Document Title	Specification of LIN Transceiver Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	257
Document Classification	Standard

Document Version	1.4.1
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History

Date	Version	Changed by	Change Description
31.03.2014	1.4.1	AUTOSAR Release Management	• Editorial Changes
31.10.2013	1.4.0	AUTOSAR Release Management	• Added intimation to LinIf for wakeup by transceiver • Modified header file structure and mandatory interfaces • Removed SWS_LinTrcv_00160 • Editorial changes • Removed chapter(s) on change documentation
21.02.2013	1.3.0	AUTOSAR Administration	• Removed SWS_LinTrcv_00141 and SWS_LinTrcv_00118 • DET Errors LINTRCV_E_PARAM_TRCV_WAKEUP_MODE and LINTRCV_E_PARAM_TRCV_OPMODE removed from SWS_LinTrcv_00050 • LINIF_TRCV_WU changed to LINTRCV_WUMODE for LINIF_TRCV_WU_ENABLE, LINIF_TRCV_WU_DISABLE and LINIF_TRCV_WU_CLEAR • Rework of configuration parameter LinTrcvDioAccess and changed scope of configuration parameters to "local" or "ECU" • Rework of header file structure • Removal of specification items covered by the new SWS BSW General • Formal rework

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
09.12.2011	1.2.0	AUTOSAR Administration	• Update of wake-up validation (power-up) • Several minor corrections (typos and wordings)
15.11.2010	1.1.0	AUTOSAR Administration	Literals changed names: 1. the imported LIN interface parameters (from LINInterface) are removed, instead 3 local parameters are introduced. • LINIF_TRCV_MODE_NORMAL -> LINTRCV_TRCV_MODE_NORMAL • LINIF_TRCV_MODE_STANDBY -> LINTRCV_TRCV_MODE_STANDBY • LINIF_TRCV_MODE_SLEEP -> LINTRCV_TRCV_MODE_SLEEP
10.12.2009	1.0.0	AUTOSAR Administration	Initial release

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction

This specification specifies functionality, API and configuration of the module LIN transceiver driver. It is responsible to handle the LIN transceiver hardware on an ECU.

A LIN bus transceiver is a hardware device. It is the interface between LIN protocol controller and physical LIN bus. On one hand the transmit data stream of a LIN protocol controller is converted into LIN physical layer compliant bus signals. On the other hand LIN bus data streams are converted into protocol controller input signals. A LIN protocol controller is typically a microcontroller implementation.

Most LIN transceivers support power supply control and wakeup via the bus. A lot of different wakeup/sleep and power supply concepts are available on the market.

In addition so called system basis chips (SBC) are available. Beside LIN transceiver functionalities these devices provide additional features, e.g. detection of electrical malfunctions (e.g. short-circuit to dominant level (GND)), power supply control, advanced watchdogs, LIN transceiver, SPI etc.

1.1 Goal of LIN transceiver driver

The target of this document is to specify interfaces and behaviour, which are applicable to most current LIN transceiver hardware implementations.

[SWS_LinTrcv_00042] 「The LIN transceiver driver abstracts the applied LIN transceiver hardware and covers hardware independent interfaces to the higher layers. It abstracts also from ECU layout by using APIs of MCAL layer to access LIN transceiver hardware.」(SRS_BSW_00162)

1.2 Explicitly uncovered LIN transceiver functionality

Some LIN bus transceivers offer additional functionality like ECU self test or error detection capability for diagnostics.

ECU self test and error detection are not defined within AUTOSAR and requiring such functionality in general would lock out most currently used transceiver hardware chips. Therefore, features like “ground shift detection”, “selective wakeup”, “slope control” and others are not supported.

2 Acronyms and abbreviations

Abbreviation	*Description*
API	Application Program Interface
Channel	A channel is a software exchange medium for data that are defined with the same criteria.
ComM	Communication Manager
Dem	Diagnostic Event Manager
Det	Development Error Tracer
Dio/DIO	Digital input output, one of the SPAL SW modules
EcuM	ECU State Manager
ECU	Electronic Control Unit
Frt	Free Running Timer
Gpt	General purpose Timer
ICU	Interrupt Control Unit
ISR	Interrupt Service Routine
LinTrcv	Lin Transceiver Driver
MCAL	Micro Controller Abstraction Layer
n/a	Not applicable
PDU	Protocol Data Unit
SBC	System Basis Chip; a device, which integrates e.g. LIN and/or LIN transceiver, watchdog and power control.
SPAL	Standard Peripheral Abstraction Layer
SW	Software
SPI	Serial Peripheral Interface
SPI Channel	A channel is a software exchange medium for data that are defined with the same criteria: configuration parameters, number of data elements with same size and data pointers (source & destination) or location. See specification of SPI driver for more details.
SPI Job	A job is composed of one or several channels with the same chip select. A job is considered to be atomic and therefore cannot be interrupted. A job has also an assigned priority. See specification of SPI driver for more details.
SPI Sequence	A sequence is a number of consecutive jobs to be transmitted. A sequence depends on a static configuration. See specification of SPI driver for more details.

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] Requirements on LIN
AUTOSAR_SRS_LIN.pdf

[5] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

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

3.2 Related standards and norms

[7] Specification of LIN Driver
AUTOSAR_SWS_LINDriver.pdf

[8] Specification of LIN Interface
AUTOSAR_SWS_LINInterface.pdf

[9] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

[10] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[11] Specification of Communication Stack Types
AUTOSAR_SWS_CommunicationStackTypes.pdf

[12] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [6] (SWS BSW General), which is also valid for LIN Transceiver Driver.

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

4 Constraints and assumptions

4.1 Limitations

The used APIs of underlying drivers like DIO or SPI shall be synchronous. Implementations of underlying drivers, which do not support synchronous behavior, cannot be used together with LIN transceiver driver.

4.2 Applicability to car domains

This driver might be applicable in all car domains using LIN for communication.

5 Dependencies to other modules

Module	*Dependencies*
LinIf	All LIN transceiver drivers are arranged below LinIf.
ComM	ComM steers LIN transceiver driver communication modes via LinIf. Independent steering of each single LIN transceiver channel is possible.
Det	Det gets development error information from LIN transceiver driver.
Dem	Dem gets production error information from LIN transceiver driver.
Dio	Dio module is used to access LIN transceiver hardware connected via ports.
EcuM	EcuM gets wakeup information from LIN transceiver driver via LinIf.
Icu	Icu module might perform LIN transceiver hardware interrupts.
Spi	Spi module is used to access LIN transceiver hardware connected via Spi

5.1 File structure

5.1.1 Naming convention for transceiver driver implementation

[SWS_LinTrcv_00070] 「In case different LIN transceiver hardware implementations are used in one ECU the function names of the different LIN transceiver drivers must be modified such that no two functions with the same names are generated. The names may be extended with a vendor ID or a type ID.」(SRS_BSW_00347)

5.1.2 Code file structure

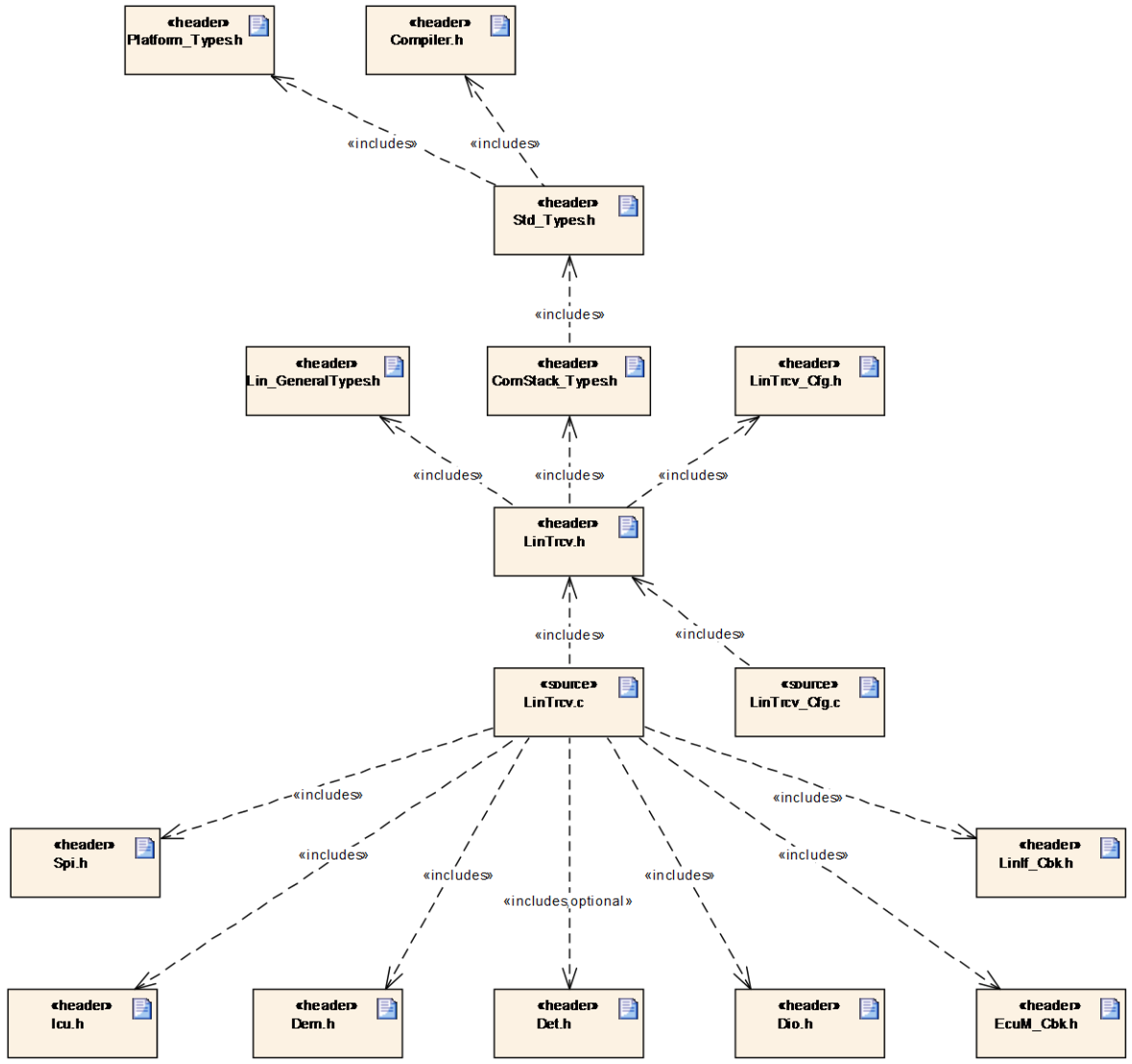
[SWS_LinTrcv_00065] 「Module consists of listed files:

File name	*Requirements*	*Description*
LinTrcv.c	LinTrcv069	The implementation general c file. It does not contain interrupt routines.
LinTrcv.h	LinTrcv052	It contains only information relevant for other BSW modules (API). Differences in API depending on configuration are encapsulated.
LinTrcv_Cfg.h	LinTrcv083	Pre-compile time configuration parameter file. It's generated by the configuration tool.
LinTrcv_Cfg.c	LinTrcv062	Pre-compile time configuration code file. It's generated by the configuration tool.

