

Document Title	Specification of LIN Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	072
Document Classification	Standard

Document Version	1.5.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History			
Date	Version	Changed by	Change Description
28.10.2011	1.5.0	AUTOSAR Administration	<ul style="list-style-type: none"> • Changed error reporting • Improved wake-up handling • Corrected call of Lin_Init
12.10.2010	1.4.0	AUTOSAR Administration	<ul style="list-style-type: none"> • Introduce Lin_GeneralTypes.h • Add missing DET error code (NULL pointer error) • Remove instance ID from Lin_GetVersionInfo API • Correct naming of “WakeUp” to “Wakeup” • Further maintenance for R4.0.2: see chapter 12
03.12.2009	1.3.0	AUTOSAR Administration	<ul style="list-style-type: none"> • Support of advanced LIN controllers (combination of Lin_SendHeader and Lin_SendResponse to Lin_SendFrame) • Integrating LIN channel initialization in LIN module initialization • Further maintenance for R4.0: see chapter 11 • Legal disclaimer revised
23.06.2008	1.2.1	AUTOSAR Administration	Legal disclaimer revised
11.12.2007	1.2.0	AUTOSAR Administration	<ul style="list-style-type: none"> • Editorial Changes • Tables generated in Chapter 8 and 10 • Document meta information extended • Small layout adaptations made

Document Change History

Date	Version	Changed by	Change Description
30.01.2007	1.1.0	AUTOSAR Administration	<ul style="list-style-type: none">• Lin Transceiver Wake Up validation function added• Incorporate Feedback from Validator2• Updated Chapter 10.2 according to the Specification of ECU Configuration Parameters • Legal disclaimer revised• Release Notes added• “Advice for users” revised• “Revision Information” added
11.05.2006	1.0.0	AUTOSAR Administration	Initial release

Disclaimer

This specification 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 specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification 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 specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

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

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

Table of Contents

1	Introduction and functional overview	7
1.1	Scope	7
1.2	Architectural overview	7
2	Acronyms, abbreviations and glossary	9
2.1	Acronyms and abbreviations	9
2.2	Glossary	9
2.3	LIN hardware unit classification	10
3	Related documentation	11
3.1	Input documents	11
3.2	Related standards and norms	11
4	Constraints and assumptions	12
4.1	Limitations	12
4.2	Applicability to car domains	12
5	Dependencies to other modules	13
5.1	File structure	13
5.1.1	Code file structure	13
5.1.2	Header file structure	14
6	Requirements traceability	16
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	28
7.4.4	Data byte mapping	29
7.5	Sleep and wake-up functionality	29
7.5.1	Background & Rationale	29
7.5.2	Requirements	30
7.6	Error classification	30
7.7	Error detection	32
7.8	Error notification	32
7.9	Debugging	33
8	API specification	34
8.1	Imported types	34
8.2	Type definitions	34

8.2.1	Lin_ConfigType	34
8.2.2	Lin_FramePidType	35
8.2.3	Lin_FrameCsModelType	35
8.2.4	Lin_FrameResponseType	35
8.2.5	Lin_FrameDType	35
8.2.6	Lin_PduType	36
8.2.7	Lin_StatusType	36
8.3	Function definitions	37
8.3.1	Services affecting the complete LIN hardware unit	37
8.3.1.1	Lin_Init	37
8.3.1.2	Lin_CheckWakeup	38
8.3.1.3	Lin_GetVersionInfo	39
8.3.2	Services affecting a single LIN channel	40
8.3.2.1	Lin_SendFrame	40
8.3.2.2	Lin_GoToSleep	42
8.3.2.3	Lin_GoToSleepInternal	43
8.3.2.4	Lin_Wakeup	44
8.3.2.5	Lin_GetStatus	45
8.4	Call-back notifications	47
8.5	Scheduled functions	47
8.6	Expected Interfaces	47
8.6.1	Mandatory Interfaces	47
8.6.2	Optional Interfaces	47
8.6.3	Configurable interfaces	48
9	Sequence diagrams	49
9.1	Receiving a LIN Frame	49
10	Configuration specification	50
10.1	How to read this chapter	50
10.1.1	Configuration and configuration parameters	50
10.1.2	Variants	51
10.1.3	Containers	51
10.2	Containers and configuration parameters	51
10.2.1	Variants	53
10.2.2	Lin	54
10.2.3	LinGeneral	54
10.2.4	LinChannel	55
10.2.5	LinGlobalConfig	56
10.2.6	LinDemEventParameterRefs	57
10.3	Published Information	57
11	Changes to Release 3	58
11.1	Deleted SWS Items	58
11.2	Replaced SWS Items	58
11.3	Changed SWS Items	58
11.4	Added SWS Items	59
12	Changes to Release 4.0 Rev1	61
12.1	Deleted SWS Items	61
12.2	Replaced SWS Items	61

12.3	Changed SWS Items.....	61
12.4	Added SWS Items.....	61
13	Changes to Release 4.0 Rev2.....	62
13.1	Deleted SWS Items.....	62
13.2	Replaced SWS Items.....	62
13.3	Changed SWS Items.....	62
13.4	Added SWS Items.....	62
14	Not applicable requirements.....	63

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 LIN 2.1 specification [16]. It is assumed that the reader is familiar with this specification. This document will not describe LIN 2.1 functionality again, but it will try to follow the same order as the LIN 2.1 specification.

The LIN driver applies to LIN 2.1 master nodes only. Operating as a slave node is out of scope. The LIN master in AUTOSAR deviates from the LIN 2.1 specification 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 slaves together with the AUTOSAR LIN master (i.e. the LIN driver).

[LIN063] 「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.」(BSW01547)

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.

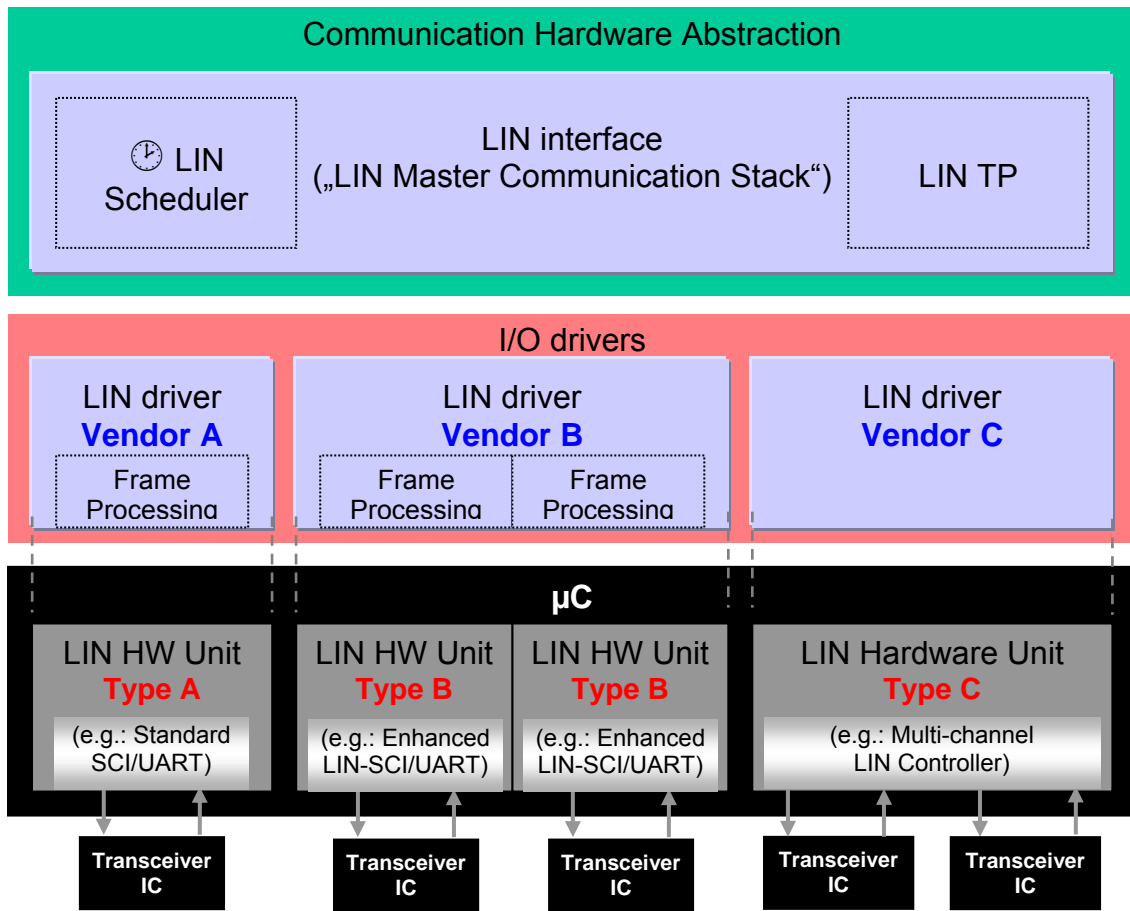


Figure 10-1: Overview LIN Software Architecture Layering

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.

Acronym:	Description:
AUTOSAR	Automotive Open System Architecture
COM	Communication
ECU	Electronic Control Unit
EcuM	ECU Manager
DEM	Diagnostic Event Manager
DET	Development 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

Abbreviation	Description:
Id	Identifier

2.2 Glossary

Besides AUTOSAR terminology this document also uses terms defined in the LIN 2.1 specification [16], e.g. LIN frame, header and message.

