIN_TX_ERROR: Erroneous response transmission such as:</p> <ul style="list-style-type: none"> <li>- Mismatch between sent and read back data</li> <li>- Physical bus error</li> </ul> <p>LIN_RX_OK: Reception of correct response</p> <p>LIN_RX_BUSY: Ongoing reception: at least one response byte has been received, but the checksum byte has not been received</p> <p>LIN_RX_ERROR: Erroneous response reception such as:</p> <ul style="list-style-type: none"> <li>- Framing error</li> <li>- Overrun error</li> <li>- Checksum error or Short response</li> </ul> <p>LIN_RX_NO_RESPONSE: No response byte has been received so far</p> <p>LIN_OPERATIONAL: Normal operation; the related LIN channel is woken up from the LIN_CH_SLEEP and no data has been sent.</p> <p>LIN_CH_SLEEP: Sleep state operation; in this state wake-up detection from slave nodes is enabled.</p> |

|                      |   |
|----------------------|---|
| <b>Description</b>   | Gets the status of the LIN driver.<br>Only used for LIN master nodes. |
| <b>Available via</b> | Lin.h   |

]()

**[SWS\_Lin\_00091]** [The function `Lin_GetStatus` shall return the current transmission, reception or operation status of the LIN driver.]()

**[SWS\_Lin\_00200]** [The return states `LIN_TX_OK`, `LIN_TX_BUSY`, `LIN_TX_HEADER_ERROR`, `LIN_TX_ERROR`, `LIN_RX_OK`, `LIN_RX_BUSY`, `LIN_RX_ERROR`, `LIN_RX_NO_RESPONSE` and `LIN_OPERATIONAL` are sub-states of the channel state `LIN_CH_OPERATIONAL`.]()

**[SWS\_Lin\_00092]** [If a SDU has been successfully received, the function `Lin_GetStatus` shall store the SDU in a shadow buffer or memory mapped LIN Hardware receive buffer referenced by `Lin_SduPtr`. The buffer will only be valid and must be read until the next [Lin\\_SendFrame](#) function call.]()

**[SWS\_Lin\_00238]** [The function `Lin_GetStatus` shall return `LIN_TX_OK`, when

- A Master Response Type frame is send and LIN header as well as LIN response of the frame are transmitted successfully or
- A Slave to Slave Response Type frame is send and the LIN header of the frame is transmitted successfully.]()

**[SWS\_Lin\_00141]** [If development error detection for the LIN module is enabled: if the function `Lin_GetStatus` is called before the LIN module was initialized, the function `Lin_GetStatus` shall raise the development error `LIN_E_UNINIT` otherwise (if DET is disabled) return `LIN_NOT_OK`.]()

**[SWS\_Lin\_00143]** [If development error detection for the LIN module is enabled: if the channel parameter is invalid or the channel is inactive, the function `Lin_GetStatus` shall raise the development error `LIN_E_INVALID_CHANNEL` otherwise (if DET is disabled) return `LIN_NOT_OK`.]()

**[SWS\_Lin\_00144]** [If development error detection for the LIN module is enabled: the function `Lin_GetStatus` shall check the parameter `Lin_SduPtr` for not being a NULL pointer. If `Lin_SduPtr` is a NULL pointer, the function `Lin_GetStatus` shall raise the development error `LIN_E_PARAM_POINTER` otherwise (if DET is disabled) return `LIN_NOT_OK`.]()

**[SWS\_Lin\_00289]** 「The function Lin\_GetStatus is only available if the Lin module is configured as LIN master node on at least one channel. In a pure LIN slave configuration, this function is not available. This depends on the configuration parameters LinNodeType. 」()

## 8.4 Call-back notifications

There are no callback functions within the LIN driver.

## 8.5 Scheduled functions

There are no scheduled functions within the LIN driver

## 8.6 Expected Interfaces

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

### 8.6.1 Mandatory Interfaces

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

**[SWS\_Lin\_00234]**

| <i>API Function</i>       | <i>Header File</i> | <i>Description</i>  |
|---------------------------|--------------------|---|
| Dem_Set-EventStatus       | Dem.h              | Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEventReportingType} == STANDARD_REPORTING) |
| EcuM_Set-WakeupEvent      | EcuM.h             | Sets the wakeup event.  |
| LinIf_Wakeup-Confirmation | LinIf.h            | The LIN Driver or LIN Transceiver Driver will call this function to report the wake up source after the successful wakeup detection during Check Wakeup or after power on by bus.   |