」(SRS_BSW_00346, SRS_BSW_00158)

5.1.3 Header file structure

[SWS_LinTrcv_00067] ⌈The include file structure shall be as follows
LinTrcv.c shall include Dem.h (needed to notify about production errors)
LinTrcv.c shall include Det.h (needed to notify about development errors) if development error detection for the module LinTrcv is enabled.
LinTrcv.c shall include EcuM_Cbk.h (needed to notify about wakeup of a LIN channel)
LinTrcv.c shall include Dio.h (DIO APIs needed to access Transceiver pins)
LinTrcv.c shall include LinTrcv.h (own function prototypes, defines …)
LinTrcv.c shall include Icu.h (if ICU APIs needed to perform LIN transceiver hardware interrupts)
LinTrcv.c shall include Spi.h (if the LIN bus transceiver driver use drivers for Spi to control the LIN bus transceiver hardware)
LinTrcv.c shall include Linif_Cbk.h (needed to notify about wakeup of a LIN channel)
LinTrcv_Cfg.c shall include LinTrcv.h (own function prototypes, defines …)
LinTrcv.h shall include LinTrcv_Cfg.h
LinTrcv_Cfg.h shall include ComStack_Types.h
⌋(SRS_BSW_00301, SRS_BSW_00409)



Header include structure 5-1

[SWS_LinTrcv_00068] 「For AUTOSAR standard data types header file Std_Types.h is included.」(SRS_BSW_00348)

[SWS_LinTrcv_00164] 「LinTrcv_TrcvWakeupModeType and LinTrcv_TrcvWakeupReasonTyp shall be defined in Lin_GeneralTypes.h, see also chapter “8.1 Imported types”.」()

[SWS_LinTrcv_00061] 「Name of compiler specific header file is Compiler.h. All mappings of not standardized keywords of compiler specific scope shall be placed

and organized in this compiler specific type and keyword header.」
(SRS_BSW_00361)

[SWS_LinTrcv_00063]「Name of platform specific header file is Platform_Types.h. All integer type definitions of target and compiler specific scope shall be placed and organized in this single type header.」(SRS_BSW_00353)

6 Requirements Traceability

Requirement	Description	Satisfied by
-	-	LinTrcv150_Conf
-	-	SWS_LinTrcv_00009
-	-	SWS_LinTrcv_00099
-	-	SWS_LinTrcv_00106
-	-	SWS_LinTrcv_00108
-	-	SWS_LinTrcv_00109
-	-	SWS_LinTrcv_00110
-	-	SWS_LinTrcv_00111
-	-	SWS_LinTrcv_00112
-	-	SWS_LinTrcv_00113
-	-	SWS_LinTrcv_00114
-	-	SWS_LinTrcv_00115
-	-	SWS_LinTrcv_00116
-	-	SWS_LinTrcv_00117
-	-	SWS_LinTrcv_00119
-	-	SWS_LinTrcv_00121
-	-	SWS_LinTrcv_00122
-	-	SWS_LinTrcv_00123
-	-	SWS_LinTrcv_00124
-	-	SWS_LinTrcv_00125
-	-	SWS_LinTrcv_00126
-	-	SWS_LinTrcv_00127
-	-	SWS_LinTrcv_00128
-	-	SWS_LinTrcv_00129
-	-	SWS_LinTrcv_00130
-	-	SWS_LinTrcv_00131
-	-	SWS_LinTrcv_00134
-	-	SWS_LinTrcv_00135
-	-	SWS_LinTrcv_00136
-	-	SWS_LinTrcv_00137
-	-	SWS_LinTrcv_00138
-	-	SWS_LinTrcv_00139
-	-	SWS_LinTrcv_00140
-	-	SWS_LinTrcv_00144
-	-	SWS_LinTrcv_00145
-	-	SWS_LinTrcv_00146

-	-	SWS_LinTrcv_00147
-	-	SWS_LinTrcv_00148
-	-	SWS_LinTrcv_00149
-	-	SWS_LinTrcv_00157
-	-	SWS_LinTrcv_00159
-	-	SWS_LinTrcv_00161
-	-	SWS_LinTrcv_00162
-	-	SWS_LinTrcv_00163
-	-	SWS_LinTrcv_00164
-	-	SWS_LinTrcv_00165
-	-	SWS_LinTrcv_00166
-	-	SWS_LinTrcv_00167
-	-	SWS_LinTrcv_00168
-	-	SWS_LinTrcv_00169
-	-	SWS_LinTrcv_00170
BSW00397	-	SWS_LinTrcv_00017
BSW00420	-	SWS_LinTrcv_00999
BSW00431	-	SWS_LinTrcv_00999
BSW00434	-	SWS_LinTrcv_00999
BSW01527	-	SWS_LinTrcv_00999
SRS_BSW_00005	Modules of the æC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_LinTrcv_00999
SRS_BSW_00006	The source code of software modules above the æC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_LinTrcv_00999
SRS_BSW_00007	All Basic SW Modules written in C language shall conform to the MISRA C 2004 Standard.	SWS_LinTrcv_00999
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_LinTrcv_00999
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_LinTrcv_00999
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_LinTrcv_00001
SRS_BSW_00158	All modules of the AUTOSAR Basic Software shall strictly separate configuration from implementation	SWS_LinTrcv_00065
SRS_BSW_00159	All modules of the AUTOSAR Basic Software shall support a tool based	SWS_LinTrcv_00999

	configuration	
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_LinTrcv_00999
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_LinTrcv_00042
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_LinTrcv_00999
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_LinTrcv_00999
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_LinTrcv_00999
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_LinTrcv_00999
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_LinTrcv_00067
SRS_BSW_00304	-	SWS_LinTrcv_00999
SRS_BSW_00305	Data types naming convention	SWS_LinTrcv_00999
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_LinTrcv_00999
SRS_BSW_00307	Global variables naming convention	SWS_LinTrcv_00999
SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_LinTrcv_00999
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_LinTrcv_00999
SRS_BSW_00310	API naming convention	SWS_LinTrcv_00001, SWS_LinTrcv_00002, SWS_LinTrcv_00005, SWS_LinTrcv_00007, SWS_LinTrcv_00008, SWS_LinTrcv_00012
SRS_BSW_00312	Shared code shall be reentrant	SWS_LinTrcv_00999
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_LinTrcv_00999
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_LinTrcv_00999

SRS_BSW_00326	-	SWS_LinTrcv_00999
SRS_BSW_00327	Error values naming convention	SWS_LinTrcv_00050
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_LinTrcv_00999
SRS_BSW_00329	-	SWS_LinTrcv_00001, SWS_LinTrcv_00002, SWS_LinTrcv_00005, SWS_LinTrcv_00007, SWS_LinTrcv_00008, SWS_LinTrcv_00012
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_LinTrcv_00999
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_LinTrcv_00999
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_LinTrcv_00999
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_LinTrcv_00999
SRS_BSW_00335	Status values naming convention	SWS_LinTrcv_00999
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_LinTrcv_00999
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_LinTrcv_00999
SRS_BSW_00342	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	SWS_LinTrcv_00999
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_LinTrcv_00999
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_LinTrcv_00065
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_LinTrcv_00016, SWS_LinTrcv_00070
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_LinTrcv_00068
SRS_BSW_00353	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	SWS_LinTrcv_00063
SRS_BSW_00355	-	SWS_LinTrcv_00999
SRS_BSW_00357	For success/failure of an API call a standard return type shall be defined	SWS_LinTrcv_00002
SRS_BSW_00358	The return type of init() functions	SWS_LinTrcv_00001

	implemented by AUTOSAR Basic Software Modules shall be void	
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_LinTrcv_00999
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_LinTrcv_00999
SRS_BSW_00361	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header	SWS_LinTrcv_00061
SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	SWS_LinTrcv_00002, SWS_LinTrcv_00005, SWS_LinTrcv_00007, SWS_LinTrcv_00008, SWS_LinTrcv_00012
SRS_BSW_00371	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	SWS_LinTrcv_00001, SWS_LinTrcv_00002, SWS_LinTrcv_00005, SWS_LinTrcv_00007, SWS_LinTrcv_00008, SWS_LinTrcv_00012
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_LinTrcv_00012
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_LinTrcv_00005, SWS_LinTrcv_00007
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_LinTrcv_00999
SRS_BSW_00383	The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description	SWS_LinTrcv_00999
SRS_BSW_00384	The Basic Software Module specifications shall specify at least in the description which other modules they require	SWS_LinTrcv_00999
SRS_BSW_00385	List possible error notifications	SWS_LinTrcv_00050
SRS_BSW_00386	The BSW shall specify the configuration for detecting an error	SWS_LinTrcv_00050
SRS_BSW_00398	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	SWS_LinTrcv_00999
SRS_BSW_00399	Parameter-sets shall be located in a separate segment and shall be loaded after the code	SWS_LinTrcv_00999
SRS_BSW_00400	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	SWS_LinTrcv_00999
SRS_BSW_00401	Documentation of multiple instances of	SWS_LinTrcv_00999

	configuration parameters shall be available	
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_LinTrcv_00999
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_LinTrcv_00999
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_LinTrcv_00002, SWS_LinTrcv_00007, SWS_LinTrcv_00008, SWS_LinTrcv_00012, SWS_LinTrcv_00105
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_LinTrcv_00008
SRS_BSW_00409	All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from Dem configuration	SWS_LinTrcv_00067
SRS_BSW_00410	Compiler switches shall have defined values	SWS_LinTrcv_00999
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_LinTrcv_00016
SRS_BSW_00414	The init function may have parameters	SWS_LinTrcv_00001
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_LinTrcv_00999
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_LinTrcv_00999
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_LinTrcv_00999
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_LinTrcv_00999
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_LinTrcv_00999
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_LinTrcv_00999
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_LinTrcv_00999
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_LinTrcv_00999
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_LinTrcv_00999