Glossary:	Description:
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:

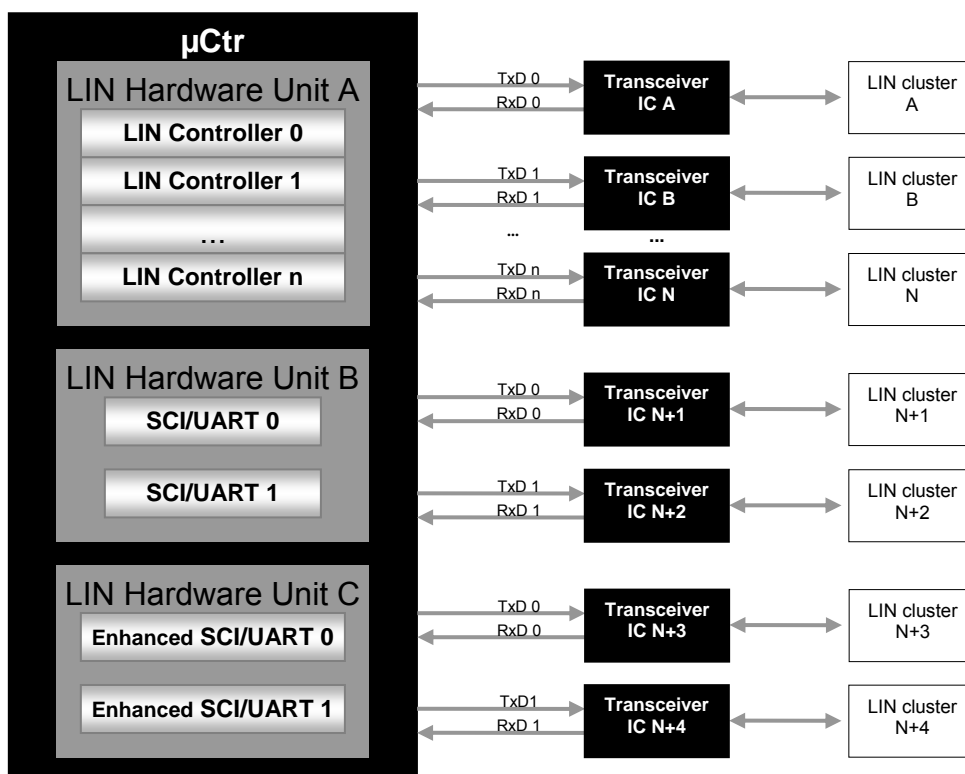


Figure 2-1: LIN hardware unit classification

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 Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.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 C Implementation Rules
AUTOSAR_TR_CImplementationRules.pdf
- [13] Specification of ECU State Manager
AUTOSAR_SWS_ECUCStateManager.pdf
- [14] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf
- [15] Specification of LIN Transceiver Driver,
AUTOSAR_SWS_LINTransceiverDriver.pdf

3.2 Related standards and norms

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

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

[LIN045] 「One LIN driver provides access to one LIN hardware unit type (simple UART or dedicated LIN hardware) that may consist of several LIN channels. 」
(BSW00347)

[LIN201] 「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.」()

[LIN177] 「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 BSW00347 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 (Development Error Tracer) [5]

In development mode, the Lin module reports development error through the Det_ReportError function of module [DET](#). (see [LIN052](#))

Module DEM (Diagnostic Event Manager) [11]

The Lin module reports production errors to the Diagnostic Event Manager. (see [LIN058](#))

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

[LIN064] 「The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files named:

- Lin_Lcfg.c – for link time configurable parameters and
- Lin_PBcfg.c – for post build time configurable parameters.

These files shall contain all link time and post-build time configurable parameters.」(BSW00380, BSW00419)

5.1.2 Header file structure

[LIN075] 「The include file structure shall be as follows:

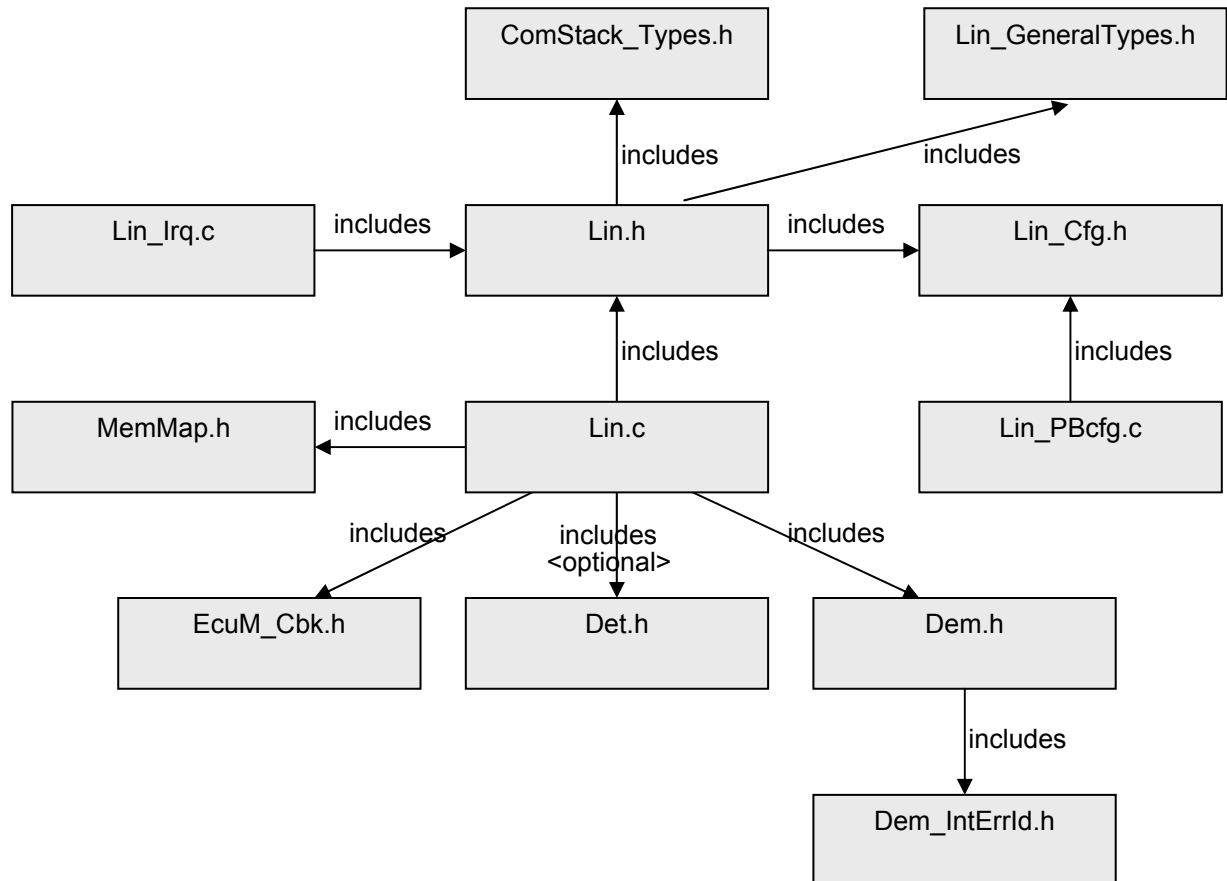


Figure 5-1: Header File structure for the LIN driver」()

[LIN202] 「Lin.c shall include Lin.h.」()

[LIN203] 「Lin.c shall include MemMap.h.」()

[LIN204] 「Lin.h shall include Lin_Cfg.h.」()

[LIN205] 「Lin.h shall include ComStack_Types.h.」()

[LIN241] 「Lin.h shall include Lin_GeneralTypes.h.」()

[LIN023] 「The source file Lin_Irq.c contains the implementation of interrupt frames.

The implementation of the interrupt service routine shall be in Lin.c.」(BSW00314)

[LIN042] 「The header file `EcuM_Cbk.h` contains the declarations of the callback functions imported by the modules calling the callbacks. 」(BSW00370)

[LIN206] 「The LIN driver itself does not provide callback functions (no `Lin_Cbk.h`)」
()

[LIN054] 「The file `Lin.h` only contains external declarations of constants, global data, type definitions and services that are specified in the LIN driver SWS. 」
(BSW00302)

[LIN207] 「Constants, global data types and functions that are only used by LIN driver internally, are declared in `Lin.c`.」()

[LIN065] 「The module shall include the `Dem.h` file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. 」(BSW00409)

[LIN208] 「This specification defines the name of the Event Id symbols which are provided by XML to the [DEM](#) configuration tool. The [DEM](#) configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in `Dem_IntErrId.h`.」()

[LIN242] 「`Lin_GeneralTypes.h` shall contain all types and constants that are shared among the AUTOSAR LIN modules `Lin`, `LinIf` and `LinTrcv`.」()

[LIN243] 「 The integrator of the LIN modules shall provide the file `Lin_GeneralTypes.h`.」()

6 Requirements traceability

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

Requirement	Satisfied by
[BSW003] Version identification	Software Documentation Requirements are not covered in the LIN driver SWS
[BSW00300] Module naming convention	Fulfilled by the function name definitions in Chapter 8.3
[BSW00301] Limit imported information	See Chapter 5.1.2
[BSW00302] Limit exported information	LIN054
[BSW00304] AUTOSAR integer data types	LIN047 , Chapter 8.2 and Chapter 10.3
[BSW00305] Self-defined data types naming convention	Fulfilled by the function name definitions in Chapter 8.2
[BSW00306] Avoid direct use of compiler and platform specific keywords	LIN055
[BSW00307] Global variables naming convention	Not applicable (requirement on implementation)
[BSW00308] Definition of global data	LIN055
[BSW00309] Global data with read-only constraint	LIN055
[BSW00310] API naming convention	See Chapter 5.1.2
[BSW00312] Shared code shall be reentrant	Not applicable
[BSW00314] Separation of interrupt frames and service routines	LIN023
[BSW00318] Format of module version numbers	See chapter 10.3
[BSW00321] Enumeration of module version numbers	See chapter 10.3
[BSW00323] API parameter checking	LIN048 , LIN049
[BSW00325] Runtime of interrupt service routines	Not applicable (requirement on implementation)
[BSW00326] Transition from ISRs to OS tasks	Not applicable (requirement on implementation)
[BSW00327] Error values naming convention	LIN048
[BSW00328] Avoid duplication of code	Not applicable (requirement on implementation, fulfilled e.g. by defining a LIN driver that controls multiple channels)
[BSW00329] Avoidance of generic interfaces	Not applicable (no generic interfaces specified within this SWS)
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable (requirement on implementation)
[BSW00331] Separation of error and status values	Not applicable
[BSW00333] Documentation of callback function context	Software Documentation Requirements are not covered in the LIN driver SWS
[BSW00334] Provision of XML file	Software Documentation Requirements are not covered in the LIN driver SWS
[BSW00335] Status values naming convention	Fulfilled by the state diagram description in chapter 7.3.3
[BSW00336] Shutdown interface	Not applicable
[BSW00337] Classification of errors	LIN048
[BSW00338] Detection and Reporting of development errors	LIN049 , LIN052
[BSW00339] Reporting of production relevant error status	Not applicable
[BSW00341] Microcontroller compatibility documentation	Software Documentation Requirements are not covered in the