()

### 8.6.2 Optional Interfaces

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

**[SWS\_Lin\_00235]**

| <b>API Function</b>       | <b>Header File</b> | <b>Description</b>  |
|---------------------------|--------------------|---|
| Det_Report-Error          | Det.h              | Service to report development errors.   |
| EcuM_Check-Wakeup         | EcuM.h             | This function can be called to check the given wakeup sources. It will pass the argument to the integrator function EcuM_CheckWakeupHook. It can also be called by the ISR of a wakeup source to set up the PLL and check other wakeup sources that may be connected to the same interrupt. |
| Icu_Disable-Notification  | Icu.h              | This function disables the notification of a channel.   |
| Icu_Enable-Notification   | Icu.h              | This function enables the notification on the given channel.  |
| LinIf_Header-Indication   | LinIf.h            | The LIN Driver will call this function to report a received LIN header. This function is only applicable for LIN slave nodes (available only if the ECU has any LIN slave channel).   |
| LinIf_LinError-Indication | LinIf.h            | The LIN Driver will call this function to report a detected error event during header or response processing. This function is only applicable for LIN slave nodes (available only if the ECU has any LIN slave channel).   |
| LinIf_Rx-Indication       | LinIf.h            | The LIN Driver will call this function to report a successfully received response and provides the reception data to the LIN Interface. This function is only applicable for LIN slave nodes (available only if the ECU has any LIN slave channel).   |
| LinIf_Tx-Confirmation     | LinIf.h            | The LIN Driver will call this function to report a successfully transmitted response. This function is only applicable for LIN slave nodes (available only if the ECU has any LIN slave channel).   |

]()

**[SWS\_Lin\_00176]** [The Lin module shall invoke the callback function EcuM\_CheckWakeup from within the wake-up ISR of the corresponding LIN channel when a valid LIN wake-up pulse has been detected.]()

Restrictions:

A wake-up ISR can only be raised if supported by the LIN hardware. Therefore, EcuM\_CheckWakeup is supported if at least for one channel wake-up is supported (see configuration parameter LinChannelWakeUpSupport).

### 8.6.3 Configurable interfaces

There is no configurable target for the LIN driver. The LIN driver always reports to LIN interface.

## 9 Sequence diagrams

Complete sequence diagrams for transmission, reception and error handling can be found in the LIN Interface Specification [8].