SRS_Can_01096	The bus transceiver driver shall provide an API to initialize the driver internally and set then all attached transceivers in their pre-selected operation modes	SWS_LinTrcv_00001
SRS_Can_01097	CAN Bus Transceiver driver API shall be synchronous	SWS_LinTrcv_00001, SWS_LinTrcv_00002, SWS_LinTrcv_00005, SWS_LinTrcv_00007, SWS_LinTrcv_00012
SRS_Can_01098	The bus transceiver driver shall support an API to send the addressed transceiver into its Standby mode	SWS_LinTrcv_00002, SWS_LinTrcv_00055
SRS_Can_01099	The bus transceiver driver shall support an API to send the addressed transceiver into its Sleep mode	SWS_LinTrcv_00002, SWS_LinTrcv_00055
SRS_Can_01100	The bus transceiver driver shall support an API to send the addressed transceiver into its Normal mode	SWS_LinTrcv_00002, SWS_LinTrcv_00055
SRS_Can_01101	The bus transceiver driver shall support an API to read out the current operation mode of the transceiver of a specified bus within the ECU	SWS_LinTrcv_00005
SRS_Can_01103	The bus transceiver driver shall support an API to read out the reason of the last wakeup of a specified bus within the ECU	SWS_LinTrcv_00007
SRS_Can_01115	The bus transceiver driver shall support an API to enable and disable the wakeup notification for each bus separately	SWS_LinTrcv_00999
SRS_Lin_01502	The LIN Interface shall support an API for RX/TX notifications.	SWS_LinTrcv_00999
SRS_Lin_01503	An API shall exist that enables the LIN driver to directly copy up to 8 byte directly from/to the frame buffers.	SWS_LinTrcv_00999
SRS_Lin_01504	The usage of AUTOSAR architecture shall be mandatory only in LIN master nodes	SWS_LinTrcv_00999
SRS_Lin_01514	The LIN Interface shall inform an upper layer about wake-up events	SWS_LinTrcv_00066
SRS_Lin_01515	The LIN Interface shall provide an API to wake-up a LIN channel cluster	SWS_LinTrcv_00999
SRS_Lin_01522	LIN-SDU shall be copied consistently for transfer	SWS_LinTrcv_00999
SRS_Lin_01523	There shall be a API call to send the LIN bus to sleep-mode.	SWS_LinTrcv_00999
SRS_Lin_01524	The LIN Driver shall be able to put the LIN hardware to a reduced power operation mode if needed	SWS_LinTrcv_00002, SWS_LinTrcv_00055
SRS_Lin_01534	The AUTOSAR LIN Transport Layer shall support half-duplex physical	SWS_LinTrcv_00999

	connections.	
SRS_Lin_01539	The Transport connection properties shall be statically configured.	SWS_LinTrcv_00999
SRS_Lin_01540	The LIN Transport Layer shall provide an API for initialization.	SWS_LinTrcv_00999
SRS_Lin_01544	Errors shall be handled	SWS_LinTrcv_00999
SRS_Lin_01545	The LIN Transport Layer services shall not be operational before initializing the module.	SWS_LinTrcv_00999
SRS_Lin_01546	The LIN Interface shall contain a Schedule Table Handler.	SWS_LinTrcv_00999
SRS_Lin_01547	The LIN Driver shall support standard UART and LIN optimized HW	SWS_LinTrcv_00999
SRS_Lin_01549	The LIN Interface needs to use a timer service for scheduling	SWS_LinTrcv_00999
SRS_Lin_01551	One LIN Interface shall support one or more LIN Drivers.	SWS_LinTrcv_00999
SRS_Lin_01552	The LIN Driver shall offer a Hardware independent interface.	SWS_LinTrcv_00999
SRS_Lin_01553	The LIN Driver shall fulfill the general SPAL requirements for Basic Software Modules.	SWS_LinTrcv_00999
SRS_Lin_01555	The LIN driver shall have an API which the driver shall use to poll for transmit / receive notifications.	SWS_LinTrcv_00999
SRS_Lin_01556	One LIN driver shall be able to handle more than one LIN channel	SWS_LinTrcv_00999
SRS_Lin_01558	The LIN Interface shall check for successful data transfer	SWS_LinTrcv_00999
SRS_Lin_01560	If a wakeup occurs during transition to sleep-mode, this channel shall go back to the running mode	SWS_LinTrcv_00999
SRS_Lin_01563	The LIN Driver shall provide a notification for wake-up events	SWS_LinTrcv_00066
SRS_Lin_01564	A Schedule Table Manager shall be available	SWS_LinTrcv_00999
SRS_Lin_01566	Transition to sleep-mode shall be handled	SWS_LinTrcv_00002, SWS_LinTrcv_00055
SRS_Lin_01568	The LIN Interface implementation and interface shall be independent from underlying LIN hardware.	SWS_LinTrcv_00999
SRS_Lin_01569	The LIN Interface shall support initialization of each LIN channel separately	SWS_LinTrcv_00999
SRS_Lin_01571	Transmission request service shall be provided	SWS_LinTrcv_00999
SRS_Lin_01572	The LIN Driver shall support the initialization of each LIN channel separately	SWS_LinTrcv_00999

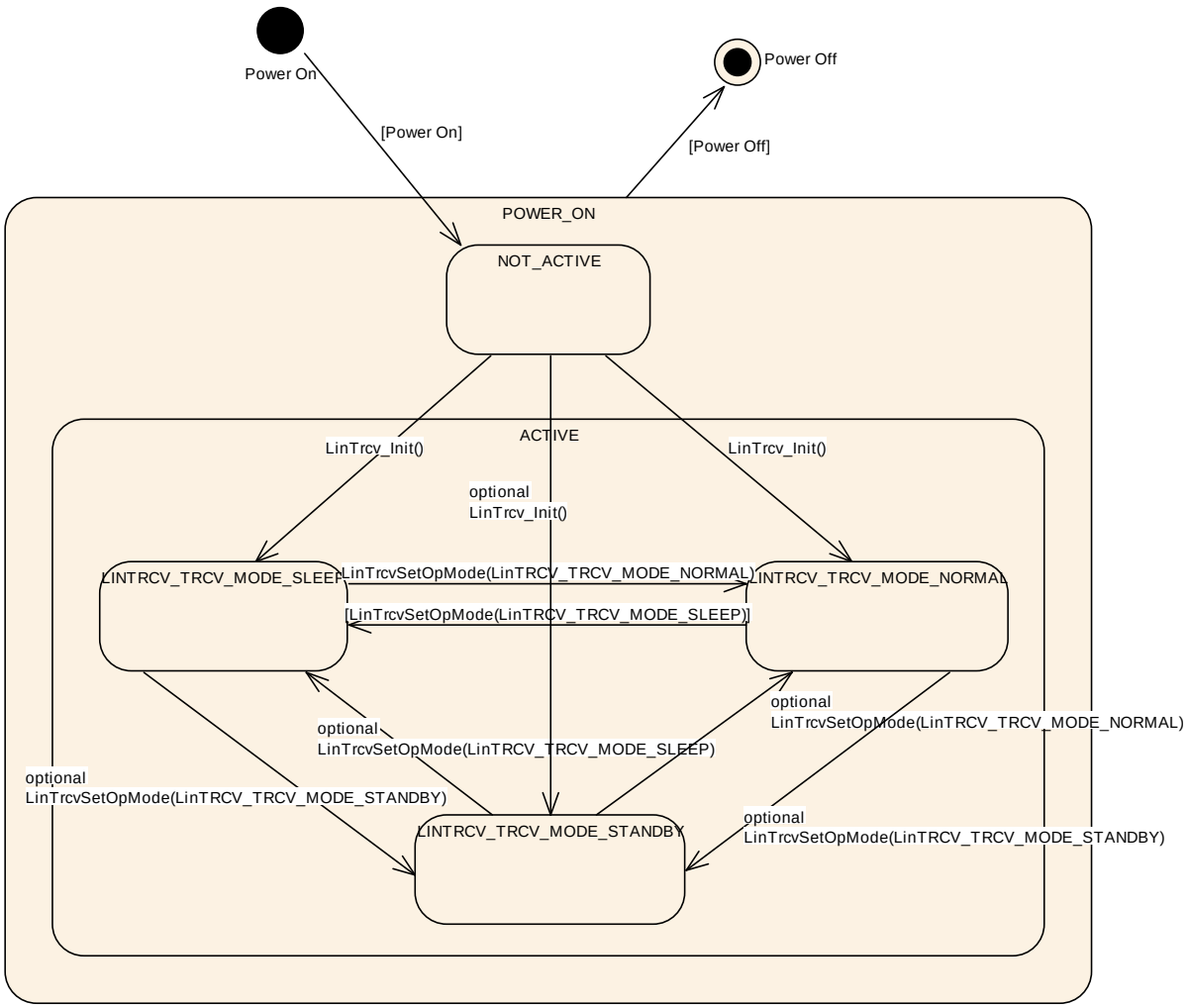
SRS_Lin_01574	It shall be possible to have one instance of the TP for each channel	SWS_LinTrcv_00999
SRS_Lin_01576	The LIN 2.1 specification shall be reused as far as possible	SWS_LinTrcv_00999
SRS_Lin_01577	It shall be compatible to LIN protocol specification	SWS_LinTrcv_00999
SRS_Lin_01579	The AUTOSAR LIN Transport Layer shall be based on the Diagnostic Transport Layer for LIN 2.1.	SWS_LinTrcv_00999
SRS_Lin_01580	The LIN Transceiver Driver shall support separate configuration parameters per bus	SWS_LinTrcv_00074, SWS_LinTrcv_00075

7 Functional specification

7.1 LIN transceiver driver operation modes

[SWS_LinTrcv_00055] 「The LIN transceiver driver operation modes are described in the state diagram below.」(SRS_Lin_01566, SRS_Lin_01524, SRS_Can_01098, SRS_Can_01099, SRS_Can_01100)

The main idea behind this diagram is to support the majority of available LIN bus transceivers in a common model view. Depending on the LIN transceiver hardware, the model may have one or two states more than necessary for a given LIN transceiver hardware, but this will clearly decouple the ComM and EcuM from the used hardware.



Lin Transceiver Operation Modes 7-1

Hint: There are several optional interfaces that might not be needed for current LIN transceiver hardware. E.g. the mode “LINTRCV_TRCV_MODE_STANDBY” might be only an internal state that is used for internal hardware transitions. Especially if functionality of “inhibit pin” is used to control the uC only the states “LINTRCV_TRCV_MODE_SLEEP” and “LINTRCV_TRCV_MODE_NORMAL” are of interest.
The function LinTrcv_Init() causes a state change to either LINTRCV_TRCV_MODE_SLEEP, LINTRCV_TRCV_MODE_NORMAL or LINTRCV_TRCV_MODE_STANDBY (any of these 3 states belong to the upper state ACTIVE). This depends on the configuration and is independent configurable for each channel.

State	Description
POWER_ON	MCU is fully powered.
NOT_ACTIVE	State of LIN transceiver hardware depend on ECU hardware and on Dio and Port driver configuration. LIN transceiver driver is not initialized and therefore not active.
ACTIVE	The function LinTrcv_Init() was called. It carries LIN transceiver driver to active state. Depending on configuration LIN transceiver driver enters state LINTRCV_TRCV_MODE_SLEEP, LINTRCV_TRCV_MODE_STANDBY or LINTRCV_TRCV_MODE_NORMAL.
LINTRCV_TRCV_MODE_NORMAL	Full bus communication. If LIN transceiver hardware controls MCU power supply, MCU is fully powered. The LIN transceiver driver detects no further wakeup information.
LINTRCV_TRCV_MODE_STANDBY	No communication is possible. If LIN transceiver hardware controls MCU power supply, the MCU is still powered. A wakeup by bus or by a local wakeup event is possible. Note: This is an optional state.
LINTRCV_TRCV_MODE_SLEEP	No communication is possible. If LIN transceiver hardware controls MCU power supply, the MCU is not powered. A wakeup by bus or by a local wakeup event is possible.