	LIN driver SWS
[BSW00342] Usage of source code and object code	Not applicable (requirement on implementation)
[BSW00343] Specification and configuration of time	Not applicable
[BSW00344] Reference to link-time configuration	LIN013
[BSW00345] Pre-compile-time configuration	See Chapter 10
[BSW00346] Basic set of module files	See Chapter 5.1.2
[BSW00347] Naming separation of different instances of BSW drivers	LIN045
[BSW00348] Standard type header	See Chapter 5.1.2
[BSW00350] Development error detection keyword	LIN066 Conf
[BSW00353] Platform specific type header	Not applicable (automatically included with standard types)
[BSW00355] Do not redefine AUTOSAR integer data types	no redefined integer types in Chapter 8.2 and Chapter 10.3
[BSW00357] Standard API return type	Not applicable (this type is not used within this SWS)
[BSW00358] Return type of init() functions	fulfilled by 8.3.1.1
[BSW00359] Return type of callback functions	Not applicable (no callback function specified)
[BSW00360] Parameters of callback functions	Not applicable (no callback function specified)
[BSW00361] Compiler specific language extension header	Not applicable (automatically included with standard types)
[BSW00369] Do not return development error codes via API	See chapter 8
[BSW00370] Separation of callback interface from API	LIN042
[BSW00371] Do not pass function pointers via API	Fulfilled by the function definitions in Chapter 8.3
[BSW00373] Main processing function naming convention	Not applicable (no main processing function specified)
[BSW00374] Module vendor identification	See chapter 10.3
[BSW00375] Notification of wake-up reason	LIN098
[BSW00376] Return type and parameters of main processing functions	Not applicable (no main processing function specified)
[BSW00377] Module specific API return types	See 8.2.7
[BSW00378] AUTOSAR boolean type	Not applicable (not used)
[BSW00379] Module identification	See chapter 10.3
[BSW00380] Separate C-File for configuration parameters	LIN064
[BSW00381] Separate configuration header file for pre-compile time parameters	See Chapter 5.1.2
[BSW00383] List dependencies of configuration files	Not applicable (implementation specific documentation)
[BSW00384] List dependencies to other modules	See Chapter 5
[BSW00385] List possible error notifications	LIN048
[BSW00386] Configuration for detecting an error	See Chapter 7.6
[BSW00387] Specify the configuration class of callback function	Chapter 8.6.3
[BSW00388] Introduce containers	See Chapter 10.2
[BSW00389] Containers shall have names	See Chapter 10.2
[BSW00390] Parameter content shall be unique within the module	See Chapter 8
[BSW00391] Parameter shall have unique names	fulfilled by parameter definitions in Chapter 10.2
[BSW00392] Parameters shall have a type	fulfilled by parameter definitions in

	Chapter 10.2
[BSW00393] Parameters shall have a range	fulfilled by parameter definitions in Chapter 10.2
[BSW00394] Specify the scope of the parameters	fulfilled by parameter definitions in Chapter 10.2
[BSW00395] List the required parameters (per parameter)	Not applicable (parameters are defined in a way that their values are independent from other settings. The dependency is in the code generation (implementation) not in the configuration description -> hardware abstraction)
[BSW00396] Configuration classes	fulfilled by parameter definitions in Chapter 10.2
[BSW00397] Pre-compile-time parameters	Not applicable (this is not a requirement, but a definition of a technical term)
[BSW00398] Link-time parameters	Not applicable (this is not a requirement, but a definition of a technical term)
[BSW00399] Loadable Post-build time parameters	Not applicable (this is not a requirement, but a definition of a technical term)
[BSW004] Version check	LIN062
[BSW00400] Selectable Post-build time parameters	Not applicable (this is not a requirement, but a definition of a technical term)
[BSW00401] Documentation of multiple instances of configuration parameters	Software Documentation Requirements are not covered in the LIN driver SWS
[BSW00402] Published information	See chapter 10.3
[BSW00404] Reference to post build time configuration	LIN013
[BSW00405] Reference to multiple configuration sets	LIN011 , LIN013
[BSW00406] Check module initialization	LIN006
[BSW00407] Function to read out published parameters	LIN001
[BSW00408] Configuration parameter naming convention	fulfilled by Chapter 10.2
[BSW00409] Header files for production code error IDs	LIN065 , LIN046
[BSW00410] Compiler switches shall have specified values	fulfilled by Chapter 10.2
[BSW00411] Get version info keyword	LIN066 Conf and 8.3.1.3
[BSW00412] Separate H-File for configuration parameters	See Chapter 5.1.2
[BSW00413] Accessing instances of BSW modules	Not applicable (this requirement has to fulfilled by the LIN Interface)
[BSW00414] Parameter of init function	fulfilled by 8.3.1.1
[BSW00415] User dependent include files	Not applicable (only one user for this module)
[BSW00416] Sequence of Initialization	Not applicable (this is a general software integration requirement)
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable (LIN driver is a Basic Software Module)
[BSW00419] Separate C-Files for pre-compile time configuration parameters	LIN064
[BSW00420] Production relevant error event rate detection	Not applicable (requirement on the DEM)
[BSW00421] Reporting of production relevant error events	LIN058
[BSW00422] Debouncing of production relevant error status	Not applicable (requirement on the DEM)

[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (this module does not provide an AUTOSAR interface)
[BSW00424] BSW main processing function task allocation	Not applicable (requirement on system design, not on a single module)
[BSW00425] Trigger conditions for schedulable objects	Not applicable (trigger conditions are system configuration specific)
[BSW00426] Exclusive areas in BSW modules	Not applicable
[BSW00427] ISR description for BSW modules	Not applicable (no ISR defined for this module, usage of interrupts are implementation specific)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (LIN driver does not contain any main processing functions)
[BSW00429] Restricted BSW OS functionality access	Not applicable (implementation requirement, not for the specification)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (applies only to BSW scheduler module)
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (no main processing function specified)
[BSW00433] Calling of main processing functions	Not applicable (requirement on system design, not on a single module)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (applies only to BSW scheduler module)
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (fulfilled by the AUTOSAR architectural concept)
[BSW006] Platform independency	LIN003
[BSW007] HIS MISRA C	Not applicable (requirement on implementation)
[BSW009] Module User Documentation	Software Documentation Requirements are not covered in the LIN driver SWS
[BSW010] Memory resource documentation	Software Documentation Requirements are not covered in the LIN driver SWS
[BSW101] Initialization interface	LIN006
[BSW158] Separation of configuration from implementation	See Chapter 5.1.2
[BSW159] Tool-based configuration	LIN029
[BSW160] Human-readable configuration data	LIN031
[BSW161] Microcontroller abstraction	LIN003
[BSW162] ECU layout abstraction	Not applicable (fulfilled by the AUTOSAR architectural concept)
[BSW164] Implementation of interrupt service routines	LIN155
[BSW167] Static configuration checking	LIN039
[BSW168] Diagnostic Interface of SW components	Not applicable (LIN driver doesn't offer a diagnostic interface)
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	See Chapter10

[BSW171] Configurability of optional functionality	LIN066 Conf , LIN067 Conf
[BSW172] Compatibility and documentation of scheduling strategy	Software Documentation Requirements are not covered in the LIN driver SWS

Document: AUTOSAR requirements on Basic Software, Cluster: SPAL general [6]

Requirement	Satisfied by
[BSW12263] Object code compatible configuration concept	LIN013
[BSW12056] Configuration of notification mechanisms	Not applicable
[BSW12267] Configuration of wake-up sources	Not applicable
[BSW12057] driver module initialization	LIN006
[BSW12125] Initialization of hardware resources	LIN006 , LIN190
[BSW12163] driver module deinitialization	not applicable (decision in Joint Meeting: no de-initialization for drivers that don't need to store non volatile information)
[BSW12461] Responsibility for register initialization	LIN008
[BSW12462] Provide settings for register initialization	See Chapter 10.3
[BSW12463] Combine and forward settings for register initialization	Not applicable (applies only for configurator)
[BSW12068] MCAL initialization sequence	Not applicable
[BSW12069] Wake-up notification of ECU State Manager	LIN098
[BSW157] Notification mechanisms of drivers and handlers	LIN022 , LIN052 , LIN053
[BSW12169] Control of operation mode	LIN032
[BSW12063] Raw value mode	LIN016 , LIN025
[BSW12075] Use of application buffers	Not applicable (LIN driver does not feature random streaming capability)
[BSW12129] Resetting of interrupt flags	LIN157
[BSW12064] Change of operation mode during running operation	LIN032
[BSW12448] Behavior after development error detection	LIN052 , LIN237
[BSW12067] Setting of wake-up conditions	LIN032
[BSW12077] Non-blocking implementation	LIN027 , LIN028 .
[BSW12078] Runtime and memory efficiency	Not applicable because this is a non-functional requirement
[BSW12092] Access to drivers	Not applicable because this is a non-functional requirement
[BSW12265] Configuration data shall be kept constant	LIN013 (stored in ROM, i.e. implicitly constant)
[BSW12264] Specification of configuration items	See Chapter10

Document: AUTOSAR requirements on Basic Software, Cluster: LIN [7]

Requirement	Satisfied by
[BSW01576] Usage of LIN specification 2.1	LIN005
[BSW01504] Usage of AUTOSAR architecture only in LIN master nodes	LIN005
[BSW01522] Consistent data transfer	LIN025 , LIN053 , LIN060
[BSW01560] Support for wake-up during transition to sleep-mode	LIN033 , LIN034 , LIN035
[BSW01551] Multiple LIN channel support for interface	Not applicable for the LIN driver
[BSW01568] Hardware independence	Not applicable for the LIN driver
[BSW01569] LIN Interface initialization	Not applicable for the LIN driver