### 9.1 Receiving a LIN Frame

#### 9.1.1 LIN Master

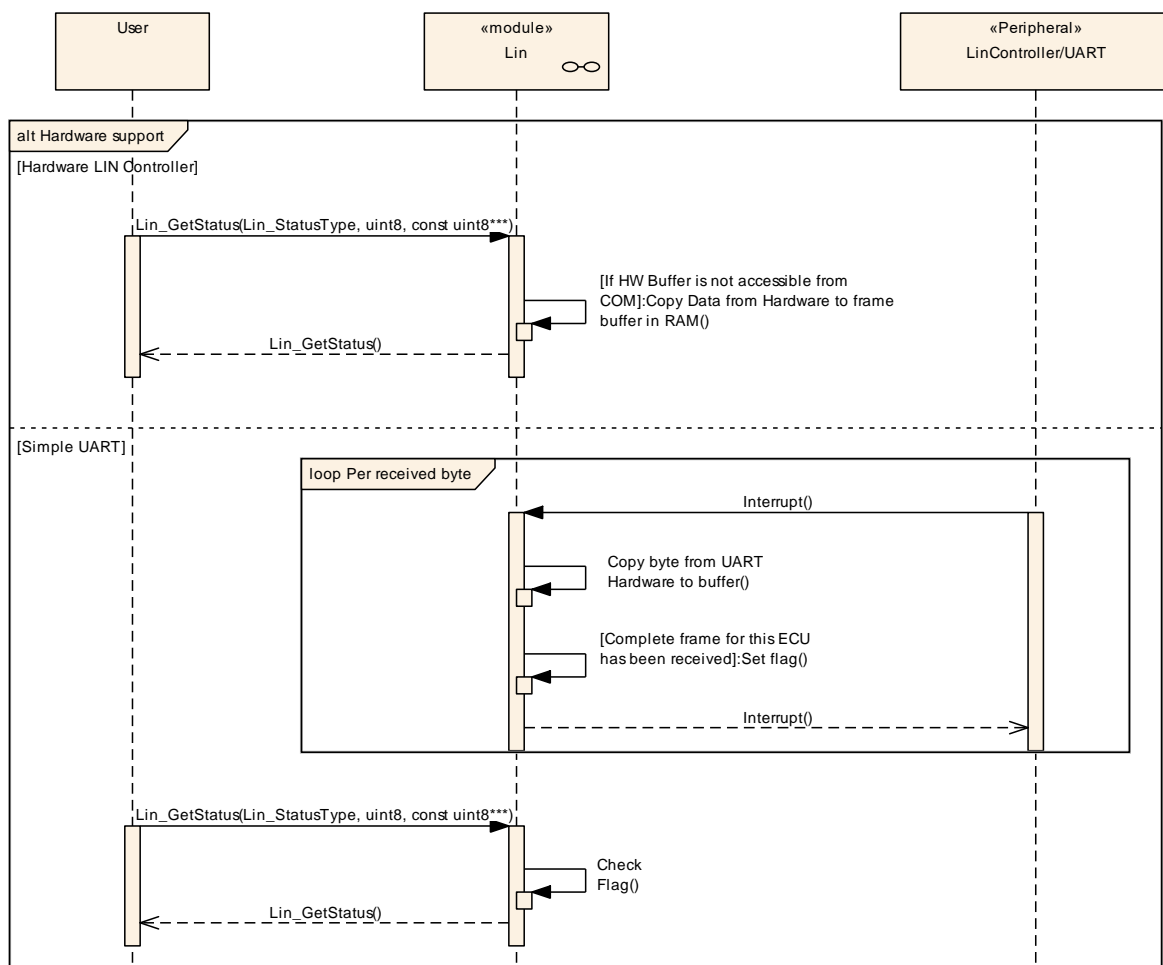
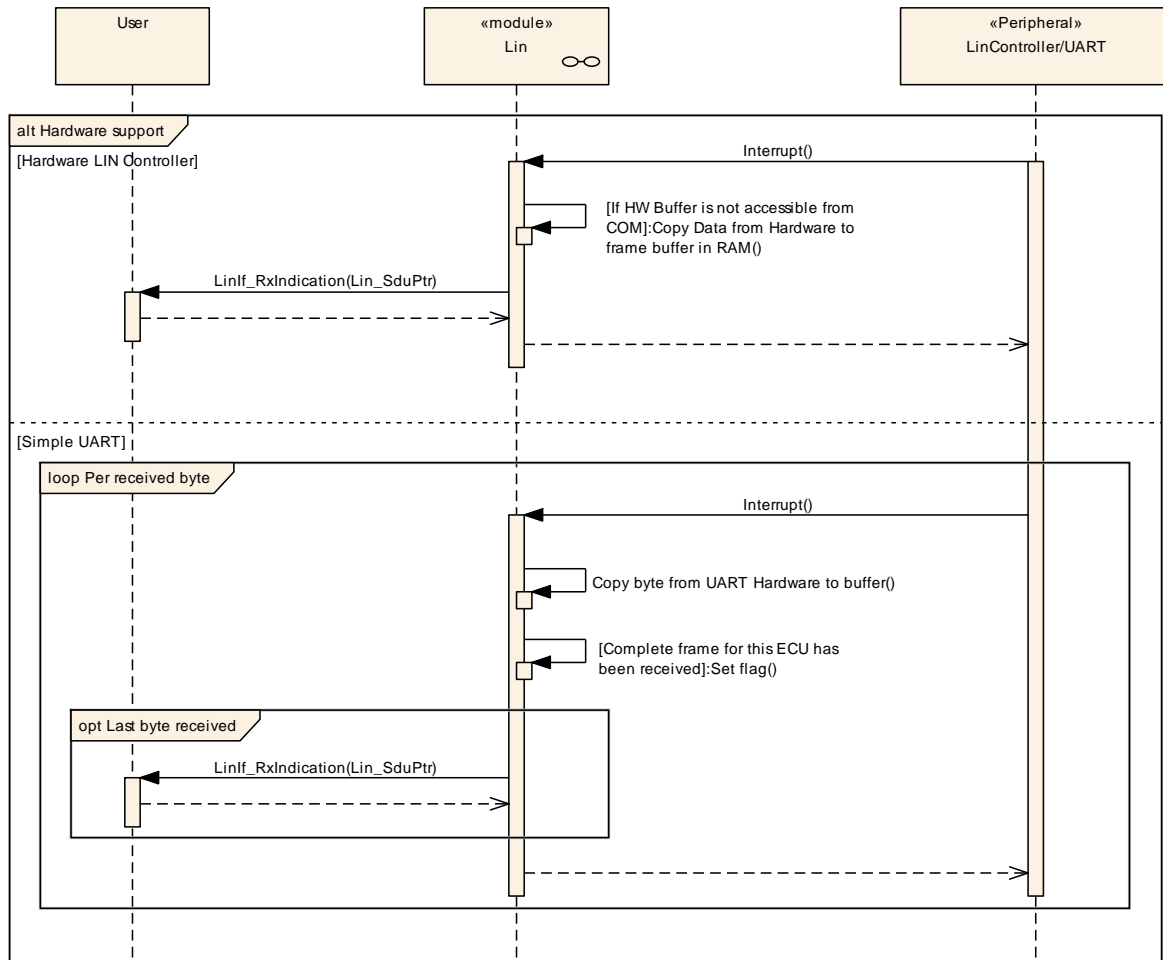