If a LIN transceiver driver covers more than one LIN channel, all channels are either in state NOT_ACTIVE or in state ACTIVE. In state ACTIVE each channel may be in a different sub state.

7.2 LIN transceiver hardware operation modes

The LIN transceiver hardware may support more mode transitions than the software. The dependencies and the recommended implementations behaviour are explained in this chapter.

It is up to the implementation to decide which LIN transceiver hardware state is covered by which LIN transceiver driver software state. An implementation has to guarantee that whole functionality of described LIN transceiver driver is given by the implementation.

7.3 LIN transceiver wakeup types

There are four different scenarios, which are often called wakeup:

1) MCU is not powered, parts of ECU including LIN transceiver hardware are powered. The considered LIN transceiver hardware is in mode LINTRCV_TRCV_MODE_SLEEP. A wakeup event on LIN is detected by LIN transceiver hardware. LIN transceiver hardware causes powering of MCU (e.g. via pin “inhibit”). In terms of AUTOSAR this is kept as a cold start and not as a wakeup.

2) MCU is in low power mode, parts of ECU including LIN transceiver hardware are powered. Depending on the hardware implementation the considered LIN transceiver hardware is either in mode LINTRCV_TRCV_MODE_STANDBY or LINTRCV_TRCV_MODE_SLEEP. A wakeup event on LIN is detected by LIN transceiver hardware. LIN transceiver hardware is informing MCU about wakeup. In terms of AUTOSAR this is kept as a wakeup of the LIN channel and of the MCU.

3) MCU is in full power mode, at least parts of the ECU including LIN transceiver hardware are powered. Depending on the hardware implementation the considered LIN transceiver hardware is either in mode LINTRCV_TRCV_MODE_STANDBY or LINTRCV_TRCV_MODE_SLEEP. A wakeup event on LIN is detected by LIN transceiver hardware. LIN transceiver hardware is informing MCU about wakeup or is polled cyclically for wakeup events. In terms of AUTOSAR this is kept as a wakeup of a LIN channel.

4) MCU is in full power mode, at least parts of the ECU including LIN transceiver hardware are powered. Depending on the hardware implementation the considered LIN transceiver hardware is either in mode LINTRCV_TRCV_MODE_STANDBY or LINTRCV_TRCV_MODE_SLEEP. The MCU is now setting the LIN transceiver hardware to mode LINTRCV_TRCV_MODE_NORMAL and is waking up the LIN channel. In terms of AUTOSAR this is kept as an internal wakeup of a LIN channel (through MCU).

7.4 LIN transceiver wakeup modes

[SWS_LinTrcv_00066] ⌈Wakeup notification must be supported by Lin Transceiver driver, therefore LIN transceiver driver covers 2 wakeup modes, internal wakeup by an upper layer or external wakeup by LIN channel.⌋(SRS_Lin_01514, SRS_Lin_01563)

1) Internal wakeup
   An internal wakeup is initiated by an upper layer, e.g. by calling LinTrcv_Init() or LinTrcv_SetOpMode.

2) External wakeup

Wakeup detected by LIN transceiver driver is forwarded to the upper layer through the API LinTrcv_CheckWakeup which has to be called by the LinIf.

Hint: WakeUp through ISR is not supported by the Lin Transceiver Driver but is only possible through ICU.

[SWS_LinTrcv_00074] ⌈Selection of wakeup mode shall be done by configuration parameter LinTrcvWakeUpSupport. (cf. LinTrcv107_Conf)⌋(SRS_Lin_01580)

[SWS_LinTrcv_00075] ⌈Support of wakeup shall be switched on and off for each LIN transceiver channel individually by configuration parameter LinTrcvWakeupByBusUsed. (cf. LinTrcv006_Conf)⌋(SRS_Lin_01580)

[SWS_LinTrcv_00161] ⌈LinTrcv driver shall use the following APIs provided by ICU driver, to enable and disable the wakeup event notification:

- Icu_EnableNotification
- Icu_DisableNotification⌋()

[SWS_LinTrcv_00162] ⌈LinTrcv driver shall enable the ICU channels when the transceiver transmits to standby mode (LINTRCV_STANDBY)⌋()

[SWS_LinTrcv_00163] ⌈LinTrcv driver shall disable the ICU channels when the transceiver transmits to Normal mode (LINTRCV_NORMAL)⌋()

Rationale: LinTrcv driver shall avoid the loss of wakeup events.

7.5 Error classification

[SWS_LinTrcv_00050] ⌈

Type or error	*Relevance*	*Related error code*	*Value [hex]*
API called with wrong parameter for LIN network	Development	LINTRCV_E_INVALID_LIN_NETWORK	0x01
API called with null pointer parameter	Development	LINTRCV_E_PARAM_POINTER	0x02
API service used without initialization	Development	LINTRCV_E_UNINIT	0x11
API service called in wrong transceiver operation mode	Development	LINTRCV_E_TRCV_NOT_SLEEP LINTRCV_E_TRCV_NOT_NORMAL	0x21 0x22
API service called with invalid mode because optional transition is not enabled	Development	LINTRCV_E_INVALID_TRCV_OPMODE	0x25

* Assignment is done in a header file of module Dem.」(SRS_BSW_00327, SRS_BSW_00385, SRS_BSW_00386)

<TrcvIdx> represents transceiver index. The symbol is generated for each transceiver that is managed in the transceiver driver module.

Remark: Development errors are notified to DET.
Production errors are notified to DEM.

7.6 Error detection

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

7.7 Error notification

[SWS_LinTrcv_00105] 「If development errors are enabled and the state of the LIN Transceiver is NOT_ACTIVE and a function is called except LinTrcv_Init or LinTrcv_GetVersionInfo the corresponding function shall raise the development error code LINTRCV_E_UNINIT.」(SRS_BSW_00406)

[SWS_LinTrcv_00106] 「If development errors are enabled and any API that uses the parameter “LinNetwork” receives an invalid value for this parameter this function shall raise the development error code LINTRCV_E_INVALID_LIN_NETWORK.」()

[SWS_LinTrcv_00159] 「If development errors are enabled and any API that uses a pointer as parameter receives a null pointer as parameter shall raise the development error code LINTRCV_E_PARAM_POINTER.」()

7.8 Debugging

For details refer to the chapter 7.1.17 “Debugging support” in *SWS_BSWGeneral.*

7.9 Preconditions for driver initialization

[SWS_LinTrcv_00099] ⌈The LIN bus transceiver driver might use drivers for Dio or Spi to control the LIN bus transceiver hardware. Thus these drivers must be available and ready to operate before the LIN bus transceiver driver is initialized.⌋()

The LIN transceiver driver may have timing requirements for the initialization sequence and the access to the transceiver device, which must be fulfilled by these used underlying drivers.

The timing requirements might be that
The call of the LIN bus transceiver driver initialization has to be performed very early after power up to be able to read all necessary information out of the transceiver hardware in time for all other users within the ECU.
The runtime of the used underlying services is very short and synchronous to enable the driver to keep his own timing requirements limited by the used hardware device.
The runtime of the driver may be enlarged, as some hardware devices have the need to have the port pin level valid for e.g. 50µs before changing it again to reach a specific state, e.g. sleep.

7.10 Instance concept

[SWS_LinTrcv_00016] ⌈For each LIN transceiver hardware type an ECU has one LIN transceiver driver instance. One instance serves all LIN transceiver hardware of the same type. ⌋(SRS_BSW_00347, SRS_BSW_00413)

7.11 Wait states

For changing operation modes, the LIN transceiver hardware may have to perform wait states.

[LinTrcv150_Conf] ⌈The wait states shall be realized with the configuration parameter
`LinTrcvWaitCount`.⌋()

7.12 Version checking

For details refer to the chapter 5.1.8 “Version Check” in *SWS_BSWGeneral.*

8 API specification

8.1 Imported types

Module	*Imported Type*
Dio	Dio_ChannelType
	Dio_LevelType
	Dio_PortLevelType
	Dio_PortType
	Dio_ChannelGroupType
EcuM	EcuM_WakeupSourceType
Icu	Icu_ChannelType
Lin_GeneralTypes	LinTrcv_TrcvWakeupModeType
	LinTrcv_TrcvWakeupReasonType
Spi	Spi_ChannelType
	Spi_DataBufferType
	Spi_NumberOfDataType
	Spi_SequenceType
	Spi_StatusType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Type definitions

8.2.1 TrcvModeType

[SWS_LinTrcv_00168] ⌜

Name:	LinTrcv_TrcvModeType	
Type:	Enumeration	
Range:	LINTRCV_TRCV_MODE_NORMAL	Transceiver mode NORMAL
	LINTRCV_TRCV_MODE_STANDBY	Transceiver mode STANDBY
	LINTRCV_TRCV_MODE_SLEEP	Transceiver mode SLEEP
Description:	Operating modes of the LIN Transceiver Driver	

⌟()

8.2.2 TrcvWakeupModeType

[SWS_LinTrcv_00169]⌜

Name:	LinTrcv_TrcvWakeupModeType	
Type:	Enumeration	
Range:	LINTRCV_WUMODE_ENABLE	The notification for wakeup events is enabled on the addressed network.
	LINTRCV_WUMODE_DISABLE	The notification for wakeup events is disabled on the addressed network.
	LINTRCV_WUMODE_CLEAR	A stored wakeup event is cleared on the addressed network.

Description:	Wake up operating modes of the LIN Transceiver Driver.

⌋()

8.2.3 TrcvWakeupReasonType

[SWS_LinTrcv_00170]⌈

Name:	LinTrcv_TrcvWakeupReasonType	
Type:	Enumeration	
Range:	LINTRCV_WU_ERROR	Due to an error wake up reason was not detected. This value may only be reported when error was reported to DEM before.
	LINTRCV_WU_NOT_SUPPORTED	The transceiver does not support any information for the wake up reason.
	LINTRCV_WU_BY_BUS	The transceiver has detected, that the network has caused the wake up of the ECU.
	LINTRCV_WU_BY_PIN	The transceiver has detected a wake-up event at one of the transceiver's pins (not at the LIN bus).
	LINTRCV_WU_INTERNALLY	The transceiver has detected, that the network has been woken up by the ECU via a request to NORMAL mode.
	LINTRCV_WU_RESET	The transceiver has detected, that the wake up is due to an ECU reset.
	LINTRCV_WU_POWER_ON	The transceiver has detected, that the wake up is due to an ECU reset after power on.
Description:	This type denotes the wake up reason detected by the LIN transceiver in detail.	