[BSW01570] Selection of static configuration sets	Not applicable for the LIN driver
[BSW01564] Schedule Table Manager	Not applicable for the LIN driver
[BSW01546] Schedule Table Handler	Not applicable for the LIN driver
[BSW01561] Main function	Not applicable for the LIN driver
[BSW01549] Timer service for Scheduling	Not applicable for the LIN driver
[BSW01571] Transmission request service	Not applicable for the LIN driver
[BSW01514] Wake-up notification support	Not applicable for the LIN driver
[BSW01515] API to wake-up by upper layer to LIN Interface	Not applicable for the LIN driver
[BSW01502] RX indication and TX confirmation call-backs	Not applicable for the LIN driver
[BSW01558] Check successful communication	Not applicable for the LIN driver
[BSW01527] Notification for missing or erroneous receive LIN-PDU	Not applicable for the LIN driver
[BSW01523] API to send the LIN to sleep-mode	Not applicable for the LIN driver
[BSW01553] Basic Software SPAL General Requirements	See table above
[BSW01552] Hardware abstraction LIN	See chapter 10.3
[BSW01503] Frame based API for send and received data	LIN024 , LIN025
[BSW01555] LIN Interface shall poll the LIN driver for transmit/receive notifications	LIN024
[BSW01547] Support of standard UART and LIN optimized HW	LIN063
[BSW01572] LIN driver initialization	LIN011
[BSW01573] Selection of static configuration sets	LIN011
[BSW01563] Wake-up Notification	LIN098
[BSW01556] Multiple LIN channel support for driver	LIN008 , LIN190
[BSW01566] Transition to sleep-mode	LIN033 , LIN034 , LIN035 , LIN073
[BSW01524] Support of reduced power operation mode	LIN032
[BSW01526] Error notification	LIN052 , LIN053
[BSW01540] LIN Transport Layer Initialization	Not applicable for the LIN driver
[BSW01545] LIN Transport Layer Availability	Not applicable for the LIN driver
[BSW01534] Concurrent connection configuration	Not applicable for the LIN driver
[BSW01574] Multiple Transport Layer instances	Not applicable for the LIN driver
[BSW01539] Transport connection properties	Not applicable for the LIN driver
[BSW01544] Error handling	Not applicable for the LIN driver
[BSW01590] Usage of schedule tables for node configuration	Not applicable for the LIN driver
[BSW01577] Compatibility to LIN 2.1 protocol specification	LIN005
[BSW01578] Compatibility to LIN protocol specification	LIN017
[BSW01579] Compatibility to TP of LIN specification	LIN TP requirement
[BSW01591] Diagnostic transmission handler	LIN Interface requirement
[BSW01580] Configuration Data for LIN Transceiver Driver.	LIN Transceiver Driver requirement
[BSW01581] Support for more than one LIN transceiver	LIN Transceiver Driver requirement
[BSW01583] API to initialize the LIN Transceiver Driver	LIN Transceiver Driver requirement
[BSW01582] LIN Transceiver Driver API shall be synchronous	LIN Transceiver Driver requirement
[BSW01584] API to request operation mode "standby"	LIN Transceiver Driver requirement
[BSW01585] API to request operation mode "sleep"	LIN Transceiver Driver requirement
[BSW01586] API to request operation mode "normal"	LIN Transceiver Driver requirement
[BSW01587] API to read out current operation mode	LIN Transceiver Driver requirement
[BSW01588] API to read out wakeup reason	LIN Transceiver Driver requirement
[BSW01589] API to enable/disable/clear wakeup Event	LIN Transceiver Driver requirement

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 LIN 2.1 specification. 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.

[LIN005] 「The Lin module shall conform to the LIN 2.1 Protocol Specification as specified in [16]. This applies to LIN 2.1 Master nodes only. 」(BSW01576, BSW01504, BSW01577)

Operating as a slave node is out of scope for this AUTOSAR LIN driver specification.

[LIN055] 「The Lin module shall fulfill all design and implementation guidelines as described in [12].」(BSW00306, BSW00308, BSW00309)

[LIN155] 「The Lin module shall implement the ISRs for all LIN hardware unit interrupts that are needed. 」(BSW164)

[LIN156] 「The Lin module shall ensure that all unused interrupts are disabled. 」()

[LIN157] 「The Lin module shall reset the interrupt flag at the end of the ISR (if not done automatically by hardware). 」(BSW12129)

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

[LIN062] 「Lin module shall perform Inter Module Checks to avoid integration of incompatible files.

The imported included files shall be checked by preprocessing directives. 」(BSW004)

The following version numbers shall be verified:

- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_RELEASE_MINOR_VERSION

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

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

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.

[LIN225] 「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.

[LIN011] 「The Lin module's configuration shall include a data communication rate set as defined by static configuration data. 」(BSW00405, BSW01572, BSW01573)

[LIN013] 「The Lin module’s configuration data, intended for hardware registers, shall be stored as hardware specific data structures in ROM (see [Lin_ConfigType](#)).」
(BSW00345, BSW00404, BSW00405, BSW12263, BSW12265)

[LIN014] 「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](#)).」()

[LIN015] 「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.

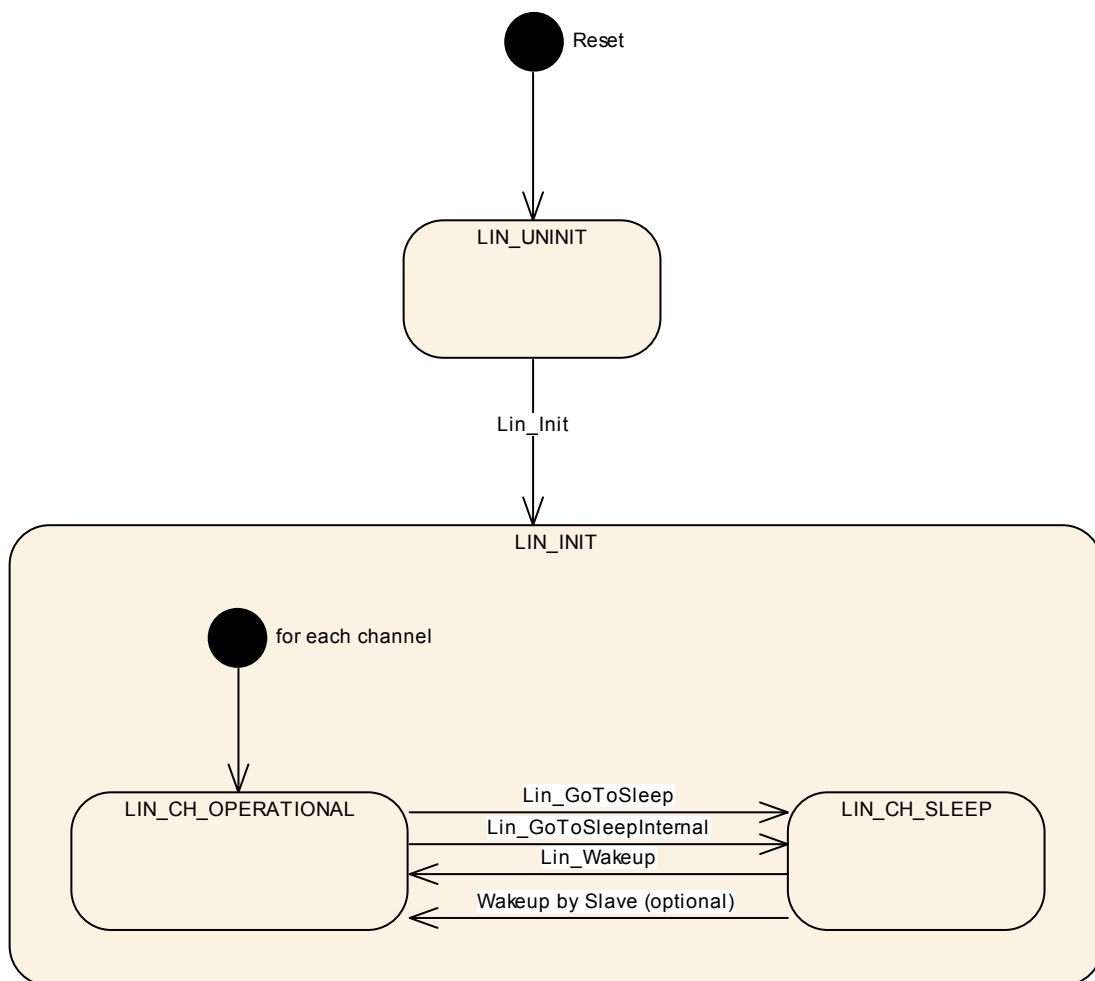


Figure 7-1: LIN driver states

Module State	Meaning / Activities in the state
LIN_UNINIT	The state LIN_UNINIT means that the Lin module has not been initialized yet and cannot be used.
LIN_INIT	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
LIN_CH_OPERATIONAL	The individual channel has been initialized (using at least one statically configured data set) and is able to participate in the LIN cluster.
LIN_CH_SLEEP	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.

[LIN145] **Reset -> LIN_UNINIT:** After reset, the Lin module shall set its state to LIN_UNINIT. `⌈()`

[LIN146] **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.

[LIN171] **On entering the state LIN_INIT,** the Lin module shall set each channel into state LIN_CH_OPERATIONAL. `⌈()`

[LIN172] **LIN_CH_OPERATIONAL -> LIN_CH_SLEEP:** If a go to sleep is requested by the LIN interface (Lin_GoToSleep), 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 state. `⌈()`

[LIN173] **LIN_CH_SLEEP -> LIN_CH_OPERATIONAL through Wake-Up by Slave:** if a LIN channel is in the state LIN_CH_SLEEP and upon detection of a valid wake-up pulse onto the bus, the Lin module shall put the LIN channel into the state LIN_CH_OPERATIONAL. The LIN 2.1 specification describes this 'wake-up' as a dominant state on the bus lasting between 250µs and 5ms. The activity during LIN_CH_SLEEP is to detect a dominant pulse, which shall be handled as valid wake-up request after 150 µs at the last. If such a wake-up was received from the bus, the master node has to begin communication to determine why the wake-up occurred. The form and content of this communication is outside the scope of the LIN driver specification. `⌈()`

A wake-up may also be directly requested from a higher layer in the AUTOSAR architecture (the LIN Interface layer will directly communicate this to the driver).

[LIN174] 「**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. 」()

[LIN209] 「**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 μ s to 5 ms.」()

7.4 Frame processing

7.4.1 Background & Rationale

From the point of view of the LIN driver module, transmissions are composed of two actions; the transmission of the LIN header, and the transmission of the response. Only the LIN master node transmits the LIN header, but either the master or one of the slaves may transmit the response [16].