Figure 9-1: LIN Frame Receiving Sequence Chart for LIN Master



**9.1.2 LIN Slave**



**Figure 9-2: LIN Frame Receiving Sequence Chart for LIN Slave**

## 10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals.

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

Chapter 10.3 specifies published information of the module LIN driver.

### 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 described parameters are inputs for the LIN driver configurator.

**[SWS\_Lin\_00029]** 「The code configurator of the LIN driver is LIN hardware Unit specific.」(SRS\_BSW\_00159)

**[SWS\_Lin\_00039]** 「Values that can be configured are hardware dependent. Therefore, the rules and constraints cannot be given in the standard. 」(SRS\_BSW\_00167)

**[SWS\_Lin\_00224]** 「The configuration tool is responsible to do a static configuration checking, also regarding dependencies between modules (e.g. Port driver, MCU driver etc.)」()

**[SWS\_Lin\_00269]** 「The Lin Driver module shall reject configurations with partition mappings which are not supported by the implementation.」()

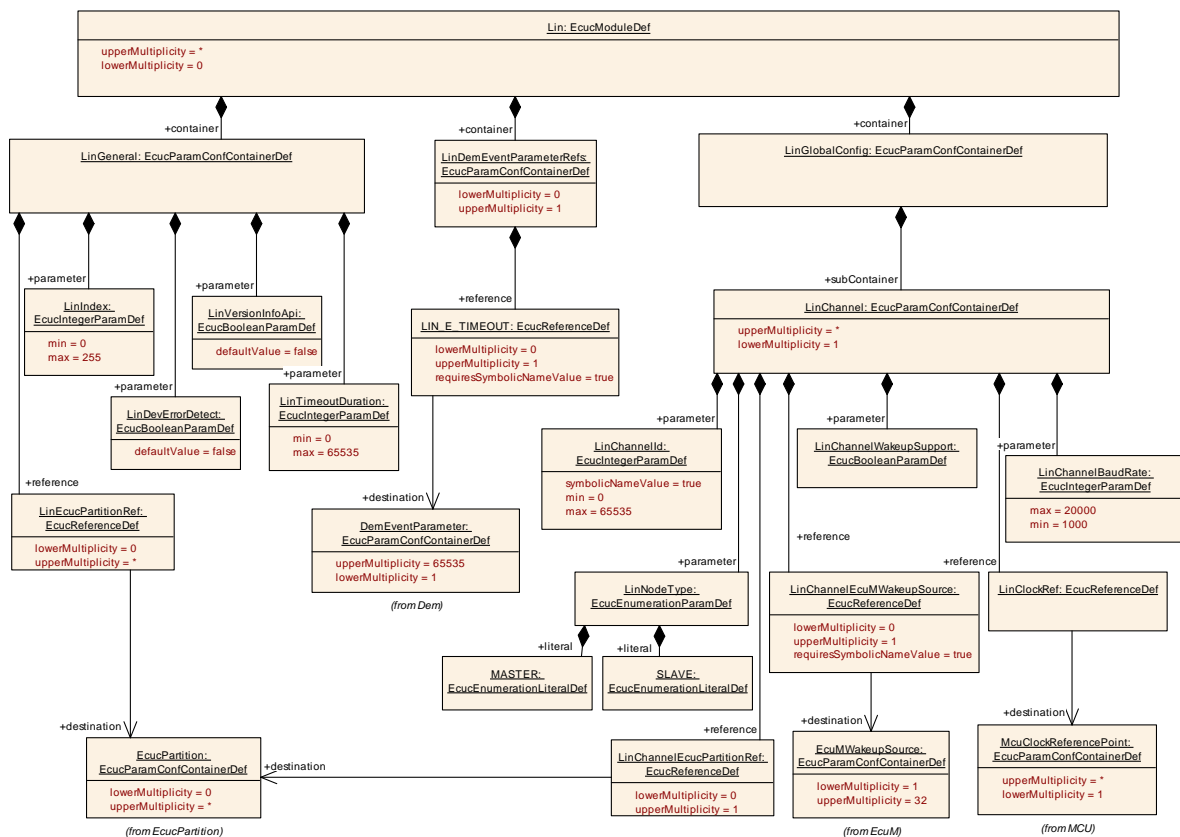


Figure 10-1: Configuration structure for the LIN driver

### 10.2.1 Lin

|                                   |   |
|-----------------------------------|---|
| <b>SWS Item</b>                   | <b>ECUC_Lin_00190 :</b>                       |
| <b>Module Name</b>                | <i>Lin</i>                                    |
| <b>Module Description</b>         | Configuration of the Lin (LIN driver) module. |
| <b>Post-Build Variant Support</b> | true  |
| <b>Supported Config Variants</b>  | VARIANT-POST-BUILD, VARIANT-PRE-COMPILE       |

| Included Containers      |              |  |
|--------------------------|--------------|--|
| Container Name           | Multiplicity | Scope / Dependency   |
| LinDemEventParameterRefs | 0..1         | Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references. |
| LinGeneral               | 1            | This container contains the parameters related to each LIN Driver Unit.  |
| LinGlobalConfig          | 1            | This container contains the global configuration parameter of the Lin driver.  |

## 10.2.2 LinGeneral

|                                 |   |
|---------------------------------|---|
| <b>SWS Item</b>                 | <b>ECUC_Lin_00183 :</b>   |
| <b>Container Name</b>           | LinGeneral  |
| <b>Parent Container</b>         | Lin   |
| <b>Description</b>              | This container contains the parameters related to each LIN Driver Unit. |
| <b>Configuration Parameters</b> |   |

|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Lin_00066 :</b>   |    |              |
| <b>Name</b>                      | LinDevErrorDetect   |    |              |
| <b>Parent Container</b>          | LinGeneral  |    |              |
| <b>Description</b>               | Switches the development error detection and notification on or off. <ul style="list-style-type: none"> <li>• true: detection and notification is enabled.</li> <li>• false: detection and notification is disabled.</li> </ul> |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucBooleanParamDef   |    |              |