⌋()

8.3 Function definitions

8.3.1 LinTrcv_Init

[SWS_LinTrcv_00001] ⌈

Service name:	LinTrcv_Init

Syntax:	`void LinTrcv_Init(`
	`  void`
	`)`

Service ID[hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Initializes the Lin Transceiver Driver module.

⌋(SRS_BSW_00310, SRS_BSW_00329, SRS_BSW_00358, SRS_BSW_00371, SRS_BSW_00414, SRS_BSW_00101, SRS_Can_01096, SRS_Can_01097)

[SWS_LinTrcv_00119] ⌈
The function `LinTrcv_Init` shall set the LIN transceiver hardware to the state configured by the configuration parameter LinTrcvInitState.
This can be LINTRCV_TRCV_MODE_NORMAL, LINTRCV_TRCV_MODE_STANDBY or LINTRCV_TRCV_MODE_SLEEP.
⌋()

[SWS_LinTrcv_00146] ⌈
The configuration value LINTRCV_TRCV_MODE_STANDBY shall be an optional value for the configuration parameter LinTrcvInitState.
⌋()
Configuration:
Configuration parameter LinTrcvInitState specifies state after call of `LinTrcv_Init`.

Caveats:
The initialization sequence after reset (e.g. power up) is a critical phase for the LIN transceiver driver. The driver will use SPAL functionality (DIO) to access the transceiver hardware. Therefore all necessary BSW drivers must be initialized and usable before.

8.3.2 LinTrcv_SetOpMode

[SWS_LinTrcv_00002] ⌈

Service name:	LinTrcv_SetOpMode	
Syntax:	`Std_ReturnType LinTrcv_SetOpMode( uint8 LinNetwork, LinTrcv_TrcvModeType OpMode )`	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	LinNetwork	LIN network to wich API call has to be applied
	OpMode	The parameter says to which operation mode the change shall be performed.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: will be returned if the transceiver state has been changed to the requested mode. E_NOT_OK: will be returned if the transceiver state change is not accepted or has failed or the parameter is out of the allowed range.
Description:	The internal state of the LIN transceiver driver is switched to mode given in the parameter OpMode.	

⌋(SRS_BSW_00310, SRS_BSW_00329, SRS_BSW_00357, SRS_BSW_00369, SRS_BSW_00371, SRS_BSW_00406, SRS_Lin_01566, SRS_Lin_01524, SRS_Can_01097, SRS_Can_01098, SRS_Can_01099, SRS_Can_01100)

[SWS_LinTrcv_00108] ⌈The function `LinTrcv_SetOpMode` shall switch the internal state of channel LinNetwork to the value of the parameter OpMode which can be LINTRCV_TRCV_MODE_NORMAL, LINTRCV_TRCV_MODE_STANDBY or LINTRCV_TRCV_MODE_SLEEP.⌋()

[SWS_LinTrcv_00109] ⌈

The function LinTrcv_SetOpMode shall switch the internal state of channel LinNetwork to the value of LINTRCV_TRCV_MODE_STANDBY if one of the follwing conditions is fullfilled:
a) the channel LinNetwork is in mode LINTRCV_TRCV_MODE_SLEEP and the optional transition from this mode to LINTRCV_TRCV_MODE_STANDBY is enabled.
b) the channel LinNetwork is in mode LINTRCV_TRCV_MODE_NORMAL and the optional transition from this mode to LINTRCV_TRCV_MODE_STANDBY is enabled.

⌋()

[SWS_LinTrcv_00110] ⌈

The function LinTrcv_SetOpMode shall switch the internal state of channel LinNetwork to the value of LINTRCV_TRCV_MODE_SLEEP if one of

the follwing conditions is fullfilled:
a) the channel LinNetwork is in mode LINTRCV_TRCV_MODE_NORMAL
b) the channel LinNetwork is in mode LINTRCV_TRCV_MODE_STANDBY and the optional transition from this mode to LINTRCV_TRCV_MODE_SLEEP is enabled.
」()

[SWS_LinTrcv_00147] 「
The function LinTrcv_SetOpMode shall switch the internal state of channel LinNetwork to the value of LINTRCV_TRCV_MODE_NORMAL if one of the follwing conditions is fullfilled:
a) the channel LinNetwork is in mode LINTRCV_TRCV_MODE_SLEEP
b) the channel LinNetwork is in mode LINTRCV_TRCV_MODE_STANDBY and the optional transition from this mode to LINTRCV_TRCV_MODE_NORMAL is enabled.
」()

[SWS_LinTrcv_00111] 「This API is applicable to each transceiver with each value for parameter `LinTrcv_SetOpMode` regardless of whether the transceiver hardware supports these modes or not. This is to simplify the view of the LinIf to the assigned bus. 」()

[SWS_LinTrcv_00112] 「If the requested mode is not supported by the underlying transceiver hardware, the function `LinTrcv_SetOpMode` shall return E_NOT_OK.
」()

[SWS_LinTrcv_00113] 「If there is no/incorrect communication to the transceiver, the function `LinTrcv_SetOpMode` shall return `E_NOT_OK`.」()

[SWS_LinTrcv_00114] 「If development error detection for the module LinTrcv is enabled:
If the function `LinTrcv_SetOpMode` is called with OpMode == LINTRCV_TRCV_MODE_STANDBY and the channel LinNetwork is in mode LINTRCV_SLEEP but the optional transition from LINTRCV_SLEEP to LINTRCV_STANDBY is not enabled, the function `LinTrcv_SetOpMode` shall raise the development error `LINTRCV_E_INVALID_TRCV_OPMODE` and return `E_NOT_OK`.」()

[SWS_LinTrcv_00148] 「If development error detection for the module LinTrcv is enabled:
If the function `LinTrcv_SetOpMode` is called with OpMode == LINTRCV_TRCV_MODE_STANDBY and the channel LinNetwork is in mode LINTRCV_NORMAL but the optional transition from LINTRCV_NORMAL to LINTRCV_STANDBY is not enabled, the function `LinTrcv_SetOpMode` shall raise the development error `LINTRCV_E_INVALID_TRCV_OPMODE` and return `E_NOT_OK`.」()

[SWS_LinTrcv_00115] ⌈If development error detection for the module LinTrcv is enabled:
If optional transition from LINTRCV_STANDBY to LINTRCV_SLEEP is not enabled and the function `LinTrcv_SetOpMode` is called with OpMode == LINTRCV_TRCV_MODE_SLEEP and the channel LinNetwork is not in mode LINTRCV_TRCV_MODE_NORMAL, the function `LinTrcv_SetOpMode` shall raise the development error `LINTRCV_E_TRCV_NOT_NORMAL` and return `E_NOT_OK`.⌋()

[SWS_LinTrcv_00149] ⌈If development error detection for the module LinTrcv is enabled:
If optional transition from LINTRCV_STANDBY to LINTRCV_NORMAL is not enabled and the function `LinTrcv_SetOpMode` is called with OpMode == LINTRCV_TRCV_MODE_NORMAL and the channel LinNetwork is not in mode LINTRCV_TRCV_MODE_SLEEP, the function `LinTrcv_SetOpMode` shall raise the development error `LINTRCV_E_TRCV_NOT_SLEEP` and return `E_NOT_OK`.⌋()

[SWS_LinTrcv_00116] ⌈If development error detection for the module LinTrcv is enabled:
If called before the LinTrcv module has been initialized, the function `LinTrcv_SetOpMode` shall raise the development error `LINTRCV_E_UNINIT` and return `E_NOT_OK`.⌋()

[SWS_LinTrcv_00117] ⌈If development error detection for the module LinTrcv is enabled:
If called with an invalid network number LinNetwork, the function `LinTrcv_SetOpMode` shall raise the development error `LINTRCV_E_INVALID_LIN_NETWORK` and return `E_NOT_OK`.⌋()

[SWS_LinTrcv_00157] ⌈A mode request of the current mode is allowed and shall not lead to an error even if DET is enabled.⌋()

8.3.3 LinTrcv_GetOpMode

[SWS_LinTrcv_00005] ⌈

Service name:	LinTrcv_GetOpMode	
Syntax:	`Std_ReturnType LinTrcv_GetOpMode( uint8 LinNetwork, LinTrcv_TrcvModeType* OpMode )`	
Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	LinNetwork	LIN network to which API call has to be applied
Parameters (inout):	None	
Parameters (out):	OpMode	Pointer to operation mode of the bus the API is applied to.
Return value:	Std_ReturnType	E_OK: will be returned if the operation mode is detected E_NOT_OK: will be returned, if service request is failed due to development errors or the operation mode is not detected.
Description:	API detects the actual software state of LIN transceiver driver.	

⌋(SRS_BSW_00310, SRS_BSW_00329, SRS_BSW_00369, SRS_BSW_00371, SRS_BSW_00377, SRS_Can_01097, SRS_Can_01101)

[SWS_LinTrcv_00121] ⌈The function `LinTrcv_GetOpMode` shall return the actual state of the LIN transceiver driver in the parameter OpMode.⌋()

[SWS_LinTrcv_00122] ⌈If there is no/incorrect communication to the transceiver, the function `LinTrcv_GetOpMode` shall return `E_NOT_OK`.⌋()

[SWS_LinTrcv_00123] ⌈If development error detection for the module LinTrcv is enabled:
If called before the LinTrcv module has been initialized, the function `LinTrcv_GetOpMode` shall raise the development error `LINTRCV_E_UNINIT` and return `E_NOT_OK`.⌋()

[SWS_LinTrcv_00124] ⌈If development error detection for the module LinTrcv is enabled:
If called with an invalid network number LinNetwork, the function `LinTrcv_GetOpMode` shall raise the development error `LINTRCV_E_INVALID_LIN_NETWORK` and return `E_NOT_OK`.⌋()

[SWS_LinTrcv_00125] ⌈If development error detection for the module LinTrcv is enabled:

If called with OpMode == NULL, the function `LinTrcv_GetOpMode` shall raise the development error `LINTRCV_E_PARAM_POINTER` and return `E_NOT_OK`.⌋()

Configuration:
The number of supported busses is statically set in the configuration phase.

8.3.4 LinTrcv_GetBusWuReason

[SWS_LinTrcv_00007] ⌈

Service name:	LinTrcv_GetBusWuReason	
Syntax:	`Std_ReturnType LinTrcv_GetBusWuReason( uint8 LinNetwork, LinTrcv_TrcvWakeupReasonType* Reason )`	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	LinNetwork	LIN network to which API call has to be applied
Parameters (inout):	None	
Parameters (out):	Reason	Pointer to wakeup reason of the bus the API is applied to.
Return value:	Std_ReturnType	E_OK: will be returned if the wake up reason is detected E_NOT_OK: will be returned, if service request is failed due to development errors or the wakeup reason is not detected.
Description:	This API provides the reason for the wakeup that the LIN transceiver has detected in the parameter "Reason". The ability to detect and differentiate the possible wakeup reasons depends strongly on the LIN transceiver hardware.	