The driver must also be able to access data concerning the checksum model and data length for each LIN PID. LIN 2.1 has a different checksum model compared to LIN1.3, but the LIN 2.1 master must be able to communicate with both LIN1.3 and LIN 2.1 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. This means that 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_MASTER_RESPONSE`) will be given at the same time as the request to send the header. The transmission of the response itself has to be triggered subsequently by another function call.

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

[LIN016] 「The LIN driver shall interpret the supplied identifier as PID. The identifier is then transmitted *as-supplied* within the LIN header (see `Lin_SendFrame`).」
(BSW12063)

[LIN017] 「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`).」(BSW01578)

[LIN018] 「The LIN driver shall be able to send a LIN header and response.」()

[LIN019] 「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.」()

[LIN021] 「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.」()

[LIN022] 「The function `Lin_GetStatus` shall return the status of the current frame transmission request for the channel.」(BSW157)

[LIN024] 「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](#)).」(BSW01555, BSW01503)

[LIN025] 「The LIN driver shall send response data as provided by the LIN interface module (see `Lin_SendFrame`).」(BSW12063, BSW01522, BSW01503)

[LIN026] 「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.」()

[LIN027] 「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).」(BSW12077)

[LIN028] 「The LIN driver shall receive data without blocking.」(BSW12077)

7.4.3 Data Consistency

Transmit Data Consistency:

[LIN053] 「The LIN driver shall directly copy the data from the upper layer buffers.」(BSW157, BSW01522, BSW01526)

[LIN210] 「The upper layer of the LIN Driver has to keep the buffer data consistent until return of function call.」()

Receive Data Consistency:

[LIN060] 「 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.」(BSW01522)

[LIN211] 「 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.」()

As long as it is guaranteed that neither the ISRs nor `Lin_GetStatus` 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.

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.

7.4.4 Data byte mapping

[LIN096] 「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

7.5.1 Background & Rationale

The 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 [LIN173](#), [LIN209](#)). 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 LINIF296, LINIF488).

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

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

7.5.2 Requirements

[LIN032] 「When the LIN channel is requested to enter sleep mode it shall perform the transition to low-power mode of the LIN hardware unit (if available) (see [Lin_GoToSleep/Lin_GoToSleepInternal](#)).」(BSW12169, BSW12064, BSW12067, BSW01524)

[LIN033] 「Each LIN channel shall be able to accept a sleep request independently of the other channel states (see [Lin_GoToSleep/Lin_GoToSleepInternal](#)).」(BSW01560, BSW01566)

[LIN035] 「The LIN channel shall activate the wake-up detection as soon as possible after completion of the go-to-sleep-command when the LIN bus becomes idle.」
(BSW01560, BSW01566)

[LIN037] 「When a LIN channel is in LIN_CH_SLEEP state, the LIN hardware unit shall monitor the bus for a wake-up request on that channel.」()

[LIN040] 「If a wake-up request was received, the LIN driver shall change state to LIN_CH_OPERATIONAL for the channel that received the wake-up pulse.」()

[LIN043] 「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](#))」()

[LIN212] 「Lin_Wakeup: If the LIN driver receives a wake-up request from the LIN interface, the wake-up detection of bus wake-up events has to be disabled for the requested channel. (see [Lin_Wakeup](#))」()

The function [Lin_GetStatus](#) returns the current state of a given LIN channel.

7.6 Error classification

The error classification depends on the time of error occurrence according to product life cycle:

- **Development Errors**
Those errors shall be detected and fixed during development phase. In most cases, those errors are software errors. The detection of errors that shall only occur during development can be switched off for production code (by static configuration namely pre-processor switches).

- Production Errors

Those errors are hardware errors and software exceptions that cannot be avoided and are also expected to occur in production code.

[LIN046] 「Values for production code Event Ids are assigned externally by the configuration of the [DEM](#). They are published in the file Dem_IntErrId.h and included via Dem.h.」(BSW00409)

[LIN047] 「Development error values are of type uint8.」(BSW00304)

[LIN048] 「The following errors and exceptions shall be detectable by the LIN driver depending on its build version (development/production mode) 」 (BSW00323, BSW00327, BSW00337, BSW00385)

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

[LIN213] 「 The LIN Driver module shall report the development error "LIN_E_STATE_TRANSITION (0x04)", when Invalid state transition occurs from the current state.」()

[LIN214] 「The LIN Driver module shall report the development error "LIN_E_UNINIT (0x00)", when the API Service is used without module initialization.」()

[LIN215] 「 The LIN Driver module shall report the development error "LIN_E_INVALID_CHANNEL (0x02)", when API Service used with an invalid or inactive channel parameter.」()

[LIN216] 「 The LIN Driver module shall report the development error "LIN_E_INVALID_POINTER (0x03)", when API Service is called with invalid configuration pointer.」()

[LIN249] 「The LIN Driver module shall report the development error "LIN_E_PARAM_POINTER (0x05)", when API Service is called with a NULL pointer. In case of this error, the API service shall return immediately without any further action, beside reporting this development error.」()

[LIN218] 「The LIN Driver module shall report the production or development error "LIN_E_TIMEOUT (value assigned by DEM)", when Timeout caused by hardware error.」()

[LIN237] 「If the LIN module detects an error and calls the Development Error Tracer, the LIN module's function that raised the development error shall return immediately」 (BSW12448)

7.7 Error detection

[LIN049] 「The detection of development errors is configurable (ON/OFF) at pre-compile time. The switch *LinDevErrorDetect* (see chapter 10) shall activate or deactivate the detection of all development errors.」(BSW00323, BSW00338)

[LIN050] 「If the *LinDevErrorDetect* switch is enabled API parameter checking is enabled.」()

[LIN051] 「The detection of production code errors cannot be switched off.」()

[LIN097] 「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.

7.8 Error notification

[LIN052] 「Detected development errors shall be reported to the *Det_ReportError* service of the Development Error Tracer ([DET](#)) if the pre-processor switch *LinDevErrorDetect* is set (see chapter 10).」(BSW00338, BSW157, BSW12448, BSW01526)

[LIN058] 「Production errors shall be reported to Diagnostic Event Manager (DEM) by calling the function Dem_ReportErrorStatus. The only production error that can be reported by the LIN driver is the LIN_E_TIMEOUT error. 」(BSW00421)

7.9 Debugging

[LIN186] 「Each variable that shall be debugged or traced by AUTOSAR shall be defined as global variable. 」()

[LIN187] 「All type definitions of variables which shall be debugged, shall be accessible by the header file Lin.h. 」()

[LIN188] 「It shall be possible to calculate the size of elements by c-“sizeof” operation. 」()

[LIN189] 「Variables available for debugging shall be described in the respective Basic Software Module Description. 」()

8 API specification

8.1 Imported types

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

[LIN226] 「

<i>Module</i>	<i>Imported Type</i>
Dem	Dem_EventIdType
	Dem_EventStatusType
EcuM	EcuM_WakeupSourceType
Icu	Icu_ChannelType
Std_Types	Std_ReturnType
	Std_VersionInfoType

」()

8.2 Type definitions

[LIN244] 「The content of `Lin_GeneralTypes.h` consists of types specified within the LIN Interface [8] and LIN Transceiver Driver SWS [15] and the following type specifications within this document except for Config Types (e.g. `Lin_ConfigType`, `LinTp_ConfigType` e.t.c.).」()

[LIN245] 「The content of `Lin_GeneralTypes.h` shall be protected by a `LIN_GENERAL_TYPES` define.」()

[LIN246] 「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

[LIN247] 「`Lin_ConfigType` shall be provided by the headerfile `Lin.h`.」()

[LIN227] 「

Name:	<code>Lin_ConfigType</code>	
Type:	Structure	
Range:	Hardware and Implementation dependent structure	The contents of the initialization data structure are LIN hardware specific
Description:	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.
--	---

┘()

8.2.2 Lin_FramePidType

[LIN228] ┘

Name:	Lin_FramePidType		
Type:	uint8		
Range:	0...0xFE	--	The LIN identifier (0...0x3F) together with its two parity bits.
Description:	Represents all valid protected identifier used by Lin_SendFrame().		

┘()

8.2.3 Lin_FrameCsModelType

[LIN229] ┘

Name:	Lin_FrameCsModelType		
Type:	Enumeration		
Range:	LIN_ENHANCED_CS		Enhanced checksum model
	LIN_CLASSIC_CS		Classic checksum model
Description:	This type is used to specify the Checksum model to be used for the LIN Frame.		

┘()

8.2.4 Lin_FrameResponseType

[LIN230] ┘

Name:	Lin_FrameResponseType		
Type:	Enumeration		
Range:	LIN_MASTER_RESPONSE		Response is generated from this (master) node
	LIN_SLAVE_RESPONSE		Response is generated from a remote slave node
	LIN_SLAVE_TO_SLAVE		Response is generated from one slave to another slave, for the master the response will be anonymous, it does not have to receive the response.
Description:	This type is used to specify whether the frame processor is required to transmit the response part of the LIN frame.		

┘()

8.2.5 Lin_FrameDlType

[LIN231] ┘

Name:	Lin_FrameDlType		
Type:	uint8		
Range:	1...8	--	Data length of a LIN Frame
Description:	This type is used to specify the number of SDU data bytes to copy.		

┘()

8.2.6 Lin_PduType

[LIN232] ▮

Name:	Lin_PduType		
Type:	Structure		
Element:	Lin_FramePidType	Pid	--
	Lin_FrameCsModelType	Cs	--
	Lin_FrameResponseType	Drc	--
	Lin_FrameDlType	Dl	--
	uint8*	SduPtr	--
Description:	This Type is used to provide PID, checksum model, data length and SDU pointer from the LIN Interface to the LIN driver.		

▮()

Description for each element of Lin_PduType is given in:

- Section 8.2.2 for Lin_FramePidType Pid
- Section 8.2.3 for Lin_FrameCsModelType Cs
- Section 8.2.4 for Lin_FrameResponseType Drc
- Section 8.2.5 for Lin_FrameDlType Dl.