| <b>Default value</b>             | false   |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                                  | <b>Link time</b>  | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: local  |    |              |

|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Lin_00179 :</b>   |    |              |
| <b>Name</b>                      | LinIndex  |    |              |
| <b>Parent Container</b>          | LinGeneral  |    |              |
| <b>Description</b>               | Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0. |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucIntegerParamDef   |    |              |
| <b>Range</b>                     | 0 .. 255  |    |              |
| <b>Default value</b>             | --  |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                                  | <b>Link time</b>  | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: local  |    |              |

|                                  |  |    |              |
|----------------------------------|--|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Lin_00093 :</b>  |    |              |
| <b>Name</b>                      | LinTimeoutDuration   |    |              |
| <b>Parent Container</b>          | LinGeneral   |    |              |
| <b>Description</b>               | Specifies the maximum number of loops for blocking function until a timeout is raised in short term wait loops |    |              |
| <b>Multiplicity</b>              | 1  |    |              |
| <b>Type</b>                      | EcucIntegerParamDef  |    |              |
| <b>Range</b>                     | 0 .. 65535   |    |              |
| <b>Default value</b>             | --   |    |              |
| <b>Post-Build Variant Value</b>  | false  |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|                                  | <b>Link time</b>   | -- |              |
|                                  | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b>        | scope: local   |    |              |

|                 |                         |  |  |
|-----------------|-------------------------|--|--|
| <b>SWS Item</b> | <b>ECUC_Lin_00067 :</b> |  |  |
|-----------------|-------------------------|--|--|

|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>Name</b>                      | LinVersionInfoApi                                   |    |              |
| <b>Parent Container</b>          | LinGeneral  |    |              |
| <b>Description</b>               | Switches the Lin_GetVersionInfo function ON or OFF. |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucBooleanParamDef                                 |    |              |
| <b>Default value</b>             | false   |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                             | X  | All Variants |
|                                  | <b>Link time</b>                                    | -- |              |
|                                  | <b>Post-build time</b>                              | -- |              |
| <b>Scope / Dependency</b>        | scope: local  |    |              |