⌋(SRS_BSW_00310, SRS_BSW_00329, SRS_BSW_00369, SRS_BSW_00371, SRS_BSW_00377, SRS_BSW_00406, SRS_Can_01097, SRS_Can_01103)

[SWS_LinTrcv_00126] ⌈The function `LinTrcv_GetBusWuReason` shall return the reason for the wake up that the LIN transceiver has detected in the parameter Reason⌋()

[SWS_LinTrcv_00127] ⌈If there is no/incorrect communication to the transceiver, the function `LinTrcv_GetBusWuReason` shall return `E_NOT_OK`.⌋()

[SWS_LinTrcv_00128] ⌈If development error detection for the module LinTrcv is enabled:
If called before the LinTrcv module has been initialized, the function `LinTrcv_GetBusWuReason` shall raise development error `LINTRCV_E_UNINIT` and return `E_NOT_OK`.⌋()

[SWS_LinTrcv_00129] ⌈If development error detection for the module LinTrcv is enabled:

If called with an invalid network number LinNetwork, the function `LinTrcv_GetBusWuReason` shall raise development error `LINTRCV_E_INVALID_LIN_NETWORK` and return `E_NOT_OK`.⌋()

[SWS_LinTrcv_00130] ⌈If development error detection for the module LinTrcv is enabled:
If called with Reason == NULL, the function `LinTrcv_GetBusWuReason` shall raise the development error `LINTRCV_E_PARAM_POINTER` and return `E_NOT_OK`.⌋()

Configuration:
The number of supported busses is statically set in the configuration phase.

Caveats:
Be aware that if more than one bus is available each bus may report a different wakeup reason. E.g. if an ECU has LIN, a wakeup by LIN may occur and the incoming data may cause an internal wakeup for another LIN bus.
The LIN transceiver driver has a “per bus” view and does not vote the more important reason or sequence internally. The same may be true if e.g. one transceiver controls the power supply and the other is just powered or un-powered.

8.3.5 LinTrcv_GetVersionInfo

[SWS_LinTrcv_00008] ⌈

Service name:	LinTrcv_GetVersionInfo	
Syntax:	`void LinTrcv_GetVersionInfo( Std_VersionInfoType* versioninfo )`	
Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to version information of this module.
Return value:	None	
Description:	This service provides the version information of this module through the parameter "versioninfo".	

⌋(SRS_BSW_00310, SRS_BSW_00329, SRS_BSW_00369, SRS_BSW_00371, SRS_BSW_00406, SRS_BSW_00407)

[SWS_LinTrcv_00131] ⌈The function `LinTrcv_GetVersionInfo` shall return the version information of this module. The version information contains all data defined in Std_VersionInfoType in “AUTOSAR_SWS_StandardTypes”.⌋()

[SWS_LinTrcv_00134] ⌈If development error detection for the module LinTrcv is enabled:

If called with VersionInfo == NULL, the function `LinTrcv_GetVersionInfo` shall raise development error `LINTRCV_E_PARAM_POINTER` and return `E_NOT_OK`.」()

8.3.6 LinTrcv_CheckWakeup

[SWS_LinTrcv_00012] 「

Service name:	LinTrcv_CheckWakeup	
Syntax:	`Std_ReturnType LinTrcv_CheckWakeup( uint8 LinNetwork )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	LinNetwork	LIN network to which API call has to be applied.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Will be returned, if a wakeup has been detected. E_NOT_OK: Will be returned, if no wakeup has been detected
Description:	Notifies the calling function if a wakeup is detected.	

」(SRS_BSW_00310, SRS_BSW_00329, SRS_BSW_00369, SRS_BSW_00371, SRS_BSW_00375, SRS_BSW_00406, SRS_Can_01097)

[SWS_LinTrcv_00144] 「If development error detection for the module LinTrcv is enabled:
If called before the LinTrcv module has been initialized, the function `LinTrcv_CheckWakeup` shall raise the development error `LINTRCV_E_UNINIT` and return `E_NOT_OK`.」()

[SWS_LinTrcv_00145] 「If development error detection for the module LinTrcv is enabled:
If called with an invalid network number LinNetwork, the function `LinTrcv_CheckWakeup` shall raise the development error `LINTRCV_E_INVALID_LIN_NETWORK` and return `E_NOT_OK`.」()

[SWS_LinTrcv_00166]「The function `LinTrcv_CheckWakeup` shall evaluate the wakeup on the addressed LIN network. When a wake-up event on the addressed LIN network is detected (e.g. dominant bus state or negative edge at wakeup pin), the function `LinTrcv_CheckWakeup` shall notify the ECU State Manager module immediately via the `EcuM_SetWakeupEvent` and LinIf via LinIf_WakeupConfirmation callback function.」()

[SWS_LinTrcv_00167]⌈If development error detection for the module LinTrcv is enabled: If the addressed LIN network is not in mode `LINTRCV_TRCV_MODE_SLEEP`, the function `LinTrcv_CheckWakeup` shall raise the development error LINTRCV_E_TRCV_NOT_SLEEP and return E_NOT_OK.⌋()

Configuration:
See configuration parameter LinTrcvWakeUpSupport.

8.3.7 LinTrcv_SetWakeupMode

[SWS_LinTrcv_00009] ⌈

Service name:	LinTrcv_SetWakeupMode	
Syntax:	`Std_ReturnType LinTrcv_SetWakeupMode( uint8 LINNetwork, LinTrcv_TrcvWakeupModeType TrcvWakupMode )`	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	non Reentrant	
Parameters (in):	LINNetwork	LIN network to which API call has to be applied
	TrcvWakupMode	Requested transceiver wakup reason.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OKwill be returned if the transceiver state has been changed to the requested mode. E_NOT_OK will be returned, if service request is failed due to development errors or the wakeup mode is not set.
Description:	This API enables, disables and clears the notification for wakeup events on the addressed network.	

⌋()

[SWS_LinTrcv_00135] ⌈**Enabled:** If the function `LinTrcv_SetWakeupMode` is called with TrcvWakeupMode == LINTRCV_WUMODE_ENABLE and if the LinTrcv module has a stored wakeup event pending for the addressed bus, the LinTrcv module shall execute the notification within the API call or immediately after (depending on the implementation).⌋()

[SWS_LinTrcv_00136] ⌈**Disabled:** If the function `LinTrcv_SetWakeupMode` is called with TrcvWakeupMode == LINTRCV_WUMODE_DISABLE, then the notifications for wakeup events are disabled on the addressed network. It is required by the transceiver device and the underlying communication driver to detect the wakeup events and store it internally in order to raise the event when the wakeup notification is enabled again.⌋()

[SWS_LinTrcv_00137] ⌈**Clear:** If the function `LinTrcv_SetWakeupMode` is called with TrcvWakeupMode == LINTRCV_WUMODE_CLEAR, then a stored wakeup event is cleared on the addressed network. Clearing of wakeup events have to be used when the wake up notification is disabled to clear all stored wake up events under control of the higher layer.⌋()

[SWS_LinTrcv_00138] ⌈If there is no/incorrect communication to the transceiver, the function `LinTrcv_SetWakeupMode` shall return `E_NOT_OK`.⌋()

[SWS_LinTrcv_00139] ⌈If development error detection for the module LinTrcv is enabled:
If called before the LinTrcv has been initialized, the function `LinTrcv_SetWakeupMode` shall raise development error `LINTRCV_E_UNINIT` and return `E_NOT_OK`.⌋()

[SWS_LinTrcv_00140] ⌈If development error detection for the module LinTrcv is enabled:
If called with an invalid network number LinNetwork, the function `LinTrcv_SetWakeupMode` shall raise development error `LINTRCV_E_INVALID_LIN_NETWORK` and return `E_NOT_OK`.⌋()

8.4 Scheduled functions

This chaper lists all functions provided by the LinTrcv module and called directly by the Basic Software Module Scheduler. There are no cyclical called functions provided by Lin Transceiver Driver.

8.5 Call-back notifications

There are no callback notifications provided by Lin Transceiver 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.

API function	*Description*

LinIf_WakeupConfirmation	The LIN Driver or LIN Transceiver Driver will call this function to report the wake up source after the successful wakeup detection during CheckWakeup or after power on by bus.

8.6.2 Optional Interfaces

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

API function	*Description*
Det_ReportError	Service to report development errors.
Dio_ReadChannel	Returns the value of the specified DIO channel.
Dio_ReadChannelGroup	This Service reads a subset of the adjoining bits of a port.
Dio_ReadPort	Returns the level of all channels of that port.
Dio_WriteChannel	Service to set a level of a channel.
Dio_WriteChannelGroup	Service to set a subset of the adjoining bits of a port to a specified level.
Dio_WritePort	Service to set a value of the port.
EcuM_SetWakeupEvent	Sets the wakeup event.
Icu_DisableNotification	This function disables the notification of a channel.
Icu_EnableNotification	This function enables the notification on the given channel.
Spi_GetStatus	Service returns the SPI Handler/Driver software module status.
Spi_ReadIB	Service for reading synchronously one or more data from an IB SPI Handler/Driver Channel specified by parameter.
Spi_SetupEB	Service to setup the buffers and the length of data for the EB SPI Handler/Driver Channel specified.
Spi_SyncTransmit	Service to transmit data on the SPI bus
Spi_WriteIB	Service for writing one or more data to an IB SPI Handler/Driver Channel specified by parameter.

[SWS_LinTrcv_00165] ⌈LinTrcv driver shall enable/disable ICU channels only if reference is configured for the parameter `LinTrcvIcuChannelRef`.⌋()

8.6.3 Configurable interfaces

There are no configurable interfaces for LIN transceiver driver.

9 Sequence diagrams

For all wakeup related sequence diagrams please refer to chapter 9 of ECU State Manager.

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 LinTrcv.

Chapter 10.3 specifies published information of the module LinTrcv.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS_BSWGeneral.*

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described in preceding chapters.

10.2.1 Variants

Three configuration variants are defined for LIN Transceiver Driver.

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

VARIANT-LINK-TIME
The variant VARIANT-LINK-TIME shall include all configuration options of the "VARIANT-PRE-COMPILE". Additionally, all parameters that are marked as link-time configurable shall be configurable at link time. For example by linking a special configured parameter object file.

VARIANT-POST-BUILD
This configuration includes all configuration options of the "VARIANT-LINK-TIME". Additionally all parameters defined below, as post build configurable shall be configurable post build for example by flashing configuration data.