8.2.7 Lin_StatusType

[LIN233] ▮

Name:	Lin_StatusType	
Type:	Enumeration	
Range:	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: - Mismatch between sent and read back data - Identifier parity error or - Physical bus error
	LIN_TX_ERROR	LIN frame operation return value. Erroneous response transmission such as: - Mismatch between sent and read back data - Physical bus error
	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: - Framing error - Overrun error - Checksum error or - Short response
	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.
Description:	LIN operation states for a LIN channel or frame, as returned by the API service Lin_GetStatus().	

」()

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

[LIN006] 「

Service name:	Lin_Init
Syntax:	void Lin_Init(const Lin_ConfigType* Config)
Service ID[hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	Config Pointer to LIN driver configuration set.
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Initializes the LIN module.

」(BSW00406, BSW101, BSW12057, BSW12125)

[LIN084] 「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.

[LIN150] 「The function Lin_Init shall initialize the module according to the configuration set pointed to by the parameter Config.」()

[LIN008] 「The function `Lin_Init` shall invoke initializations for relevant hardware register settings common to all channels available on the LIN hardware unit.」
(BSW12461, BSW01556)

[LIN190] 「The function `Lin_Init` shall also invoke initializations for LIN channel specific settings.」(BSW12125, BSW01556)

[LIN106] 「The `Lin` module's environment shall not call any function of the `Lin` module before having called `Lin_Init` except `Lin_GetVersionInfo`.」()

[LIN250] 「If wake-up is supported by hardware (i.e. `LinChannelWakeUpSupport == true`), during LIN channel initialization it shall be checked if there was a wake-up event on the specific LIN channel, (if supported by hardware). If a wake-up event has been detected, the wake-up shall directly be reported to the `Ecum` via `Ecum_SetWakeUpEvent` call-back function.」()

Symbolic names of the available configuration sets are provided by the configuration description of the LIN driver. See [chapter 10](#) about configuration description.

[LIN099] 「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`.」()

[LIN105] 「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`.」()

8.3.1.2 `Lin_CheckWakeUp`

[LIN160] 「

Service name:	<code>Lin_CheckWakeUp</code>	
Syntax:	<code>Std_ReturnType Lin_CheckWakeUp(uint8 Channel)</code>	
Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	<code>Channel</code>	LIN channel to be addressed
Parameters (inout):	None	
Parameters (out):	None	
Return value:	<code>Std_ReturnType</code>	<code>E_OK</code> : No error has occurred during execution of the API

		E_NOT_OK: An error has occurred during execution of the API
Description:	This function checks if a wakeup has occurred on the addressed LIN channel.	

」()

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 LINIF503).

[LIN098] 「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` callback function.」(BSW00375, BSW01563)

[LIN251] 「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` and return with `E_NOT_OK`.」()

[LIN107] 「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`.」()

[LIN109] 「If development error detection for the LIN module is enabled: the function `Lin_CheckWakeup` shall raise the development error `LIN_E_STATE_TRANSITION` if no LIN channel of the driver is in the `LIN_CH_SLEEP` state.」()

8.3.1.3 `Lin_GetVersionInfo`

[LIN161] 「

Service name:	<code>Lin_GetVersionInfo</code>	
Syntax:	<pre>void Lin_GetVersionInfo(Std_VersionInfoType* versioninfo)</pre>	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where is stored the version information of this module.
Return value:	None	
Description:	Returns the version information of this module.	

」()

[LIN001] 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. (BSW00407)

[LIN110] If source code for caller and callee of `Lin_GetVersionInfo` is available, the LIN module should realize `Lin_GetVersionInfo` as a macro, defined in the module's header file. ()

[LIN111] The function `Lin_GetVersionInfo` shall configurable On/Off by the configuration parameter: `LinVersionInfoApi`. ()

[LIN248] 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`. ()

[LIN219] The function `Lin_GetVersionInfo` shall be pre-compile time configurable On/Off. ()

8.3.2 Services affecting a single LIN channel

8.3.2.1 `Lin_SendFrame`

[LIN191] ()

Service name:	<code>Lin_SendFrame</code>	
Syntax:	<pre>Std_ReturnType Lin_SendFrame(uint8 Channel, Lin_PduType* PduInfoPtr)</pre>	
Service ID[hex]:	0x04	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Channel	LIN channel to be addressed
	PduInfoPtr	Pointer to PDU containing the PID, checksum model, response type, DI and SDU data pointer
Parameters	None	

(inout):	
Parameters (out):	None
Return value:	Std_ReturnType E_OK: Send command has been accepted. E_NOT_OK: Send command has not been accepted, development or production error occurred.
Description:	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.

」()

[LIN192] 「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.」()

[LIN193] 「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().」()

[LIN194] 「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.」()

[LIN239] 「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.」()

[LIN240] 「In case of response transmission errors, the LIN 2.1 specification describes 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, LIN 2.1 specifies that the transmission shall be aborted.」()

[LIN195] 「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 and return with E_NOT_OK.」()

[LIN197] 「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 and return with E_NOT_OK.」()

[LIN198] 「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 and return with E_NOT_OK.」()

[LIN199] 「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 and return with E_NOT_OK.」()

8.3.2.2 Lin_GoToSleep

[LIN166] 「

Service name:	Lin_GoToSleep	
Syntax:	Std_ReturnType Lin_GoToSleep(uint8 Channel)	
Service ID[hex]:	0x06	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Channel	LIN channel to be addressed
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Sleep command has been accepted E_NOT_OK: Sleep command has not been accepted, development or production error occurred
Description:	The service instructs the driver to transmit a go-to-sleep-command on the addressed LIN channel.	

」()

[LIN089] 「The function Lin_GoToSleep shall send a go-to-sleep-command on the addressed LIN channel as defined in LIN Specification 2.1.」()

[LIN073] 「The function Lin_GoToSleep shall set the channel state to LIN_CH_SLEEP, even in case of an erroneous transmission of the go-to-sleep-command.」(BSW01566)

[LIN220] 「The function `Lin_GoToSleep` shall enable the wake-up detection, even in case of an erroneous transmission of the go-to-sleep-command.」()

[LIN221] 「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.」()

[LIN034] 「The LIN channel shall enter `LIN_CH_SLEEP` state upon completion of the go-to-sleep-command, even in case of an erroneous transmission.」(BSW01560, BSW01566)

[LIN074] 「The function `Lin_GoToSleep` shall terminate ongoing frame transmission of prior transmission requests, even if the transmission is unsuccessfully completed.」()

[LIN129] 「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`.」()

[LIN131] 「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.」()

[LIN132] 「If development error detection for the LIN module is enabled: the function `Lin_GoToSleep` shall raise the development error `LIN_E_STATE_TRANSITION` if the LIN channel state-machine is in the state `LIN_CH_SLEEP`.」()

8.3.2.3 `Lin_GoToSleepInternal`

[LIN167] 「

Service name:	<code>Lin_GoToSleepInternal</code>	
Syntax:	<code>Std_ReturnType Lin_GoToSleepInternal(uint8 Channel)</code>	
Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Channel	LIN channel to be addressed
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	<code>E_OK</code> : Command has been accepted <code>E_NOT_OK</code> : Command has not been accepted,

	development or production error occurred
Description:	Sets the channel state to LIN_CH_SLEEP, enables the wake-up detection and optionally sets the LIN hardware unit.

」()

[LIN095] 「The function Lin_GoToSleepInternal shall set the channel state to LIN_CH_SLEEP.」()

[LIN222] 「The function Lin_GoToSleepInternal shall enable the wake-up.」()

[LIN223] 「The function Lin_GoToSleepInternal shall optionally set the LIN hardware unit to reduced power operation mode (if supported by HW).」()

[LIN133] 「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.」()

[LIN135] 「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.」()

[LIN136] 「If development error detection for the LIN module is enabled: the function Lin_GoToSleepInternal shall raise the development error LIN_E_STATE_TRANSITION if the LIN channel state-machine is in the state LIN_CH_SLEEP.」()

8.3.2.4 Lin_Wakeup

[LIN169] 「

Service name:	Lin_Wakeup	
Syntax:	Std_ReturnType Lin_Wakeup(uint8 Channel)	
Service ID[hex]:	0x07	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Channel	LIN channel to be addressed
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Wake-up request has been accepted E_NOT_OK: Wake-up request has not been accepted, development or production error occurred
Description:	Generates a wake up pulse.	

」()

[LIN090] 「The function Lin_Wakeup shall generate a wake up pulse on the addressed LIN channel.」()

[LIN154] 「The Lin driver's environment shall only call Lin_Wakeup when the LIN channel is in state LIN_CH_SLEEP.」()

[LIN137] 「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.」()

[LIN139] 「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.」()

[LIN140] 「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.」()

8.3.2.5 Lin_GetStatus

[LIN168] 「

Service name:	Lin_GetStatus	
Syntax:	<pre>Lin_StatusType Lin_GetStatus(uint8 Channel, uint8** Lin_SduPtr)</pre>	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Channel	LIN channel to be checked
Parameters (inout):	None	
Parameters (out):	Lin_SduPtr	Pointer to pointer to a shadow buffer or memory mapped LIN Hardware receive buffer where the current SDU is stored.
Return value:	Lin_StatusType	<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"> - Mismatch between sent and read back data - Identifier parity error or Physical bus error <p>LIN_TX_ERROR: Erroneous response transmission such as:</p> <ul style="list-style-type: none"> - Mismatch between sent and read back data

	<p>- Physical bus error</p> <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"> - Framing error - Overrun error - Checksum error or Short response <p>LIN_RX_NO_RESPONSE: No response byte has been received so far</p> <p>LIN_OPERATIONAL: Normal operation; the related LIN channel is just initialized or waked 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>
Description:	Gets the status of the LIN driver.

⌋()

[LIN091] ⌈The function `Lin_GetStatus` shall return the current transmission, reception or operation status of the LIN driver.⌋()

[LIN200] ⌈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`.⌋()

[LIN092] ⌈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.⌋()

[LIN238] ⌈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.⌋()

[LIN141] ⌈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` and return `LIN_NOT_OK`.⌋()

[LIN143] ¶ 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` and return `LIN_NOT_OK`. ¶()

[LIN144] ¶ 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` and return `LIN_NOT_OK`. ¶()

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.