|   |  |    |              |
|---|--|----|--------------|
| <b>SWS Item</b>                         | <b>ECUC_Lin_00192 :</b>  |    |              |
| <b>Name</b>                             | LinEcucPartitionRef  |    |              |
| <b>Parent Container</b>                 | LinGeneral   |    |              |
| <b>Description</b>                      | Maps the Lin driver to zero or multiple ECUC partitions to make the modules API available in this partition. The Lin driver will operate as an independent instance in each of the partitions. |    |              |
| <b>Multiplicity</b>                     | 0..*   |    |              |
| <b>Type</b>                             | Reference to [ EcucPartition ]   |    |              |
| <b>Post-Build Variant Multiplicity</b>  | true   |    |              |
| <b>Post-Build Variant Value</b>         | true   |    |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|   | <b>Link time</b>   | -- |              |
|   | <b>Post-build time</b>   | -- |              |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X  | All Variants |
|   | <b>Link time</b>   | -- |              |
|   | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b>               | scope: ECU   |    |              |

#### No Included Containers

[SWS\_Lin\_CONSTR\_00270] [The module will operate as an independent instance in each of the partitions, means the called API will only target the partition it is called in. ]()

### 10.2.3 LinChannel

|                                 |  |  |  |
|---------------------------------|--|--|--|
| <b>SWS Item</b>                 | <b>ECUC_Lin_00069 :</b>  |  |  |
| <b>Container Name</b>           | LinChannel   |  |  |
| <b>Parent Container</b>         | LinGlobalConfig  |  |  |
| <b>Description</b>              | This container contains the configuration (parameters) of the LIN Controller(s). |  |  |
| <b>Configuration Parameters</b> |  |  |  |

|                         |  |  |  |
|-------------------------|--|--|--|
| <b>SWS Item</b>         | <b>ECUC_Lin_00180 :</b>                    |  |  |
| <b>Name</b>             | LinChannelBaudRate                         |  |  |
| <b>Parent Container</b> | LinChannel                                 |  |  |
| <b>Description</b>      | Specifies the baud rate of the LIN channel |  |  |
| <b>Multiplicity</b>     | 1  |  |  |
| <b>Type</b>             | EcucIntegerParamDef                        |  |  |
| <b>Range</b>            | 1000 .. 20000                              |  |  |

|                                  |                         |    |                     |
|----------------------------------|-------------------------|----|---------------------|
| <b>Default value</b>             | --                      |    |                     |
| <b>Post-Build Variant Value</b>  | true                    |    |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b> | X  | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>        | -- |                     |
|                                  | <b>Post-build time</b>  | X  | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>        | scope: local            |    |                     |

|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Lin_00181 :</b>   |    |              |
| <b>Name</b>                      | LinChannelId  |    |              |
| <b>Parent Container</b>          | LinChannel  |    |              |
| <b>Description</b>               | Identifies the LIN channel. Replaces LIN_CHANNEL_INDEX_NAME from the LIN SWS. |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucIntegerParamDef (Symbolic Name generated for this parameter)              |    |              |
| <b>Range</b>                     | 0 .. 65535  |    |              |
| <b>Default value</b>             | --  |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                                  | <b>Link time</b>  | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: local  |    |              |

|                                  |  |    |              |
|----------------------------------|--|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Lin_00182 :</b>  |    |              |
| <b>Name</b>                      | LinChannelWakeupSupport  |    |              |
| <b>Parent Container</b>          | LinChannel   |    |              |
| <b>Description</b>               | Specifies if the LIN hardware channel supports wake up functionality |    |              |
| <b>Multiplicity</b>              | 1  |    |              |
| <b>Type</b>                      | EcucBooleanParamDef  |    |              |
| <b>Default value</b>             | --   |    |              |
| <b>Post-Build Variant Value</b>  | false  |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|                                  | <b>Link time</b>   | -- |              |
|                                  | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b>        | scope: local   |    |              |