[SWS_LinTrcv_00017] ⌈
Only pre-compile time configuration is allowed.
Thus only "VARIANT-PRE-COMPILE" is allowed.⌋(BSW00397)

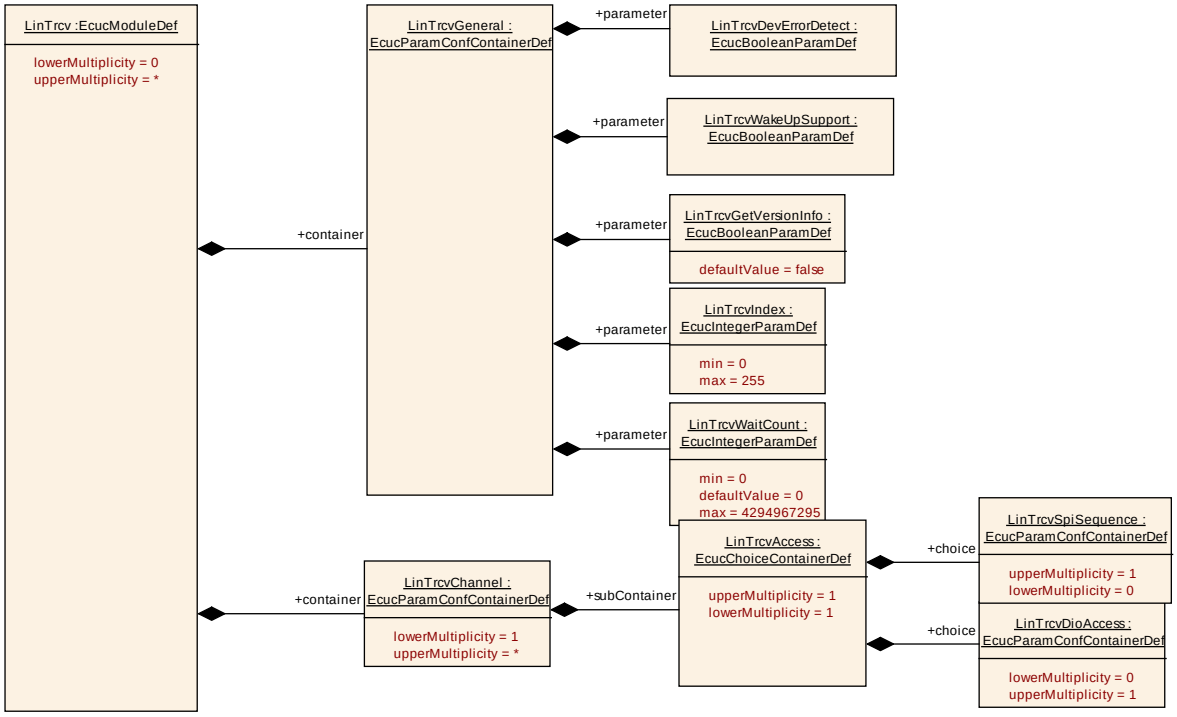
10.2.2 General configuration requirements

Configuration information is part of files LinTrcv_Cfg.h and LinTrcv_Cfg.c.

10.2.3 LinTrcv

Module Name	*LinTrcv*
Module Description	Configuration of LIN Transceiver Driver module

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
LinTrcvChannel	1..*	Container gives LIN transceiver driver information about a single LIN transceiver channel. Any LIN transceiver driver has such LIN transceiver channels.
LinTrcvGeneral	1	Container gives LIN transceiver driver basic information.



10.2.4 LinTrcvGeneral

SWS Item	**ECUC_LinTrcv_00090 :**
Container Name	LinTrcvGeneral
Description	Container gives LIN transceiver driver basic information.
Configuration Parameters	

SWS Item	**ECUC_LinTrcv_00001 :**		
Name	LinTrcvDevErrorDetect		
Description	Switches development error detection and notification on and off. If switched on, #define LINTRCV_DEV_ERROR_DETECT ON shall be generated. If switched off, #define LINTRCV_DEV_ERROR_DETECT OFF shall be generated. Define shall be part of file LinTrcv_Cfg.h. True: Is used False: Is not used		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinTrcv_00003 :**
Name	LinTrcvGetVersionInfo
Description	Switches version information API on and off. If switched off, function need not be present in compiled code. True: Is used False: Is not used
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false

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

SWS Item	**ECUC_LinTrcv_00153 :**		
Name	LinTrcvIndex		
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: local		

SWS Item	**ECUC_LinTrcv_00150 :**		
Name	LinTrcvWaitCount		
Description	Wait count for transceiver state changes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	0		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinTrcv_00107 :**		
Name	LinTrcvWakeUpSupport		
Description	Informs whether wake up is supported or not. In case wake up is not supported by LIN transceiver hardware the setting shall be false. The wake up ability may be switched on or off for each channel of one LIN transceiver by LinTrcvWakeupSourceRef. True: Is used False: Is not used		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: LinTrcvWakeupByBusUsed		

No Included Containers

10.2.5 LinTrcvChannel

SWS Item	**ECUC_LinTrcv_00091 :**
Container Name	LinTrcvChannel
Description	Container gives LIN transceiver driver information about a single LIN

	transceiver channel. Any LIN transceiver driver has such LIN transceiver channels.
Configuration Parameters	

SWS Item	**ECUC_LinTrcv_00011 :**		
Name	LinTrcvChannelId		
Description	Unique identifier of the LIN Transceiver Channel.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinTrcv_00004 :**		
Name	LinTrcvChannelUsed		
Description	Shall the related LIN transceiver channel be used? True: Is used False Is not used		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

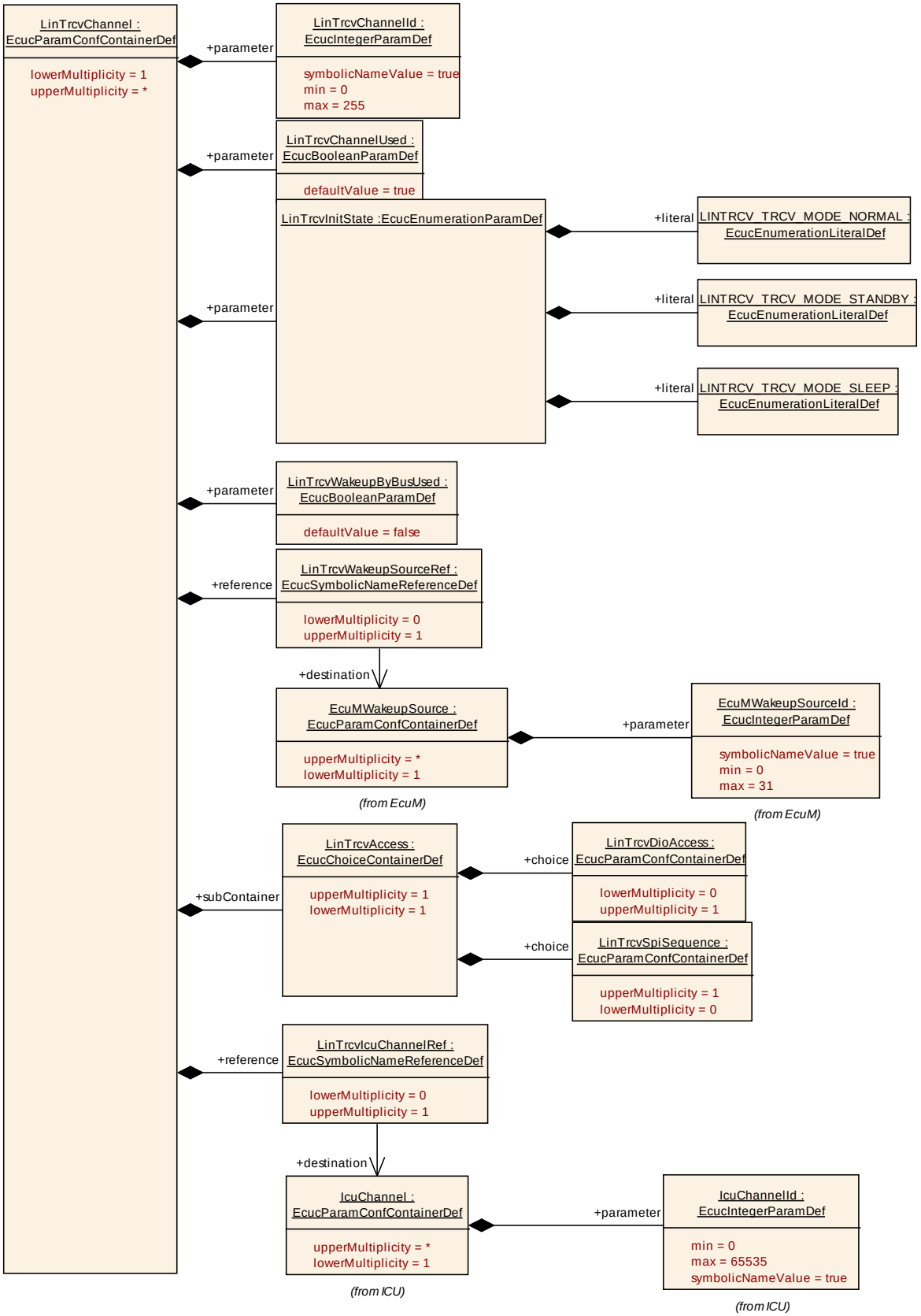
SWS Item	**ECUC_LinTrcv_00005 :**		
Name	LinTrcvInitState		
Description	State of LIN transceiver after call to LinTrcv_Init.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	LINTRCV_TRCV_MODE_NORMAL		Normal operation mode
	LINTRCV_TRCV_MODE_SLEEP		Sleep operation mode
	LINTRCV_TRCV_MODE_STANDBY		Standby operation mode
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinTrcv_00006 :**		
Name	LinTrcvWakeupByBusUsed		
Description	Is wake up by bus supported? If LIN transceiver hardware does not support wake up by bus value is always FALSE. If LIN transceiver hardware supports wake up by bus value is TRUE or FALSE depending whether it is used or not. TRUE = Is used. FALSE = Is not used.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: LinTrcvWakeUpSupport		

SWS Item	**ECUC_LinTrcv_00157 :**		
Name	LinTrcvIcuChannelRef		
Description	Reference to the IcuChannel to enable/disable the interrupts for wakeups.		
Multiplicity	0..1		
Type	Symbolic name reference to [IcuChannel]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_LinTrcv_00012 :**		
Name	LinTrcvWakeupSourceRef		
Description	Reference to a wakeup source in the EcuM configuration. This reference is only needed if LinTrcvWakeupByBusUsed is true. Implementation Type: reference to EcuM_WakeupSourceType.		
Multiplicity	0..1		
Type	Symbolic name reference to [EcuMWakeupSource]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: LinTrcvWakeupByBusUsed		

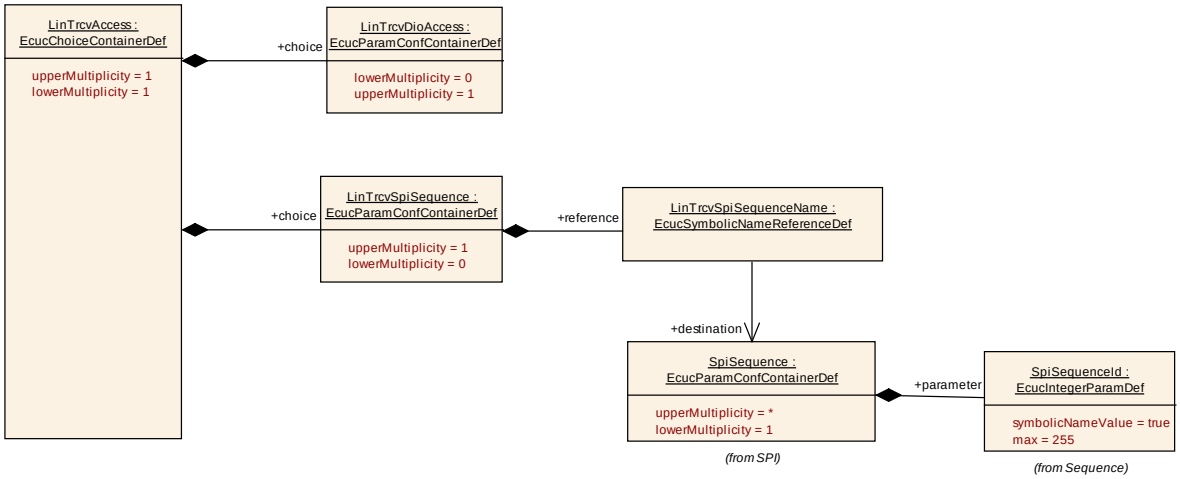
Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
LinTrcvAccess	1	Container gives LIN transceiver driver access about a single LIN transceiver channel.