[LIN234] ¶

<i>API function</i>	<i>Description</i>
<code>Dem_ReportErrorStatus</code>	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function.
<code>EcuM_SetWakeupEvent</code>	Sets the wakeup event.

¶()

8.6.2 Optional Interfaces

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

[LIN235] ¶

<i>API function</i>	<i>Description</i>
<code>Det_ReportError</code>	Service to report development errors.

EcuM_CheckWakeup	This callout is called by the EcuM to poll a wakeup source. It shall 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_DisableNotification	This function disables the notification of a channel.
Icu_EnableNotification	This function enables the notification on the given channel.

」()

[LIN176] 「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

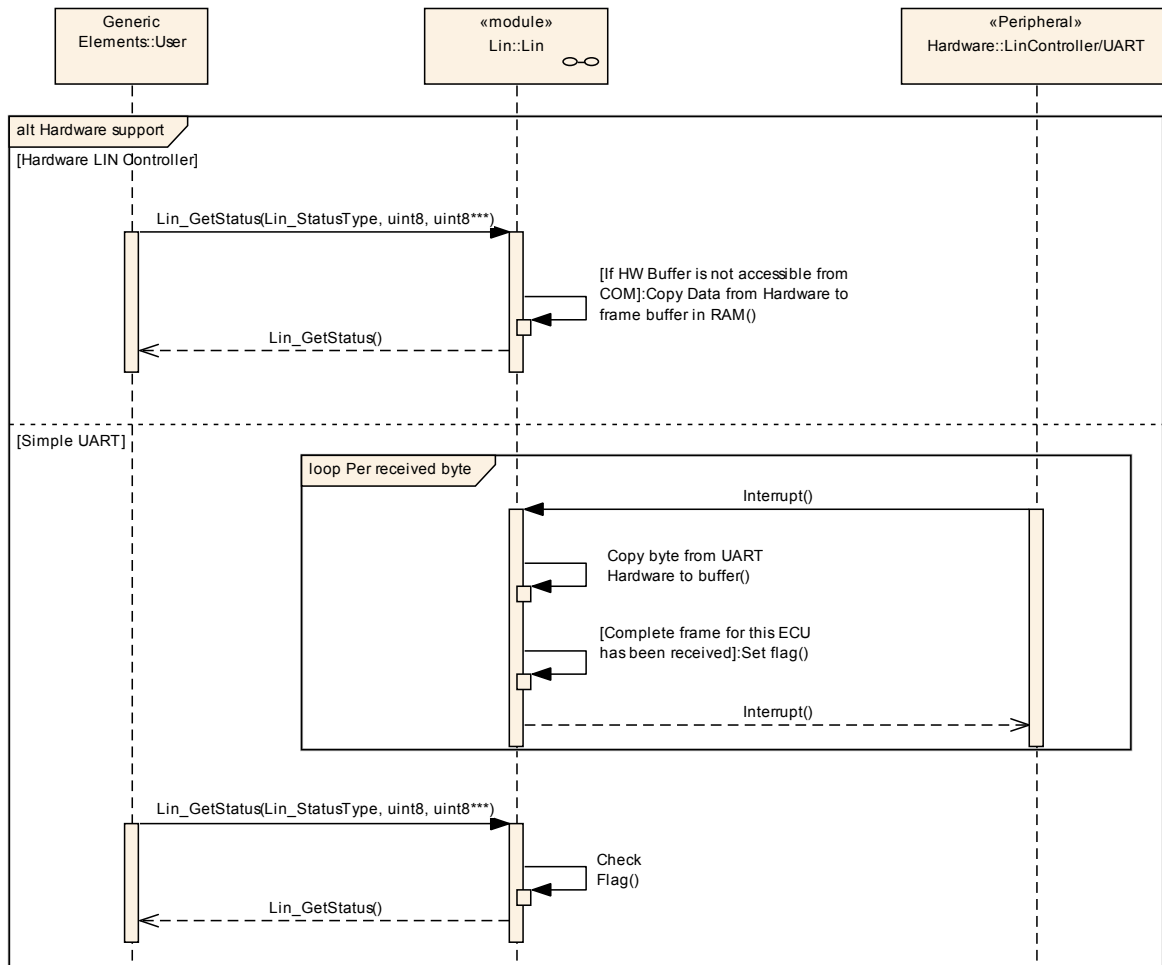


Figure 9-1: LIN Frame Receiving Sequence Chart

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

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

- AUTOSAR Layered Software Architecture [2]
- AUTOSAR ECU Configuration Specification [9]
This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

In the following tables the configuration class per configuration parameter is specified. In fact, it is important to distinguish between the configuration-classes, because they will result in different implementations and design processes.

- Pre-compile time - specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

Label	Description
X	The configuration parameter shall be of configuration class <i>Pre-compile time</i> .
--	The configuration parameter shall never be of configuration class <i>Pre-compile time</i> .

- Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	Description
X	The configuration parameter shall be of configuration class <i>Link time</i> .
--	The configuration parameter shall never be of configuration class <i>Link time</i> .

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	Description
X	The configuration parameter shall be of configuration class <i>Post Build</i> and no specific implementation is required.
L	<i>Loadable</i> - the configuration parameter shall be of configuration class <i>Post Build</i> and only one configuration parameter set resides in the ECU.
M	<i>Multiple</i> - the configuration parameter shall be of configuration class <i>Post Build</i> and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class <i>Post Build</i> .

10.1.2 Variants

Variants describe sets of configuration parameters. E.g., Variant 1: only pre-compile time configuration parameters; Variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant a parameter can only be of one configuration class.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

Configuration parameters will be clustered into one container whenever

- the configuration parameters logically belong together (e.g., general parameters which are valid for the entire module)
- the configuration parameters need to be instantiated (e.g., parameters of a LIN cluster – those parameters must be instantiated for each LIN channel separately)

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The described parameters are input for the LIN driver configurator.

[LIN029] 「The code configurator of the LIN driver is LIN hardware Unit specific.」 (BSW159)

[LIN031] 「The configuration data shall have a symbolic format that is human readable and understandable.」(BSW160)

[LIN039] 「Values that can be configured are hardware dependent. Therefore, the rules and constraints cannot be given in the standard. 」(BSW167)

[LIN224] 「The configuration tool is responsible to do a static configuration checking, also regarding dependencies between modules (e.g. Port driver, MCU driver etc.)」()

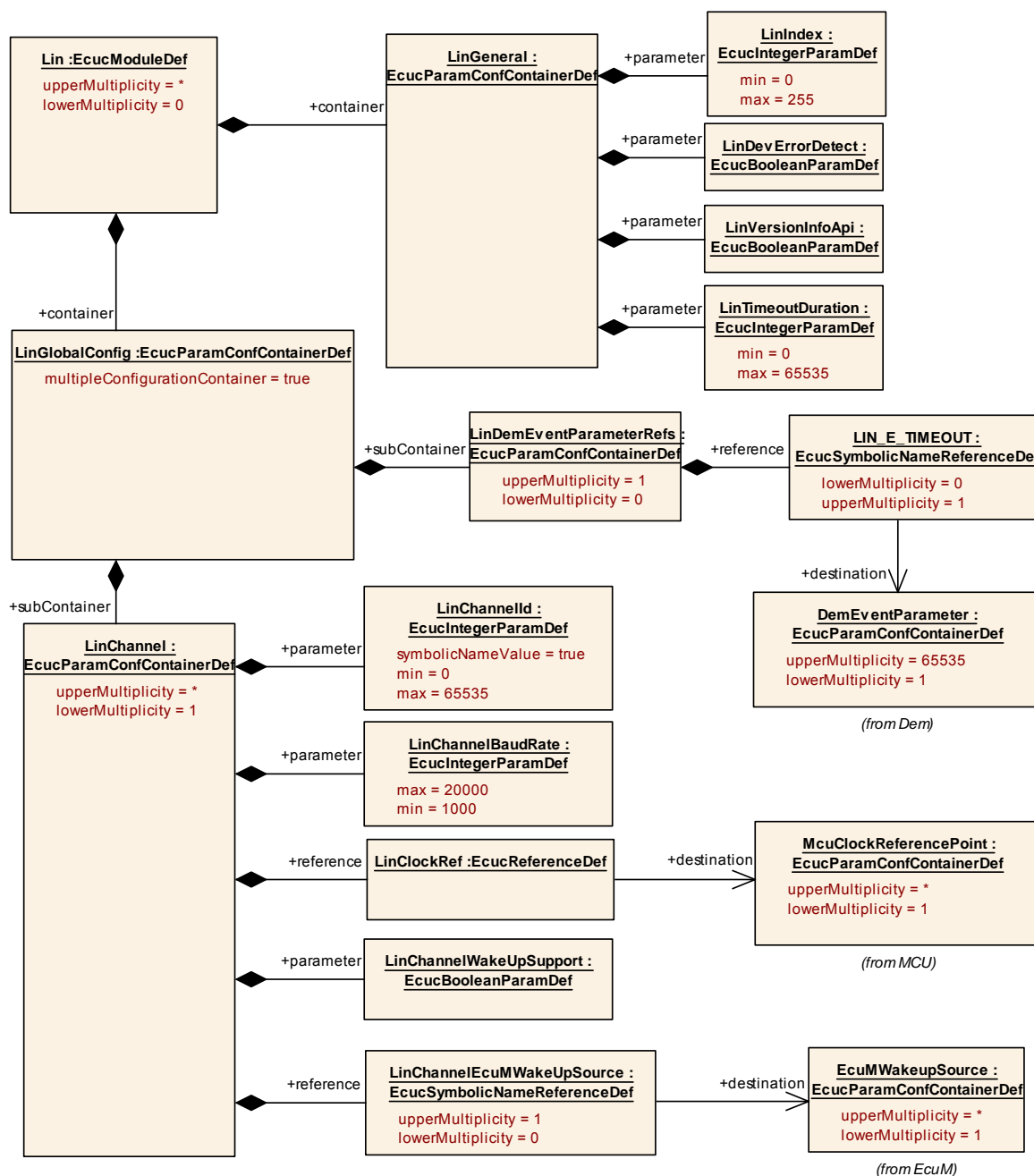


Figure 10-1: Configuration structure for the LIN driver

10.2.1 Variants

Two configuration variants are defined for the LIN driver:

[LIN103] [VARIANT-PRE-COMPILE: Only parameters with "Pre-compile time" configuration are allowed in this variant.]()

[LIN104] [VARIANT-POST-BUILD: Parameters with "Pre-compile time", "Link time" and "Post-build time" are allowed in this variant.]()

10.2.2 Lin

Module Name	Lin
Module Description	Configuration of the Lin (LIN driver) module.