|                                  |  |    |              |
|----------------------------------|--|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Lin_00191 :</b>                      |    |              |
| <b>Name</b>                      | LinNodeType                                  |    |              |
| <b>Parent Container</b>          | LinChannel                                   |    |              |
| <b>Description</b>               | Specifies the LIN node type of this channel. |    |              |
| <b>Multiplicity</b>              | 1  |    |              |
| <b>Type</b>                      | EcucEnumerationParamDef                      |    |              |
| <b>Range</b>                     | MASTER                                       |    | Master node  |
|                                  | SLAVE  |    | Slave node   |
| <b>Post-Build Variant Value</b>  | false  |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                      | X  | All Variants |
|                                  | <b>Link time</b>                             | -- |              |
|                                  | <b>Post-build time</b>                       | -- |              |
| <b>Scope / Dependency</b>        | scope: local                                 |    |              |

|                         |   |  |  |
|-------------------------|---|--|--|
| <b>SWS Item</b>         | <b>ECUC_Lin_00193 :</b>   |  |  |
| <b>Name</b>             | LinChannelEcucPartitionRef  |  |  |
| <b>Parent Container</b> | LinChannel  |  |  |
| <b>Description</b>      | Maps one single Lin channel to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the Lin driver |  |  |

|   |                               |    |              |
|---|-------------------------------|----|--------------|
|   | is mapped to.                 |    |              |
| <b>Multiplicity</b>                     | 0..1                          |    |              |
| <b>Type</b>                             | Reference to [ EcuPartition ] |    |              |
| <b>Post-Build Variant Multiplicity</b>  | true                          |    |              |
| <b>Post-Build Variant Value</b>         | true                          |    |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>       | X  | All Variants |
|   | <b>Link time</b>              | -- |              |
|   | <b>Post-build time</b>        | -- |              |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>       | X  | All Variants |
|   | <b>Link time</b>              | -- |              |
|   | <b>Post-build time</b>        | -- |              |
| <b>Scope / Dependency</b>               | scope: ECU                    |    |              |

|   |  |    |              |
|---|--|----|--------------|
| <b>SWS Item</b>                         | <b>ECUC_Lin_00185 :</b>  |    |              |
| <b>Name</b>                             | LinChannelEcuMWakeupSource   |    |              |
| <b>Parent Container</b>                 | LinChannel   |    |              |
| <b>Description</b>                      | This parameter contains a reference to the Wakeup Source for this channel as defined in the ECU State Manager. This reference is only needed if LinChannelWakeupSupport is true. |    |              |
| <b>Multiplicity</b>                     | 0..1   |    |              |
| <b>Type</b>                             | Symbolic name reference to [ EcuMWakeupSource ]  |    |              |
| <b>Post-Build Variant Multiplicity</b>  | false  |    |              |
| <b>Post-Build Variant Value</b>         | false  |    |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|   | <b>Link time</b>   | -- |              |
|   | <b>Post-build time</b>   | -- |              |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X  | All Variants |
|   | <b>Link time</b>   | -- |              |
|   | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b>               | scope: ECU<br>dependency: LinChannelWakeupSupport  |    |              |

|                                  |  |    |                     |
|----------------------------------|--|----|---------------------|
| <b>SWS Item</b>                  | <b>ECUC_Lin_00094 :</b>  |    |                     |
| <b>Name</b>                      | LinClockRef  |    |                     |
| <b>Parent Container</b>          | LinChannel   |    |                     |
| <b>Description</b>               | Reference to the LIN clock source configuration, which is set in the MCU driver configuration. |    |                     |
| <b>Multiplicity</b>              | 1  |    |                     |
| <b>Type</b>                      | Reference to [ McuClockReferencePoint ]  |    |                     |
| <b>Post-Build Variant Value</b>  | true   |    |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X  | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>   | -- |                     |
|                                  | <b>Post-build time</b>   | X  | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: LIN clock source configuration in MCU Driver                       |    |                     |

|                               |
|-------------------------------|
| <b>No Included Containers</b> |
|-------------------------------|

The configuration parameter LinChannelWakeupSupport can be ignored during validation of wakeup signal.

[SWS\_Lin\_CONSTR\_00278] [The ECUC partitions referenced by LinChannelEcucPartitionRef shall be a subset of the ECUC partitions referenced by LinEcucPartitionRef. ]()

[SWS\_Lin\_CONSTR\_00291] [If LinEcucPartitionRef references one or more ECUC partitions, LinChannelEcucPartitionRef shall have a multiplicity of one and reference one of these ECUC partitions as well. ]()

[SWS\_Lin\_CONSTR\_00279] [LinChannel and LinTrcvChannel of one communication channel shall all reference the same ECUC partition. ]()