LinTrcvChannel : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
LinTrcvChannelId : EcucIntegerParamDef
symbolicNameValue = true
min = 0
max = 255
+parameter
LinTrcvChannelUsed : EcucBooleanParamDef
defaultValue = true
+parameter
LinTrcvInitState :EcucEnumerationParamDef
+literal
LINTRCV_TRCV_MODE_NORMAL : EcucEnumerationLiteralDef
+literal
LINTRCV_TRCV_MODE_STANDBY : EcucEnumerationLiteralDef
+literal
LINTRCV_TRCV_MODE_SLEEP : EcucEnumerationLiteralDef
+parameter
LinTrcvWakeupByBusUsed : EcucBooleanParamDef
defaultValue = false
+reference
LinTrcvWakeupSourceRef : EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
EcuMWakeupSource : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
(from EcuM)
+parameter
EcuMWakeupSourceId : EcucIntegerParamDef
symbolicNameValue = true
min = 0
max = 31
(from EcuM)
+subContainer
LinTrcvAccess : EcucChoiceContainerDef
upperMultiplicity = 1
lowerMultiplicity = 1
+choice
LinTrcvDioAccess : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+choice
LinTrcvSpiSequence : EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 0
+reference
LinTrcvIcuChannelRef : EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
IcuChannel : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
(from ICU)
+parameter
IcuChannelId : EcucIntegerParamDef
min = 0
max = 65535
symbolicNameValue = true
(from ICU)

10.2.6 LinTrcvAccess

SWS Item	**ECUC_LinTrcv_00154 :**
Choice container Name	LinTrcvAccess
Description	Container gives LIN transceiver driver access about a single LIN transceiver channel.

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
LinTrcvDioAccess	0..1	Container gives LIN transceiver driver information about accessing ports and port pins. In addition relation between LIN transceiver hardware pin names and Dio port access information is given. If a LIN transceiver hardware has no Dio interface, there is no instance of this container.
LinTrcvSpiSequence	0..1	Container gives LIN transceiver driver information about one SPI sequence. One SPI sequence used by LIN transceiver driver is in exclusive use for it. No other driver is allowed to access this sequence. LIN transceiver driver may use one sequence to access n LIN transceiver hardwares chips of the same type or n sequences are used to access one single LIN transceiver hardware chip. If a LIN transceiver hardware has no SPI interface, there is no instance of this container.



10.2.7 LinTrcvDioAccess

SWS Item	**ECUC_LinTrcv_00094 :**
Container Name	LinTrcvDioAccess
Description	Container gives LIN transceiver driver information about accessing ports and port pins. In addition relation between LIN transceiver hardware pin names and Dio port access information is given. If a LIN transceiver hardware has no Dio interface, there is no instance of this container.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
LinTrcvDioChannelAccess	1..*	Container gives DIO channel access by single Lin transceiver channel.

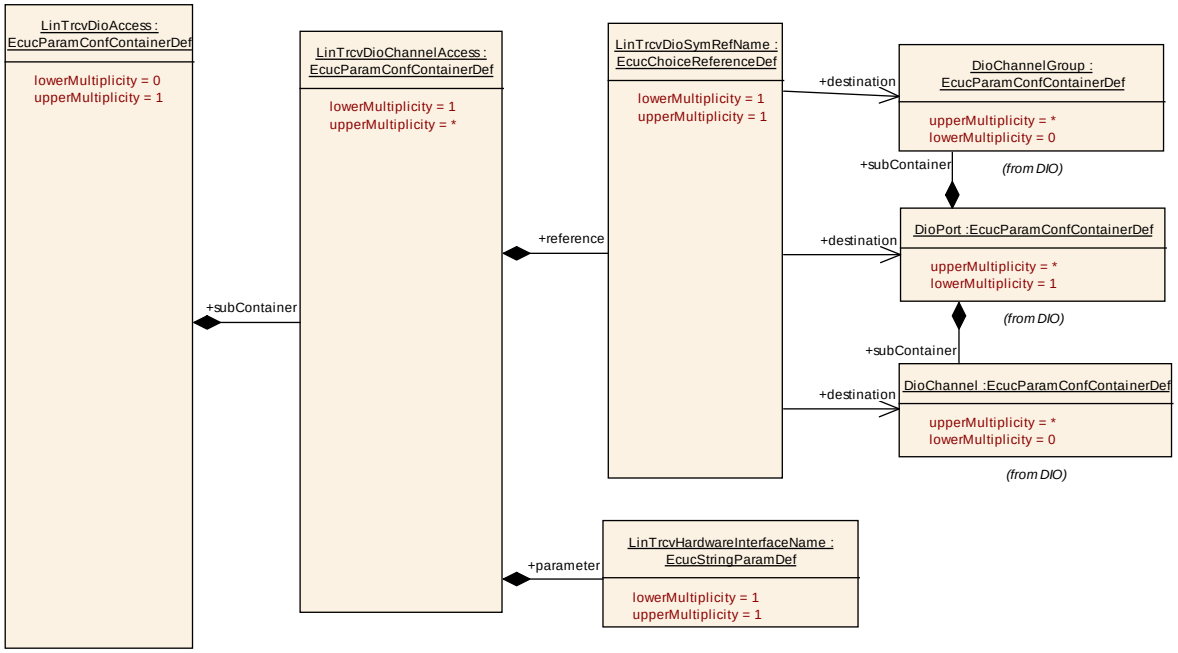
10.2.8 LinTrcvDioChannelAccess

SWS Item	**ECUC_LinTrcv_00158 :**
Container Name	LinTrcvDioChannelAccess
Description	Container gives DIO channel access by single Lin transceiver channel.
Configuration Parameters	

SWS Item	**ECUC_LinTrcv_00009 :**		
Name	LinTrcvHardwareInterfaceName		
Description	LIN transceiver hardware interface name. It is typically the name of a pin. From a Dio point of view it is either a port, a single channel or a channel group. Depending on this fact either LINTRCV_DIO_PORT_SYMBOLIC_NAME or LINTRCV_DIO_CHANNEL_SYMBOLIC_NAME or LINTRCV_DIO_CHANNEL_GROUP_SYMBOLIC_NAME shall reference a Dio configuration. The LIN transceiver driver implementation description shall list up this name for the appropriate LIN transceiver hardware.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_LinTrcv_00102 :**		
Name	LinTrcvDioSymRefName		
Description	Choice Reference to a DIO Port, DIO Channel or DIO Channel Group. This reference replaces the LINTRCV_DIO_PORT_SYM_NAME, LINTRCV_DIO_CHANNEL_SYM_NAME and LINTRCV_DIO_GROUP_SYM_NAME references in the Lin Trcv SWS.		
Multiplicity	1		
Type	Choice reference to [DioChannel , DioChannelGroup , DioPort]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers



10.2.9 LinTrcvSpiSequence

SWS Item	**ECUC_LinTrcv_00155 :**
Container Name	LinTrcvSpiSequence{LinTransceiverSPISequences}
Description	Container gives LIN transceiver driver information about one SPI sequence. One SPI sequence used by LIN transceiver driver is in exclusive use for it. No other driver is allowed to access this sequence. LIN transceiver driver may use one sequence to access n LIN transceiver hardwares chips of the same type or n sequences are used to access one single LIN transceiver hardware chip. If a LIN transceiver hardware has no SPI interface, there is no instance of this container.
Configuration Parameters	

SWS Item	**ECUC_LinTrcv_00156 :**		
Name	LinTrcvSpiSequenceName {LINTRCV_SPI_SEQUENCE_NAME}		
Description	Reference to a Spi sequence configuration container.		
Multiplicity	1		
Type	Symbolic name reference to [SpiSequence]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: SpiSequence		

No Included Containers

10.3 Published Information

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

11 Not applicable requirements

[SWS_LinTrcv_00999] ⌈These requirements are not applicable to this specification.⌋(SRS_BSW_00304, SRS_BSW_00305, SRS_BSW_00306, SRS_BSW_00307, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00312, SRS_BSW_00321, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00328, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00333, SRS_BSW_00334, SRS_BSW_00335, SRS_BSW_00336, SRS_BSW_00341, SRS_BSW_00342, SRS_BSW_00344, SRS_BSW_00355, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00378, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00401, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00410, SRS_BSW_00416, SRS_BSW_00417, BSW00420, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00429, BSW00431, SRS_BSW_00432, SRS_BSW_00433, BSW00434, SRS_BSW_00005, SRS_BSW_00006, SRS_BSW_00007, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00159, SRS_BSW_00161, SRS_BSW_00164, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00170, SRS_Lin_01576, SRS_Lin_01504, SRS_Lin_01522, SRS_Lin_01560, SRS_Lin_01577, SRS_Lin_01551, SRS_Lin_01568, SRS_Lin_01569, SRS_Lin_01564, SRS_Lin_01546, SRS_Lin_01549, SRS_Lin_01571, SRS_Lin_01515, SRS_Lin_01502, SRS_Lin_01558, BSW01527, SRS_Lin_01523, SRS_Lin_01577, SRS_Lin_01553, SRS_Lin_01552, SRS_Lin_01503, SRS_Lin_01555, SRS_Lin_01547, SRS_Lin_01572, SRS_Lin_01556, SRS_Lin_01579, SRS_Lin_01540, SRS_Lin_01545, SRS_Lin_01534, SRS_Lin_01574, SRS_Lin_01539, SRS_Lin_01544, SRS_Can_01115)