Included Containers		
Container Name	Multiplicity	Scope / Dependency
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. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exit once per configuration set.

10.2.3 LinGeneral

SWS Item	LIN183_Conf :
Container Name	LinGeneral
Description	This container contains the parameters related to each LIN Driver Unit.
Configuration Parameters	

SWS Item	LIN066_Conf :		
Name	LinDevErrorDetect {LIN_DEV_ERROR_DETECT}		
Description	Switches the Development Error Detection and Notification ON or OFF.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Lin module		

SWS Item	LIN179_Conf :		
Name	LinIndex {LIN_INDEX}		
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Lin module		

SWS Item	LIN093_Conf :		
Name	LinTimeoutDuration {LIN_TIMEOUT_DURATION}		
Description	Specifies the maximum number of loops for blocking function until a timeout is raised in short term wait loops		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		

ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Lin module		

SWS Item	LIN067_Conf :		
Name	LinVersionInfoApi {LIN_VERSION_INFO_API}		
Description	Switches the Lin_GetVersionInfo function ON or OFF.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Lin module		

No Included Containers

10.2.4 LinChannel

SWS Item	LIN069_Conf :		
Container Name	LinChannel		
Description	This container contains the configuration (parameters) of the LIN Controller(s).		
Configuration Parameters			

SWS Item	LIN180_Conf :		
Name	LinChannelBaudRate {LIN_CHANNEL_BAUD_RATE}		
Description	Specifies the baud rate of the LIN channel		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1000 .. 20000		
Default value	-		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Lin module		

SWS Item	LIN181_Conf :		
Name	LinChannelId		
Description	Identifies the LIN channel. Replaces LIN_CHANNEL_INDEX_NAME from the LIN SWS.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Lin module		

SWS Item	LIN182_Conf :		
Name	LinChannelWakeupSupport {LIN_CHANNEL_WAKE_UP_SUPPORT}		
Description	Specifies if the LIN hardware channel supports wake up functionality		

Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Lin module		

SWS Item	LIN185_Conf :		
Name	LinChannelEcuMWakeupSource		
Description	This parameter contains a reference to the Wakeup Source for this controller as defined in the ECU State Manager.		
Multiplicity	0..1		
Type	Reference to [EcuMWakeupSource]		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Lin module dependency: ECU State Manager Wakeup Sources		

SWS Item	LIN094_Conf :		
Name	LinClockRef {LIN_CLOCK_SRC_REFERENCE}		
Description	Reference to the LIN clock source configuration, which is set in the MCU driver configuration.		
Multiplicity	1		
Type	Reference to [McuClockReferencePoint]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Lin module dependency: LIN clock source configuration in MCU Driver		

No Included Containers

10.2.5 LinGlobalConfig

SWS Item	LIN184_Conf :		
Container Name	LinGlobalConfig [Multi Config Container]		
Description	This container contains the global configuration parameter of the Lin driver. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.		
Configuration Parameters			

Included Containers			
Container Name	Multiplicity	Scope / Dependency	
LinChannel	1..*	This container contains the configuration (parameters) of the LIN Controller(s).	
LinDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.	

10.2.6 LinDemEventParameterRefs

SWS Item	LIN186_Conf :
Container Name	LinDemEventParameterRefs
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
Configuration Parameters	

SWS Item	LIN187_Conf :		
Name	LIN_E_TIMEOUT		
Description	Reference to the DemEventParameter which shall be issued when the error "Timeout caused by hardware error" has occurred.		
Multiplicity	0..1		
Type	Reference to [DemEventParameter]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Lin module		

No Included Containers

10.3 Published Information

[LIN252] 「 The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1].」()

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

11 Changes to Release 3

11.1 Deleted SWS Items

SWS Item	
LIN004	
LIN101	
LIN009, LIN086, LIN152, LIN178, LIN116	
LIN007, LIN112, LIN113, LIN151, LIN100, LIN114, LIN115, LIN147	
LIN153	
LIN108, LIN196, LIN130, LIN134, LIN138, LIN142	
LIN217	
LIN012, LIN158	
LIN041	
LIN102	

11.2 Replaced SWS Items

SWS Item of Release 2.1	replaced by SWS Item	
LIN164, LIN165	LIN191	
LIN087, LIN088	LIN192	
LIN122, LIN128	LIN194	
LIN117, LIN123	LIN195	
LIN118, LIN124	LIN196	
LIN119, LIN125	LIN197	
LIN120, LIN126	LIN198	
LIN121, LIN127	LIN199	

11.3 Changed SWS Items

SWS Item	
LIN075	
LIN179	
LIN176	
LIN062	
LIN174 , LIN154 , LIN160 , LIN109	
LIN130 , LIN134 , LIN138 , LIN142	
LIN075 , LIN042	
LIN084 , LIN171	
LIN103 , LIN104	
LIN169 , LIN166	
LIN045 , LIN075 , LIN042 , LIN054 , LIN065 , LIN174 , LIN053 , LIN060 , LIN043 , LIN111 ,	

LIN073 , LIN095 , LIN039	
LIN168 , LIN233	
LIN228 , LIN092 , LIN016 , LIN017 , LIN021 , LIN025	
LIN168	
LIN001	
LIN183 Conf	
LIN005 , LIN173	
LIN041	
LIN089	
LIN066 Conf , LIN179 Conf , LIN093 Conf , LIN067 Conf , LIN180 Conf , LIN181 Conf , LIN182 Conf , LIN185 Conf , LIN094 Conf	
LIN013 , LIN014 , LIN015 , LIN016 , LIN017 , LIN21 , LIN24 , LIN025 , LIN033 , LIN043 , LIN212 , LIN160 , LIN022 , LIN106	
LIN184 Conf	
LIN177	
LIN023	
LIN214	
LIN218	
LIN200 , LIN168 , LIN233	
LIN062	
LIN231 , LIN232	

11.4 Added SWS Items

SWS Item	Rationale
LIN183	
LIN184	
LIN185	
LIN186 , LIN187 , LIN188 , LIN189	
LIN190	
LIN193	
LIN200	
LIN201 , LIN202 , LIN203 , LIN204 , LIN205 , LIN206 , LIN207 , LIN208 , LIN209 , LIN210 , LIN211 , LIN212 , LIN213 , LIN214 , LIN215 , LIN216 , LIN217 , LIN218 ,	

LIN219 , LIN220 , LIN221 , LIN222 , LIN223 , LIN224	
LIN225 , LIN226 , LIN227 , LIN228 , LIN229 , LIN230 , LIN231 , LIN232 , LIN233 , LIN234 , LIN235	
LIN237	
LIN238	
LIN239 , LIN240	
LIN186 Conf , LIN187 Conf	
LIN001 PI	Rework of Published Information

12 Changes to Release 4.0 Rev1

12.1 Deleted SWS Items

SWS Item	

12.2 Replaced SWS Items

SWS Item of Release 2.1	replaced by SWS Item	

12.3 Changed SWS Items

SWS Item	
LIN062	Updated due to changes in BSW004
LIN062	
LIN177	
LIN001	
LIN048 , LIN198 , LIN144	Add missing DET error code (null pointer error)
LIN174 , LIN209 , LIN043 , LIN212 , LIN160 , LIN098 , LIN107 , LIN109 , LIN169 , LIN090 , LIN154 , LIN137 , LIN139 , LIN140 , LIN182_Conf , LIN185_Conf	Correct naming of "WakeUp" to "Wakeup" for Lin_WakeUp, Lin_WakeUpValidation, LinChannelWakeUpSupport, LinChannelEcuMWakeUpSource

12.4 Added SWS Items

SWS Item	Rationale
LIN241 , LIN242 , LIN243 , LIN244 , LIN245 , LIN246 , LIN247	
LIN248 , LIN249	Add missing DET error code (NULL pointer error)

13 Changes to Release 4.0 Rev2

13.1 Deleted SWS Items

SWS Item	

13.2 Replaced SWS Items

SWS Item of Release 4.0.2	replaced by SWS Item	

13.3 Changed SWS Items

SWS Item	
LIN225	Bug 48611 (call of Lin_Init)
[LIN168]	Correct definition of LIN_OPERATIONAL in Lin_GetStatus API according to bug #45954 , IT #49030
LIN048 , LIN218 , LIN097	Bug #51028 (LIN_E_TIMEOUT from DEM to DET/DEM error)
LIN160 , LIN098 , LIN107 , LIN109	Bug #50986 (Lin_WakeupValidation -> Lin_CheckWakeup)

13.4 Added SWS Items

SWS Item	Rationale
[LIN250]	incorporate bugzilla #46380 , IT #48920 (detection that MCU power on was caused by LIN communication)
LIN251	Bug #50986 (Lin_WakeupValidation -> Lin_CheckWakeup)

14 Not applicable requirements

[LIN999] 「 These requirements are not applicable to this specification. 」 (BSW00307, BSW00312, BSW00325, BSW00326, BSW00328, BSW00329, BSW00330, BSW00331, BSW00336, BSW00339, BSW00342, BSW00343, BSW00353, BSW00357, BSW00359, BSW00360, BSW00361, BSW00373, BSW00376, BSW00378, BSW00383, BSW00395, BSW00397, BSW00398, BSW00399, BSW00400, BSW00413, BSW00415, BSW00416, BSW00417, BSW00420, BSW00422, BSW00423, BSW00424, BSW00425, BSW00426, BSW00427, BSW00428, BSW00429, BSW00431, BSW00432, BSW00433, BSW00434, BSW005, BSW007, BSW162, BSW168, BSW12056, BSW12267, BSW12163, BSW12463, BSW12075, BSW12078, BSW12092, BSW01551, BSW01568, BSW01569, BSW01570, BSW01564, BSW01546, BSW01561, BSW01549, BSW01571, BSW01514, BSW01515, BSW01502, BSW01558, BSW01527, BSW01523, BSW01540, BSW01545, BSW01534, BSW01574, BSW01539, BSW01544, BSW01590)