## 10.2.4 LinGlobalConfig

|                                 |   |
|---------------------------------|---|
| <b>SWS Item</b>                 | <b>ECUC_Lin_00184 :</b>   |
| <b>Container Name</b>           | LinGlobalConfig   |
| <b>Parent Container</b>         | Lin   |
| <b>Description</b>              | This container contains the global configuration parameter of the Lin driver. |
| <b>Configuration Parameters</b> |   |

|                            |                     |  |
|----------------------------|---------------------|--|
| <b>Included Containers</b> |                     |  |
| <b>Container Name</b>      | <b>Multiplicity</b> | <b>Scope / Dependency</b>  |
| LinChannel                 | 1..*                | This container contains the configuration (parameters) of the LIN Controller(s). |

## 10.2.5 LinDemEventParameterRefs

|                                 |  |
|---------------------------------|--|
| <b>SWS Item</b>                 | <b>ECUC_Lin_00188 :</b>  |
| <b>Container Name</b>           | LinDemEventParameterRefs   |
| <b>Parent Container</b>         | Lin  |
| <b>Description</b>              | Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references. |
| <b>Configuration Parameters</b> |  |

|   |   |    |              |
|---|---|----|--------------|
| <b>SWS Item</b>                         | <b>ECUC_Lin_00189 :</b>   |    |              |
| <b>Name</b>                             | LIN_E_TIMEOUT   |    |              |
| <b>Parent Container</b>                 | LinDemEventParameterRefs  |    |              |
| <b>Description</b>                      | Reference to the DemEventParameter which shall be issued when the error "Timeout caused by hardware error" has occurred. If the reference is not configured the error shall be reported as DET error. |    |              |
| <b>Multiplicity</b>                     | 0..1  |    |              |
| <b>Type</b>                             | Symbolic name reference to [ DemEventParameter ]  |    |              |
| <b>Post-Build Variant Multiplicity</b>  | false   |    |              |
| <b>Post-Build Variant Value</b>         | false   |    |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>   | X  | All Variants |
|   | <b>Link time</b>  | -- |              |



|                                  |                         |    |              |
|----------------------------------|-------------------------|----|--------------|
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b> | X  | All Variants |
|                                  | <b>Link time</b>        | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: local            |    |              |

**No Included Containers**

### 10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in *SWS\_BSWGeneral*.

## 11 Not applicable requirements

**[SWS\_Lin\_00999]** 「 These requirements are not applicable to this specification. 」

(SRS\_BSW\_00307, SRS\_BSW\_00312, SRS\_BSW\_00325, SRS\_BSW\_00328, SRS\_BSW\_00330, SRS\_BSW\_00331, SRS\_BSW\_00336, SRS\_BSW\_00339, SRS\_BSW\_00342, SRS\_BSW\_00343, SRS\_BSW\_00353, SRS\_BSW\_00357, SRS\_BSW\_00359, SRS\_BSW\_00360, SRS\_BSW\_00361, SRS\_BSW\_00373, SRS\_BSW\_00378, SRS\_BSW\_00383, SRS\_BSW\_00395, SRS\_BSW\_00397, SRS\_BSW\_00398, SRS\_BSW\_00399, SRS\_BSW\_00400, SRS\_BSW\_00413, SRS\_BSW\_00415, SRS\_BSW\_00416, SRS\_BSW\_00417, SRS\_BSW\_00422, SRS\_BSW\_00423, SRS\_BSW\_00424, SRS\_BSW\_00425, SRS\_BSW\_00426, SRS\_BSW\_00427, SRS\_BSW\_00428, SRS\_BSW\_00429, SRS\_BSW\_00432, SRS\_BSW\_00433, SRS\_BSW\_00005, SRS\_BSW\_00007, SRS\_BSW\_00162, SRS\_BSW\_00168, SRS\_Lin\_01551, SRS\_Lin\_01568, SRS\_Lin\_01569, SRS\_Lin\_01570, SRS\_Lin\_01564, SRS\_Lin\_01546, SRS\_Lin\_01561, SRS\_Lin\_01549, SRS\_Lin\_01571, SRS\_Lin\_01514, SRS\_Lin\_01515, SRS\_Lin\_01502, SRS\_Lin\_01558, SRS\_Lin\_01523, SRS\_Lin\_01540, SRS\_Lin\_01545, SRS\_Lin\_01534, SRS\_Lin\_01574, SRS\_Lin\_01539, SRS\_Lin\_01544, SRS\_Lin\_01590, SRS\_Lin\_01594, SRS\_Lin\_01595, SRS\_Lin\_01596, SRS\_Lin\_01597)